Platform Policy and its Effect on Diffusion: The Case Study of Android and iOS

Ilbae Yoon

Working Paper CISL# 2014-09

May 2014

Composite Information Systems Laboratory (CISL)
Sloan School of Management, Room E62-422
Massachusetts Institute of Technology
Cambridge, MA 02142

Platform Policy and its Effect on Diffusion:

The Case Study of Android and iOS

By

Ilbae Yoon

B.S. Physics, Yonsei University, 2003
M.S. I.E., Seoul National University, 2005
MBA, Sungkyunkwan University, 2014

SUBMITTED TO THE MIT SLOAN SCHOOL OF MANAGEMENT IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF

MASTER OF SCIENCE IN MANAGEMENT STUDIES
AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2014

Signature of Author: _____

MIT Sloan School of Management
May 9, 2014

Certified By: _____

Stuart E. Madnick
John Norris Maguire Professor of Information Technologies, MIT Sloan School of Management
And Professor of Engineering Systems, MIT School of Engineering
Thesis Supervisor

Accepted By: _____

Michael A. Cusumano
SMR Distinguished Professor of Management
Program Director, M.S. in Management Studies Program
MIT Sloan School of Management

[Page intentionally left blank]

Platform Policy and its Effect on Diffusion:

The Case Study of Android and iOS

By

Ilbae Yoon

Submitted to the MIT Sloan School of Management on May 9, 2014 in
partial fulfillment of the requirements for the degree of
Master of Science in Management Studies

ABSTRACT

Platform has played an important role in telecommunication industry. As the industry evolves from value chain to value network industry, firms can no longer pursue innovation and maintain competitiveness solely by improving value chain activities. Rather, they have to become a platform intermediary among various participants' value network. Firms that successfully built up and internalized the platform now lead the telecommunication industry: on one side, there's opened Android platform, and on the other side, there's closed iOS platform.

Each platform provider employs different platform policies, and these policies show different characteristics in the diffusion of each platform. To illustrate, the adoption rates of new operating systems are different: Android has adopted open policy, so it enjoyed relatively fast growth, but lags the adoption rate of new version's installed base far behind that of iOS. Meanwhile, iOS has adopted proprietary policy, so it the growth of user adoption was relatively slower than Android. However, frequent upgrade of Android platform resulted in so called fragmentation: while consumers enjoy wider choice of devices under opened platform, not all end users get benefit of version upgrade and this sometimes eliminates surplus generated around the platform. We see smaller fragmentation in more proprietary platform, iOS.

In this study, the difference in the diffusion patterns of open and closed platforms will be analyzed using system dynamics analysis, and then the managerial implication regarding the policy and diffusion pattern will be discussed. (1) The diffusion of those two different types of platforms will be modelled using system dynamics modeling. To incorporate open and closed policy in the model, each model will be constructed in parallel. These models will be fit into historical data and used to predict the future trend. Predicted trend will be compared with real data to evaluate the accuracy of the model. (2) As a second step, the effect of open and close platform policy on diffusion pattern will be tested with variables used in the modeling. (3) Lastly, sensitivity test will be conducted to see whether platform intermediaries or platform participants can somehow manipulate the speed of diffusion.

Thesis Supervisor: Professor Stuart Madnick
Title: John Norris Maguire Professor of Information Technologies, MIT Sloan School of Management and Professor of Engineering Systems, School of Engineering

[Page intentionally left blank]

Acknowledgement

First of all, I would like to thank my thesis advisor, Professor Stuart Madnick, for his sincere advice and support. He always encouraged me to find better ideas whenever I got astray and helped my study be right on track. Also, I'm thankful for his guide to researcher group within which I could freely interact with professionals. Interacting with research group members and sharing ideas with them was a great chance to make my idea concrete and learn a lot.

Secondly, I would like to thank my employer, Samsung Mobile, for providing me such a precious opportunity to study outside the company. Spending years in academy helped me gain diverse aspects to look at our business from broader perspective and meet with colleagues from different industries.

In addition, I express my sincere gratitude to Chanh Phan and Julia Sargeaunt in the MSMS program office, from the admission process to the graduation. Their effort to tie up MSMS students and turn diverse ethnic groups into one community helped me feel comfortable while staying overseas.

Lastly, I would like to express my special thanks to my wife and family for long lasted support. It was their love and belief that supported me to concentrate on academic life and successfully finish the study.

[Page intentionally left blank]

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Mobile industry trend

The introduction of the smartphone has dramatically transformed daily lives of consumers. With the help of seamless data network, consumers can search information or download applications. The introduction of smartphone not only changed the behaviors of consumers, but also transformed the telecommunication industry. In addition, the innovation of information and communication technology has changed key players in the industry. Once dominated by local network operators and global OEM manufacturers, the telecommunication industry now is dominated by the owners of software platforms. By constructing strong platform, these owners gathered transactions once took place in different areas around the platform. On such a reliable and standardized software platform, application providers escaped from network operators' walled garden and started to contact with more consumers through application store. Also, smartphone users have enjoyed more services and applications through platform. The whole mobile industry is under transformation to platform industry.

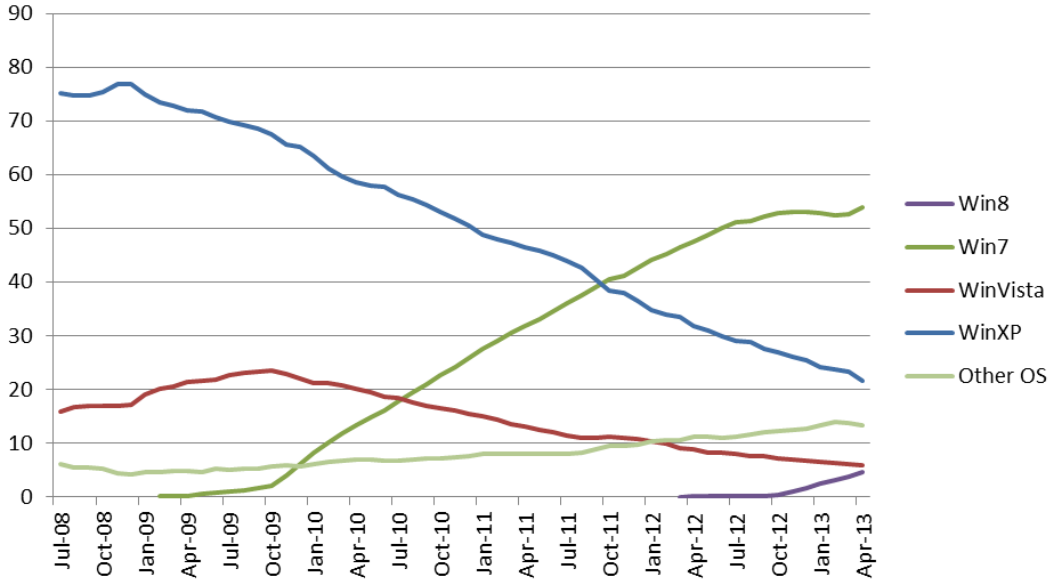


Figure 1-1. PC operating system installed base[1] (Unit: %)

Meanwhile, software platform providers have released software periodically to provide better service and usability and to differentiate own platforms from competitors. In that most of the revenue should be

[1] http://gs.statcounter.com

generated from newest copy of software, these platform providers have tried to promote new version and increase the install base as high as possible. However, the newest OS version was not always instantaneously accepted by large consumer base. As Figure 1.1 shows, even after the one year of its launch in July 2009, Windows 7 only replaced half of old users of Windows XP.

1.2. Transformation of mobile industry and mobile platform competition

Mobile platform is becoming more important because of exponentially increasing number of users and value added services around it (Nikou and Bouwman 2012). With the growing smartphone penetration, mobile platform transformed traditional mobile telephone industry into whole new industry, which is a combination of internet service and personal computing industry, and generated new opportunity for IT companies. Now that mobile platform plays an important role in such a diverse business environment, competition is fierce between entrants and incumbent companies. In that competing firms have diverse backgrounds and traditional core competencies, each other's business models and strategies are different (Kenney and Pon 2011). Interaction between these firms generated new service around mobile platform. Recently, as we observe from the competition of Amazon, Google, Facebook, and Apple, competition between IT service firms are converging into several dominant areas, such as advertisement, cloud service, and entertainment (Figure 1-2).

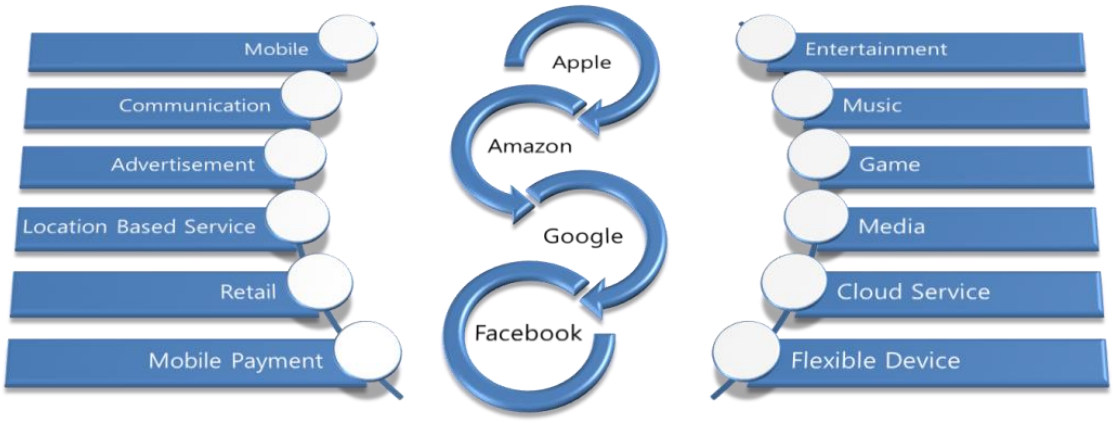


Figure 1-2.Competition around mobile industry

Therefore, traditional omni-directional value chain flowing from device manufacturers to customers has been expanded to diverse value network of industry landscape, including software developer, contents provider, media buyer, network operator, and even accessory manufacturers (Figure 1-3). By constructing

strong platform and connecting diverse service providers and consumers, mobile platforms such as Android and iOS have become a crucial element of mobile industry and made several companies to construct their own mobile platform.

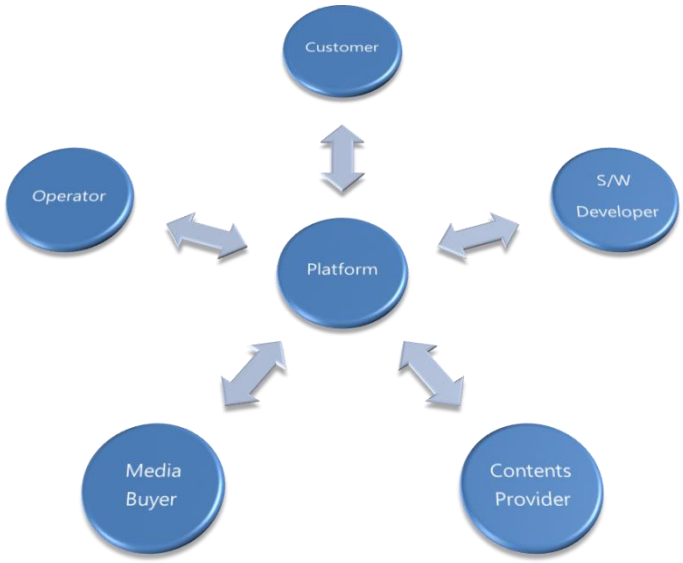


Figure 1-3. Multi-sided network in mobile platform

Now the industry is being dominated by companies that hold strong mobile platform, Apple and Google. One can observe strong platform economy in this industry. According to the platform strategy framework of Eisenmann, Parker, and Alstyne, there is still room for other firms to join the competition, because network effect is strong and switching cost is high, but differentiation is hard (Eisenmann, Parker, and Alstyne 2006). Actually, several potential entrants did not stop entering the industry. Recently, Tizen alliance led by Samsung and Intel, is preparing for a new mobile platform based on Linux and Meego. Meanwhile, although Apple enjoyed first mover advantage, google successfully differentiated itself from iOS by constructing another multisided network. In 2007, Google initiated the Open Handset Alliance and attracted diverse parties onto its platform. It differentiated itself from competitors by changing profit sharing scheme of application store and simplifying quality assurance process.

The dominance of major mobile platform was so tremendous that concentration ratio has become larger and industry is under oligopoly situation. According to IDC report[2], Android accounted for 81.0% of all smartphone shipment in 3Q of 2013, supplying more than 200 million units, and Apple accounted for

[2] http://www.idc.com

12.9%, supplying more than 33 million units (Figure 1-4). Now head of Google Android, Sundar Pichai says that their goal for 2014 would be reaching 1 billion android users. They are targeting at emerging countries, where several manufacturers are still preinstalling two year old Gingerbread version 2.3. Meanwhile, Apple is also struggling to increase installed base by introducing low end product line.

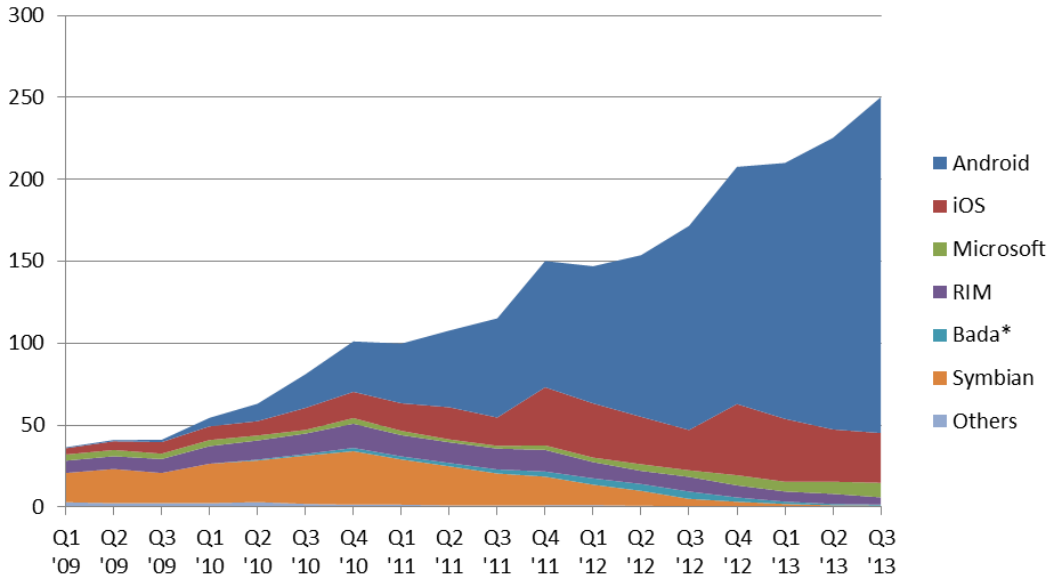


Figure 1-4. Quarterly smartphone sales by OS (Unit: million)[3]

Both firms continuously released upgraded versions to maintain growth, adding differentiated features (Figure 1-5). Newly released versions were either employed in new flagship products or customized for previously launched products. Customer's satisfaction was increased as they downloaded the new software and installed it on their products for free. The procedure for upgrading smartphone is even easier than upgrading PC, since user data are backed up on PC or cloud server and restores previously installed applications and user data automatically. Just by installing new firmware (mobile platform software or operating system designed for specific device), users can enjoy updated user interfaces and new features. However, the upgrade rate for each platform is different according to platform policy. For Android, it shows slower diffusion speed of new version than iOS. This slow conversion rate makes fragmentation worse among users and hampers OEM's or complementary product providers' decision to support version upgrade. In addition, application developers have to decide which segmentation or configuration they have to focus when they publish new applications. Meanwhile, fast software upgrade and device support

[3] http://www.statista.com

was employed as a marketing tool and marketed as a good profit generator from Apple side.

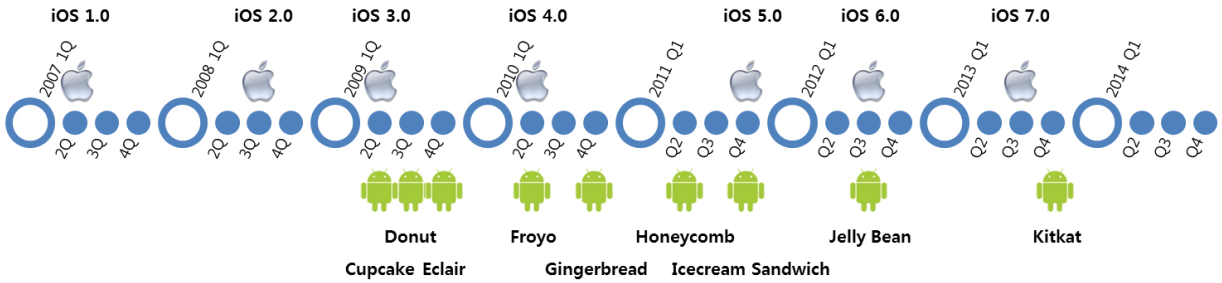


Figure 1-5. iOS and Android version history

Such difference originates from policies those two dominant mobile platform owners have employed, open and closed software development policy. While Apple has proprietary control on software release, Google does not control software release for products. Therefore, OEMs or complement product providers employing Android do not have to support software release for every product, because of constraint in cost and manpower. One the one hand, this created more choices for customer, and on the other hand, this created a "Fragmentation", where vast consumers cannot keep up with latest version of firmware.

1.3. Mobile platform fragmentation

This sluggish penetration speed of new software is apparently bad news for platform owners or OS providers. They do not get high return from investment on new product, but still have to bear cost to support old products. This slow substitution phenomenon not only undermines profits for firm itself, but sometimes harms consumers' benefit. To illustrate, when Microsoft announced that they would stop supporting Windows XP from April, 2014, the BusinessWeek reported a concern that 95% ATMs worldwide running on Windows XP won't get any security patches any more[4]. To overcome such a slow adoption from consumers, Microsoft cut its license price by 75% from $120 to $30 from Windows 8[5]. But

[4] http://www.businessweek.com/articles/2014-01-16/atms-face-deadline-to-upgrade-from-windows-xp

[5] http://www.extremetech.com/computing/150130-microsoft-cuts-the-price-of-windows-8-and-office-2013-to-spur-slow-adoption

pricing alone cannot fix the problem, since platform bears complex interaction among various platform participants and inter platform competition. No matter how such an effort is effective or not, it is still apparent that such a slow penetration of new OS has brought about concerns to platform owner.

The same phenomenon has occurred in mobile industry. Even after OEMs and OS providers released new version of smartphone firmware, sometimes the number of users installing new firmware did not increase fast. One reason might be that customers did not feel the need to install new version. Another reason would be that OEMs did not support software upgrade for some of their past products or continued developing products based on older versions to decrease hardware cost and thereby device price. Although the release interval has been larger, Google still releases Android platform several times a year, including minor upgrades. As Google has open policy, which allows OEMs to manipulate and customize the OS onto each product, even though new version is released, it takes time to release upgraded firmware for each device. Sometimes many products are excluded from software upgrade service. For iOS, situation is better because the company maintains proprietary ownership for the OS and controls every sequence from software development to final release. Nowadays this phenomenon of sluggish software update and division of software versions between users are called “Fragmentation”.

There are lots of discourses about the comparison between fragmentations of iOS and Android, but it is true that customers are frustrated when they do not get proper software upgrade for their smartphones. For OEMs, they have to invest huge resources to upgrading firmware for old products and this is burdensome when they had launched many smartphone products previously. Application developers have to redesign and attune their software to support various versions of hardware and software. Against this phenomenon, Google finally announced that they will support backward compatibility from recent version 4.4, KitKat, by lowering memory requirement to 512MB of RAM. Figure 1-6 shows different adoption speed of two platforms, iOS and Android. One can observe different degree of fragmentation here. At certain time frame, iOS always has main version dominating the version share, while Android does not. While it takes about 1 year for new Android version to become dominant in the market, iOS shows faster adoption speed.

While fragmentation is headache for companies around a platform, it also has benefits. OEMs in developing countries still manufacture low cost smartphones with older versions of Android, since it requires low device performance. The availability of low cost smartphones helps Google reach more customers and reap more Google web access, thus bringing in more advertisement revenue. Meanwhile, consumers have more variety of devices and choose whatever product they want according to their appetite.

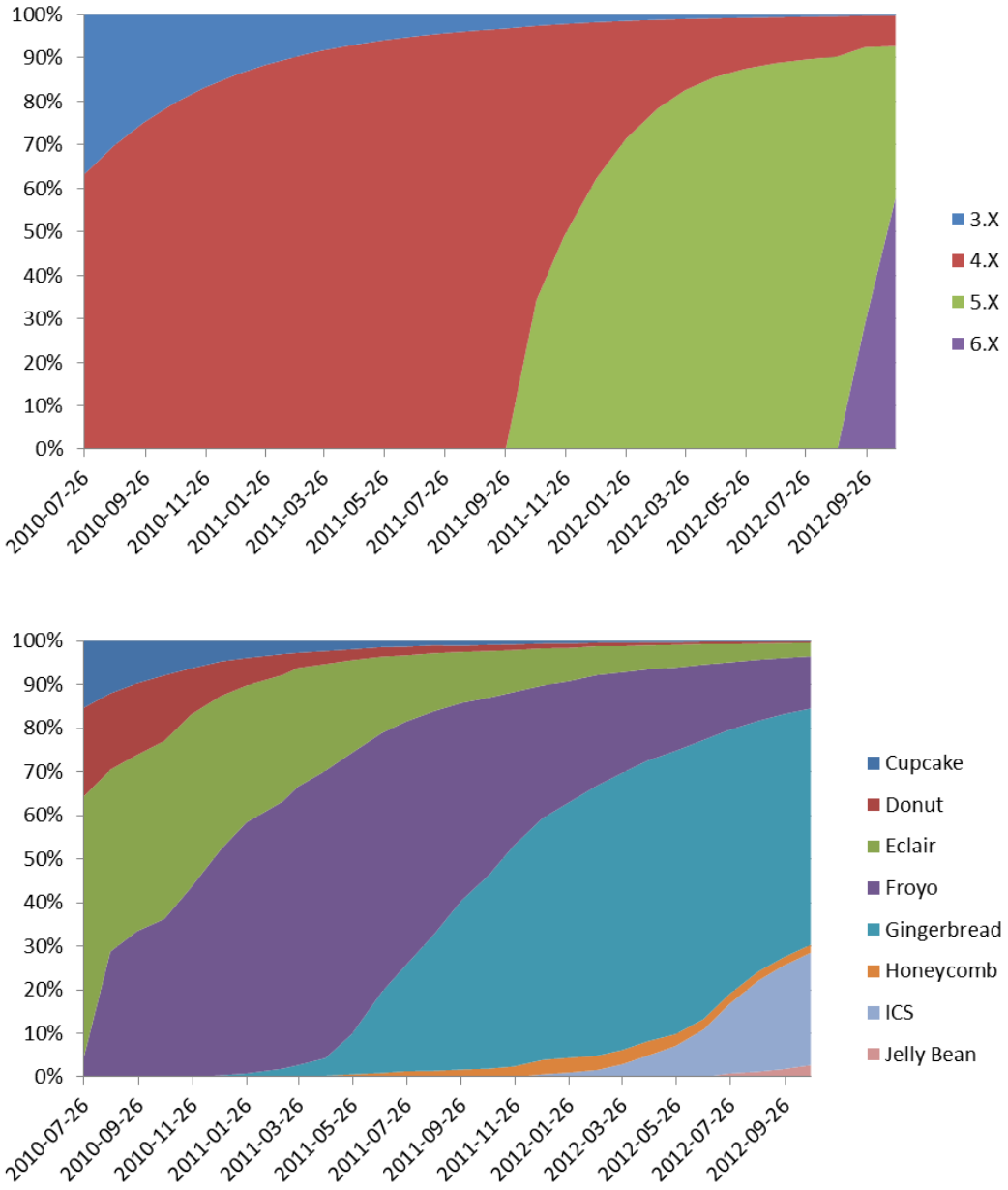


Figure 1-6. iOS(above) and Android(below) version statistics[6]

1.4. Motivation to research

Fragmentation in mobile platform has resulted in several issues. In that adoption speed of new version is

[6] http://opensignal.com

slow, platform participants started to show concerns about version support problems. For mobile platform owners, they cannot deliver seamless mobile experience to end users. When the firmware version is upgraded, new features are added and these new services require compatibility between each devices. Fragmentation of version hampers such a seamless service. To illustrate, Nike once announced that they won't support Android mobile platform for their wearable device, Nike+ FuelBand, because of incompatibility between each versions and devices. In that each firmware version supports different kinds of sensors and APIs for those sensors, to support common denominator function, Nike could not introduce full featured products. To be specific, Fuelband employed Bluetooth Low Energy technology, which requires Bluetooth 4.0 version. However, Android supported Bluetooth 4.0 only from Jellybean, which took less than 30% of the whole installed base. In this way, fragmentation sometimes results in shortage of killer applications or killer complementary products that fully exploit features of the latest version.

There's also another problem, which is called a backwards compatibility. While platform providers try to deliver most updated features that exploit more advanced hardware and software, consumers and application developers are concerned with backwards compatibility. When new product is introduced, these new products frequently do not support old versions of applications. In 2007, when Xbox 360 was launched, it announced that the device would not support older version of applications. However, consumers demanded backwards compatibility for older game software. Therefore, Microsoft started to release compatible versions of games. Observing the Xbox's compatibility issue, PS4, launched in 2013 started to provide streaming service for older versions of games. Xbox One, which was announced that it won't support any of older versions, posted on official website that Microsoft will start such a streaming service for older game titles from 2015.

For OEMs employing open platform, it is troublesome to support upgrade for all products. They have to customize newly released versions according to each different hardware configurations and buyer's requirements. It not only costs expense, but also incurs shortage in manpower that has to be invested in supporting new product development. Therefore, in reality, version upgrade supports for old flagships are normally preceded by new flagship development, delaying support for older products. Also, update schedule for each product varies from 2 month to several months, and sometimes is cancelled, frustrating customers (Figure 1-7).

The mobile platform economy incurs a lot of cost and benefit around the platform participants through cascading release of version upgrades. When new version is released, OEMs need to customize it for both new products and old products. Application providers need to adjust their applications or sometimes have to give up supporting older software version and give up revenue from users of older products. There's no

specific gauge to support decision making whether to support older versions or not, neither in the decision of how many versions or configurations OEMs or application providers need to support. For platform providers, it is important to increase adoption rate to deliver its new services and values to old customers and increase loyalty to their platform.

Once platform participants understand how different diffusions of open and closed platforms are and why fragmentation occurs, they can better foresee market evolution and formulate strategy that reduces wasted cost and effort. Throughout the study, by comparing two dominant platform's diffusion speed and underlying dynamics with the aid of system dynamics, the process of mobile platform diffusion will be analyzed and managerial implications will be discussed. For case study, two successful platforms, Android and iOS, both representing open and closed policies, were selected for modelling extreme ends of the platform policy and compared.

Figure 1-7. iOS and Android fragmentation[7]

1.5. Thesis structure and methodology

Mobile Platform plays an important role in consumer's decisions to adopt and use particular mobile services and applications. However, literatures on mobile platform have been focused mainly on strategic

[7] http://www.fidlee.com/android-support-vs-ios-support/

issues in managing multi sided platforms or economic issues of two sided markets. Using System dynamics modeling, the thesis will emulate software diffusion pattern of both cases (iOS and Android). Using several important dependent variables, such as firmware release interval and manufacturer's software upgrade delay, a system dynamics model will be constructed to investigate how decision variables effect the fragmentation of software versions in both open and closed platform cases. To verify the viability of the model, the study will further include the prediction of the S/W fragmentation of mobile OS versions and sensitivity analysis of important decision variables that have effects on S/W fragmentation.

In this research, diverse studies related with platform economy, network, value network, and open and closed innovation will be reviewed in section 2. In section 3, both historical and technological background of case study platforms, Android and iOS will be described. Case study includes not only the diffusion pattern of platforms, but also the diffusion pattern of complementary products, which was used to model the platform value. In section 4, starting with brief discussion of historical studies done on technology diffusion, the diffusion of mobile platform will be modeled using system dynamics modelling framework. The model will be used to examine the real world platform diffusion, both Android and iOS, to compare how platform policy, openness or closeness, influences the diffusion pattern. Modeling result will also include sensitivity analysis to further analyze the effect of platform policy on diffusion pattern. Finally in section 5 and section 6, conclusion and recommendation for further study will be described.

2. Literature Review

2.1. Platform strategy

Platform is defined as products and services that bring diverse parties together in two or multi sided networks (Eisenmann, Parker, and Alstyne 2006). Platform owners or intermediaries provide infrastructure and guide that foster two or more different groups' transactions. Different from stand-alone products or services, platform requires external ecosystem that generates complementary innovation (Cusumano 2010). By serving different markets around the platform, platform intermediaries normally collect revenue from each side. Example of platform can be found in many industries (Table 2-1).

Product Category	Market 1	Intermediary	Market 2
Operating Systems	Complementary applications	Microsoft, Apple, Sun	Systems developer toolkits
Credit Cards	Consumer credit	Bank	Merchant processing
Broadcast	Content	Magazine, TV, Radio	Advertisements
TV format	UHF, VHF, HDTV	Manufacturers	Broadcast equipment
Reservation systems	Travelers	Expedia, Travelocity	Hotels, Airlines, Rental Cars
Streaming A/V	Content	Microsoft, Apple	Servers
Paid Search	Searchers	Google	Marketers

Table 2-1. Two sided market examples (Parker and Van Alstyne 2005)

This form of industry is different from traditional industries in several aspects. In traditional value chain, value creation is omnidirectional, flowing from suppliers to buyers, but in platform, value is created through any directions. The other characteristic of platform is increasing return. Also, in traditional businesses, growth slows down after reaching certain saturation point. However, in platform, as more participants participate at the platform, return grows further and this again increases platform value, attracting more participants and creating positive feedback loop. Value created grows faster as each participant' demand and number of users increase. Furthermore, once positive feedback loop among

participants and platform intermediary is created around the platform, network effect further fosters growth of the whole ecosystem. Therefore, customer's willingness to pay increases as the network effect grows larger. Another interesting characteristics of platform industry is that first comer advantage is so enormous that winner sometimes takes the entire market (Katz and Shapiro 1994; Sun and Tse 2007). For example, markets such as PC operating system, web browser, and VCR have been under monopoly. Sun and Use (2007) figured out that if participants tend to single-home, it is most likely that one network will dominate the market, which is so called "winner takes all" situation.

On the contrary, these dynamic characteristics of platform place platform owner or intermediary on vulnerable position both from entrants and incumbents. Eisenmann, Parker, and Alstyne (2006) summarized three key challenges for platform providers: (1) proper pricing or subsidization for each user groups, (2) decision to open or maintain proprietary control of the platform, and (3) threat of envelopment from new innovator. While latter two issues have been dealt in platform strategy field and less studied, pricing and economic effect was more elaborately studied (Parker and Van Alstyne 2005). Studies warn platform owner that ignoring cross elasticity between network externalities (parties participating at the platform economy) can lead to pricing and product design errors.

2.2. Network effects

Network effect refers to a phenomenon that the utility, a value that user receives from the consumption goods or services, increases as the number of other users consuming the same goods grows (Katz and Shapiro 1985). In the ecosystem of platforms, products and services have lower value in isolation than in combination (Katz and Shapiro 1994). Like smartphone and applications, products and services in combination generate more value than in isolation. The platform's value is largely dependent upon the number of participants. According to the Metcalfe's law, the value of a network quadruples as the number of users doubles.

Network effect is composed of two different components. First one is the bandwagon effect. It refers to a situation that adoption from early adopters lead to larger adoption of the same products or services (Cusumano, Mylonadis, and Rosenbloom 2014). Second one is the penguin effect. It refers to a situation that adoption from majority groups lags behind the adoption from the early adoptor groups, because of market uncertainty and lack of reference for the utilization. According to the study of Cusumano, Mylonadis, and Rosenbloom (2014), in late 1970s, late mover, VHS, caught up the first mover, BETA and finally dominated the home video market through the successful platform strategy. To catch up with Sony,

Matsushita tried hard to form an alliance to get more participants in their platform and widen distribution channel and mass production. With the boom of home video rental business, the strategy gained momentum and the company succeeded in constructing strong platform. Another one is the penguin effect. Faced with the uncertainty, users postpone purchasing decision until they see clear market potential. RCA delayed contraction on VHS until Matsushita meet technical requirement on longer recording time and expand market share in Europe. The initial hesitation represents the penguin effect and latter huge diffusion represents the bandwagon effect.

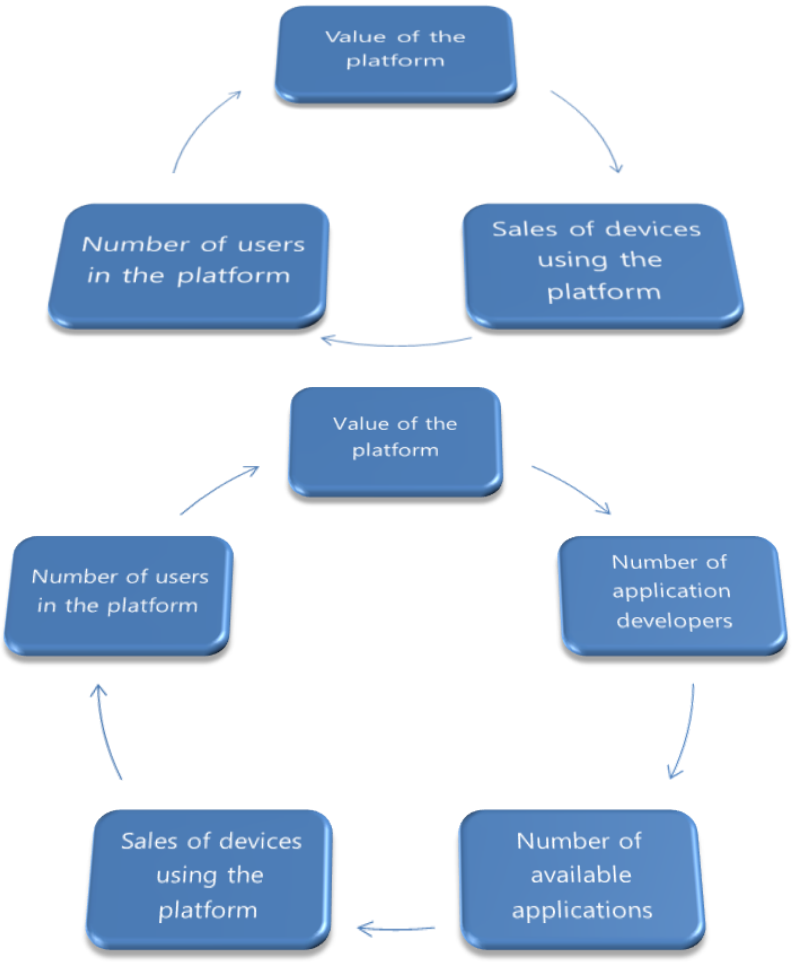


Figure 2-1. Direct (above) and indirect (below) network effects

In multisided platform, that has at least two interfaces between platform intermediaries and network externalities, other kinds of network effects also emerge. One is direct network effect and the other is indirect network effect (Katz and Shapiro 1985). Direct network effect (or same side network effect) is a feedback loop created within the same market. As is the case with telecommunication network, as the

number of customer participating a platform grows, the value of the platform increases and brings in more customers to the platform. Indirect network effect (or cross side network effect) is a feedback loop created across the platform. As is the case with mobile application store, the increased value of platform attracts more application developers. At the same time, more customers are attracted to the abundant applications and this in turn increases the value of platform. Simplified diagrams for direct and indirect network effect are depicted in Figure 2-1.

Network effects also bring unique challenges to platform intermediaries in pursuing their missions. Three important challenges are expectation, coordination, and compatibility (Katz and Shapiro 1994).

(1) expectation: consumers form expectations about availability, price, and quality of the products and services that they are willing to buy in the future. Constructing positive expectation is crucial for future growth of the platform.

(2) coordination: coordination among system components is important, such as development of new processors and development of driver to run the processors. These components form networks that give rise to feedback effects. Timing among these components is crucial to maximize network effect.

(3) compatibility: it is important to determine right degree of compatibility. Decisions such as whether to open software to other developers or not would be an example for compatibility. Normally, dominant players do not open system.

Network effect enables us to estimate the utility or value of the platform or ecosystem. Katz and Shapiro (1985) modeled this network effect as following equation. According to their study, the value of the platform (U(x)) to customers is proportional to basic willingness to pay (r) and size of the network (v(y)), and disproportional to price of the product or service (P). In this study, the equation was further modified to better fit the mobile platform.

$$U(x) = r + v(y) - P$$

2.3. Value Network

As the mobile industry evolves into platform industry, traditional value chain has been transformed to value network, where diverse participants exchange different channels of transaction (Li and Whalley 2002). While traditional telecommunications industry was composed of the equipment, network, and

services, current telecommunications industry is composed of equipment, software, network, applications, middleware, and so on (Basole 2009). The role of platform providers became more important. Study done by Basole (2009) indicates that platform provider has significant relationship with diverse network participants (Figure 2-2).

Mobile platform is closely related with diverse values that platform externalities bring to each other's customers. While traditional mobile industry had value chain starting from suppliers, manufacturers, and network operators to customers, current mobile industry has been transformed to value network structure. With different types of roles and business models, those participants contribute to the value of the whole network. As an example, description for each network role in mobile industry is given in Table 2-2. To illustrate, one can observe weaker relationship between mobile network operators and stronger relationship between platform providers and application providers (bold lines represent more intensive relationship).

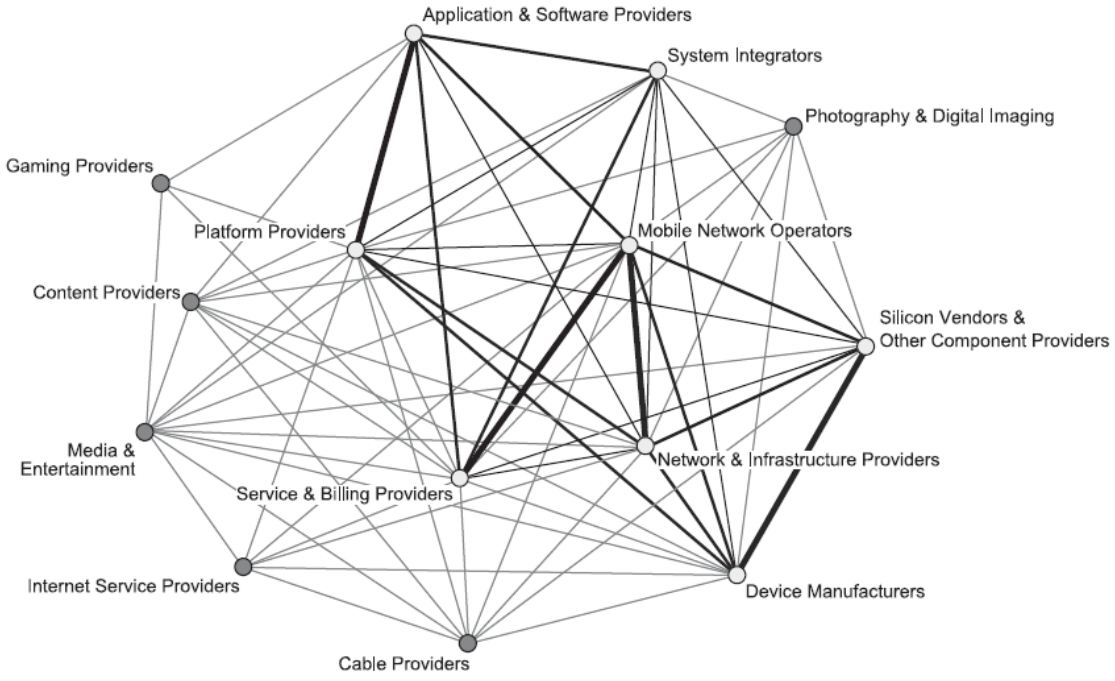


Figure 2-2. Visualization of inter-firm relationship in mobile industry (Basole 2009)

Mobile network operators used to have a dominant power in mobile service market. They dominated mobile network transaction and application developers had to follow restricted rules. However, penetration of expansion of smartphone changed the environment. The change has created new opportunities and challenges for platform providers. New market players have emerged and started to attract consumer's attentions with diverse advantages. Pagani and Fine (2008) constructed detailed value

network relationship, and it is depicted in Figure 2-3.

This rapidly changing market dynamics have pushed platform owners to continuously reshape their strategy and business model. The introduction of TIME (telecommunications, information technology, media, and entertainment) industries (Hacklin, Battistini, and Krogh 2013) has accelerated such a trend. Under this converging circumstance, firms that were not competitors suddenly become rivals. Firms successfully transformed and positioned well in new landscape captured opportunities and avoided severe challenges, while others not.

Value Network Role Name	Description	Example
End User	The owner of a smartphone	-
Network Operator	The supplier of a network connection for smartphones	T-Mobile
Payment Broker	The supplier of information regarding payments	Paypal, Visa
Advertisement Broker	The supplier of advertisements for in applications	AdMob, Microsoft
Marketplace Owner	The owner of the app store	Apple, Google
Operating System Owner	The developer of the OS	Apple, Google
Testing Party	The third party assigned with testing and verification of software	Sogeti HT
Signing Partner	The third party supplying a signing certificate	VeriSign
Software Developer	The software provider	Layar
Content Provider	The content provider for software	Disney, Warner Bros.
Software Distributer	The software aggregator, intermediating in providing software	EA Games
OEMs	The designer and manufacturer of the smartphone hardware	Samsung

Table 2-2. Mobile Platform's Value Network Roles (Müller, Kijl, and Martens 2011)

The uniqueness of mobile platform competition is that it is not purely a competition among software companies with similar business model and strategy. Rather, major platform providers own different business models in complex web of value network. Sometimes the platform is just a part of each company's business. Therefore, the mobile platform competition can be said that it is not a face to face relationship (Lin and Ye 2009). Lin and Ye (2009) analyzed four different mobile platforms and found out commonality that OEMs and application developers play crucial role in mobile platform's value network. According to the study, while OEMs become direct customers of platform owner, application developer is also an important resource to platform value.

Within huge network of value chains, it is important for platform providers to consider the impact of network effects, economies of scale, platform differentiation, quality assurance, transaction costs, and application market construction (Müller, Kijl, and Martens 2011).

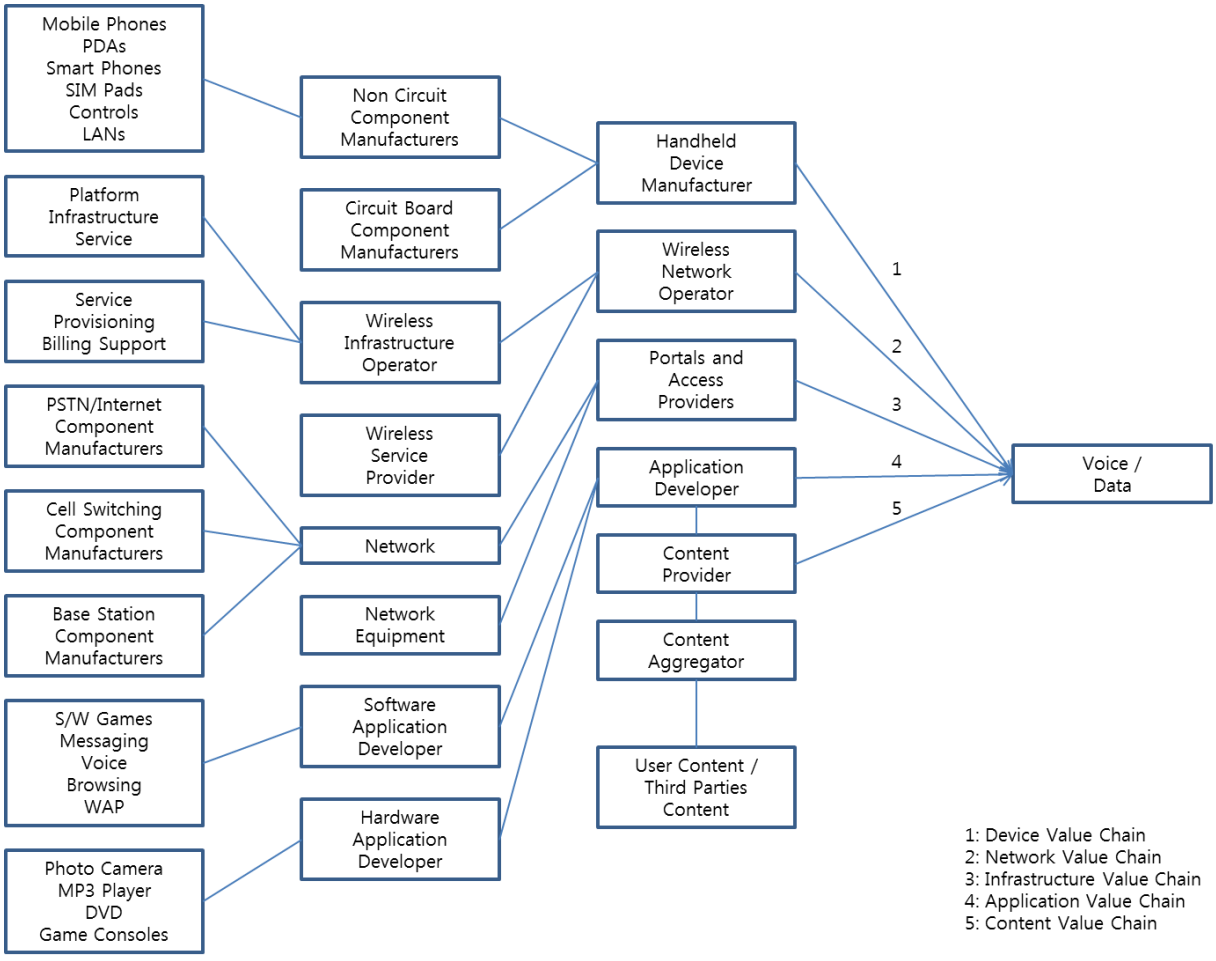


Figure 2-3. Value network of mobile industry (Pagani and Fine 2008)

2.4. Open and closed innovation

While the diffusion of technology, product, or service, is dependent upon factors, such as market situation and firm's strategy, it has been known to be largely dependent upon the types of innovation. There are several studies that explain different types of technology innovation. Baldwin and Hippel (2010) categorized the types according to the origin that initiates an innovation; producer innovation and open collaborative innovation. Producer innovation refers to one that is initiated by a single, non-collaborating firm. These firms normally patent their inventions and license them to other firms. Meanwhile, open collaborative innovation gathers contributors who share the work and open the outputs from their individual or collective works to everyone. In this case, critical prerequisite for participants is that incurred cost should be lower than in-house innovation. The paradigm of open innovation became so pervasive nowadays. Boudreau (2010) investigated 21 handheld computing systems and compared the diffusion speed between open collaborative and non-collaborative platform strategy. According to the research, by giving up control and opening the platform to grant outsiders access to it, platform owner built up wider network with OEMs. The number of new device built on open platform was five times higher than closed producer platform. Moreover, Raymond (1999) denoted that open innovation accelerate innovation in that it brings more eyeballs onto the project.

While the improvement of technology fosters collaborative efforts and make open collaborative innovation pervasive (Baldwin and Hippel 2010), still the decision to open or close is highly related with platform strategy. Decision to open or close depends on the strategic position of a platform: firms with good reputations or large existing networks tend to be against open policy (Katz and Shapiro 1985). In platform economy, open and closed policy seems to play an decisive role, but still other factors, such as network effect, economies of scale, platform differentiation, quality assurance, and transaction cost still play important role in competition of platform (Müller, Kijl, and Martens 2011). The degree of openness also matters. Decision to maintain proprietary ownership, to license it to other OEMs, or to open whole mobile platform architecture has different amount of effect on performance of the platform.

The degree of openness or closeness is hard to express in functional forms, but study done by Jansen, Brinkkemper, and Cusumano (2013) provides a framework to estimate the degree of openness by slicing down mobile operating system into layers. They divided mobile operating system according to software architecture, and evaluate the possibility of licensing or modification of each layer. The aggregate of evaluation result represents the degree of openness or closeness of certain platform. Example is given in Table 2-3. Meanwhile, Rahmandad and Sibdari (2012) incorporated degree of openness in value ranging from 0 to 1 for the estimation of software openness.

	Android	iOS
Integrate extended applications	Partially possible	Partially possible
Extend extended applications	Partially possible	Partially possible
Modify extended applications	Partially possible	Partially possible
Integrate native applications	Possible	Partially possible
Extend native applications	Possible	Impossible
Modify native applications	Possible	Impossible
Integrate middleware	Possible	Possible
Extend middleware	Possible	Possible
Modify middleware	Possible	Impossible
Integrate kernel	Possible	Possible
Extend kernel	Partially possible	Partially possible
Modify kernel	Partially possible	Impossible

Table 2-3. Comparison of openness strategy in the main mobile software platforms (Jansen, Brinkkemper, and Cusumano 2013)

3. Case Study of iOS and Android

3.1. Mobile platform history and architecture

An operating system is the heart of the smartphone, since it determines features, performances, security, and availability of applications (Lin and Ye 2009). It controls device hardware components and provides information processing for applications, as an intermediary between application and hardware components. It also schedules time to efficiently use device resources, such as processor, memory, printing, and so on (Cho and Jeon 2007). While early mobile operating system was real-time operating system, which was designed to execute tasks according to pre-determined (or hard coded) schedule, smartphone platforms are general purpose operating system, which allows more flexible time scheduling and multitasking. Mobile platform consists of kernel, middleware, and applications (Figure 3-1) (Jansen, Brinkkemper, and Cusumano 2013).

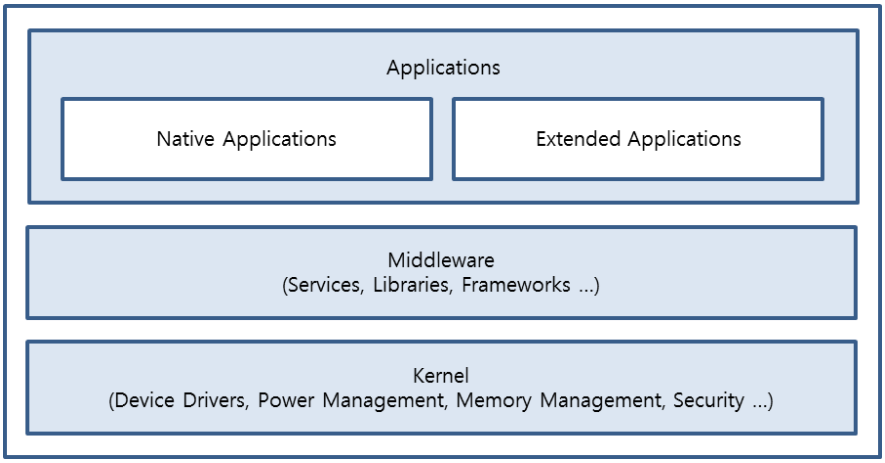


Figure 3-1. Mobile platform architecture (Jansen, Brinkkemper, and Cusumano 2013)

(1) kernel: kernel's main function is managing the device hardware resources and connecting applications to run on these resources. Typical resources are CPU (central processing unit), memory, and input/output components.

(2) middleware: middleware is an abstraction that provides general functionality manipulated by additional code. Therefore, it is general and reusable software to run application. Middleware includes libraries, compilers, and application programming interfaces (API).

(3) application: application connects mobile device with consumers. It forces a device to perform certain tasks. In mobile platform, there are native applications, which are provided by manufacturer or platform

provider, and extended applications, which are installed by users.

Mobile computing started with personal data assistants (PDA) in mid 1990s (Hall and Anderson 2009). Palm introduced mobile computing device that ran simple applications, such as contact, calendar, and note pad. In 2003, Blackberry introduced smartphone that supported push email and web browsing. In mid 2000s, Symbian took the dominant position, by licensing its platform to Ericson, Sony, and Samsung. After the launch of iPhone in 2007 and Android in 2008, those early platforms gave market dominance to latter two giants. In late 2000s, mobile industry became obsessed with the platform battle between Android and iOS. Several new entries like Palm OS and bada failed to enter the market due to lack of understanding of platform dynamics. No matter how hard those new entries tried to share the market, it was hard to avoid the dynamics of platform where fast followers or imitators cannot compete with existing entrants without distinct strategy. Meanwhile, Google was keen on platform strategy and utilized its dynamics well to compete against iOS and become dominant competitor. Brief explanation for historical mobile platform is summarized in Table 3-1.

Operating System	Company	Product Launch	Application Store	Policy
Symbian	Nokia	1999	Ovi	Proprietary → Open
Windows Mobile	Microsoft	2002	Windows Phone Store	Proprietary Licensed
Palm OS	Palm	1998	-	Proprietary Licensed
BlackBerry	RIM	2002	BlackBerry World	Proprietary Not licensed
iOS	Apple	2007	App Store	Proprietary Not licensed
Linux Mobile	LiMo Foundation			Open
Android	OHA, Google	2008	Google Play	Open

Table 3-1. Key mobile platforms (Kenney and Pon 2011)

iOS and Android has distinctive difference in their strategy and policy, such as operating system licensing, application development and registration, and application store profit sharing. Android differentiated itself from iOS well. As a follower, Google utilized platform dynamics strategically. To draw more stake holders around mobile industry onto their side, Android licensed the OS for free in 2008, lowered developer registration fee to $25, and cut the long quality checking process for newly registered applications. In addition, Android utilized value network. It has been known that platform depends on value network, while classical industries depend on value chain, and that value does not flow into one direction, but crosses through various networks of participants. Against Apple App Store's 30:70 revenue share scheme, where Apple took 30% of revenue, Android distributed 30% of Google play store's revenue to network carriers. For network carriers who had lost value added service revenue to App Store, it was good news and they started adopt more Android products on their product portfolio.

Criteria	Apple Appstore	Google Play
Platform (OS)	iOS	Android OS
Platform Source Code License	Proprietary	Open Source
Licensing Costs	99$ Yearly	$20 one-time fee
Revenue share	70% Developers 30% Apple	70% Developers 30% Google/Telco Carriers
Selling Mechanisms	iTunes	Google Checkout Carrier Billing
Refund Policy	Not allowed	Users can return any application before 15 minutes after the purchase
Development Restrictions	Attempted to limit development tools and programming language. Restriction relaxed in 2010	Available open source development tools. Freedom to choose any development tool, programming language or framework.
Development tools	Closed IDE for specific OS (XCode), and additional supporting tools	Open Source reference tools (Eclipse-based). Freedom for different approaches

Approval Process	Internally managed, Restrictive criteria, Pessimistic publication	Crowd sourced, Permissive criteria, Optimistic publication
Typical Approval Time	One week	Almost instantaneous
Partnered Telco Operators	Initially one per country, open to all since July 2010	Every major operator (with some devices sold exclusively)
Main Programming Language	Objective C	Java
Device vendors supporting the platform	Apple	Open Handset Alliance (20 members including Samsung, HTC, and Motorola)
Hardware heterogeneity	Fixed platform with yearly device upgrades	Minimum set of capabilities but large flexibility for device manufacturers
Platform customizability	Disallowed. Applications overlapping existing functionality are rejected	Base platform functions can be replaced by market applications
Application install sources	Exclusively the Appstore	Google Play, alternative markets
Look and feel	Homogeneous	Large diversity in H/W and S/W

Table 3-2. Difference between iOS and Android platform policy (Cuadrado 2012)

3.2. iOS

3.2.1. iOS history

iOS is a mobile operating system that is developed and distributed to run iPad, iPhone, and iPod touch devices[8]. It was first introduced in 2007 at Mac world Conference & Expo, January 2007[9]. Software

[8]https://developer.apple.com/library

development kit (SDK) was released in next year, March 2008. The iOS catalyzed a transition of traditional mobile industry into value network industry (Kenney and Pon 2011). Once mobile internet was only connectable through mobile operators' portals, but iOS collapsed the boundary between the mobile device and mobile internet. Also, iOS created an ecosystem, which was originally under operation for iPod products, where application developers and customers voluntarily transact with each other and bring in more value to platform. Therefore, in platform strategy aspect, Apple successfully implemented tipping strategy (Kenney and Pon 2011). This strategy brought Apple huge first mover advantage in terms of locking in users inside iOS ecosystem. Even after the market share was surpassed by Android, still 54.5% of mobile internet connection is through iOS devices, while Android is 34.6% and Java ME is 4.3%[10].

3.2.2. iOS architecture

iOS architecture is written in Objective-C language and comprised of four layers, Cocoa touch, Media, Core service, and Core OS. System interfaces are provided by framework, which contains libraries and resources such as header and image.

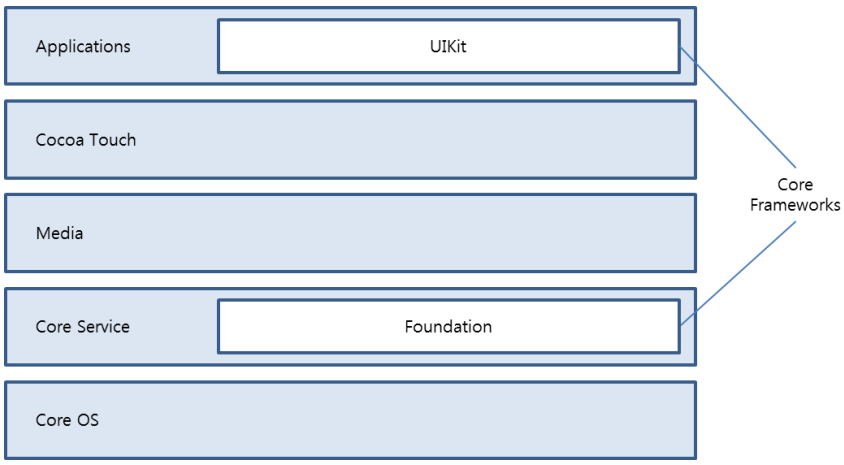


Figure 3-2. iOS architecture

(1) Cocoa touch layer: The Cocoa touch layer is a basic infrastructure to run applications and provide key technologies such as multitasking, touch input, and push notifications. Main frameworks of Cocoa touch layer contain Address Book UI, Event Kit UI (Calendar), Game Kit (Game center), iAd, Map Kit, Message UI, and UIKit. As a main framework, UIKit provides main user interactions such as event

[9] http://en.wikipedia.org/wiki/IOS

[10] http://www.netmarketshare.com

handling, windows, and controls for touch UI.

(2) media layer: The media layer contains technologies to implement multimedia functions, such as graphics, Audio, Video, and Airplay. It contains frameworks for audio/video codecs and OpenGL features.

(3) core services layer: The core services layer provides fundamental system services for applications. Core services layer contains frameworks such as location, file sharing, in app purchase, account management, telephony. Core foundation framework and foundation framework, which provide basic data management and service features, are contained in core services layer.

(4) core OS layer: The core OS layer sits directly on top of the device hardware and provides low level features mostly used by upper layer frameworks. To connect the iOS device with an external accessory or to deal with security features, core OS layer framework need to be used. The core OS layer contains TCP/IP networking protocol, Bluetooth framework, 64 bit support mode, and kernel. Known as Darwin[11], the kernel is opened for open source development, while other layers are proprietary.

3.2.3. iOS application development

To develop applications for iOS, developers need integrated development environment (IDE) called Xcode, which runs on Mac OS X platform, and software development kit (SKD), that provides APIs in the form of libraries. iOS applications are written in Objective-C as well. iOS SDK was first released in February 2008. To publish and upload applications on the devices, developers need to pay $99.00 per year.

3.3. Android

3.3.1. Android history

Android is open-source software for a wide range of mobile devices and a corresponding open-source project led by Google[12]. It was first announced in 2007 with the introduction of the Open Handset Alliance, a consortium of hardware, software, and operator companies. The first product, HTC Dream was launched in October 2008. Android started from entirely different business model (Kenney and Pon 2011), originating from dominant position in PC based web search service. With its free and open policy,

[11] http://www.puredarwin.org/

[12] http://source.android.com/index.html

Android captured attention of OEMs who were weak at smartphone operating system technology.

3.3.2. Android architecture

Android architecture is built on a modified Linux kernel (Butler 2011). System components are written in Java, C, C++, and XML. Applications are run on JAVA virtual machine. Android consists of number of layers as Linux Kernel, Android Runtime, Libraries, Application framework, and Applications. (Gandhewar and Sheikh 2010).

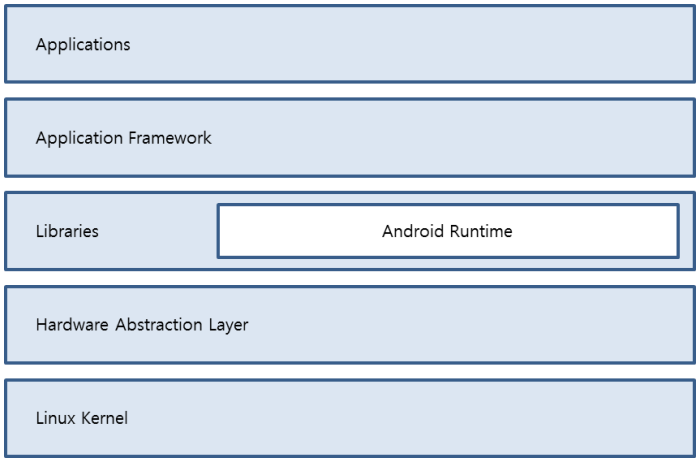


Figure 3-3. Android architecture[13]

(1) Linux kernel: The Android operating system is built on top of Linux Kernel with some architectural change done by Google. Linux kernel controls functions such as security, memory management, process management, and network protocol.

(2) libraries: Android's libraries are written in C and C++ languages. These libraries are called through Java interface. Libraries include graphics, media codecs, database, and web browser.

(3) Android runtime: Android runtime layer is composed of Dalvik virtual machine and core JAVA libraries. Dalvik VM executes files in the Dalvik Executable (.dex) format, which is optimized for minimal memory usage.

(4) application framework: Framework refers to a software framework designed to implement standard structure of programs. Application framework is a set of application programming interfaces to develop applications for Android. It includes tools and application programming interfaces for UI design, media,

[13] http://en.wikipedia.org/wiki/Android_(operating_system)

phone control, and data base management.

(5) application layer: Application layer provides set of core applications such as email, SMS, calendar, maps, contacts, and so on. These applications are written in JAVA programming language.

3.3.3. iOS application development

Android applications are mostly developed on Eclipse IDE. In that android applications are developed with JAVA, Java development kit (JDK) is required, along with Android SDK. To facilitate the Android application development on Eclipse, Android development tools (ADT) is required for streamlined export of *.apk file. Developers pay $25.00 for application registration and distribution on the Google Play application store.

3.4. Diffusion of iOS and Android

3.4.1. iOS and Android installed base

The sales amount of devices based on both platforms has grown up in tremendous speed. After the first product iPhone 2G was launched in Jun 2007, iPhone sales maintained steady growth. Since then, six generations of products were launched, iPhone 3G in July 2008, iPhone 3GS in June 2009, iPhone 4 in June 2010, iPhone 4S in Oct 2011, iPhone 5 in Sep 2012, and iPhone 5C and 5S in Sep 2013[14]. According to Gartner's quarterly industry report[15], quarterly sales of iPhone would reach 50 million on Q1 2014. Meanwhile, the number of Android device sales increased faster than that of iPhone, exploiting the strength of open platform policy. Within two years of first product launch, HTC Dream, the platform surpassed every other smartphone sales, reaching quarterly sales of 10 million units, and now takes more than 80% of the whole smartphone sales.

Considering the fact that Android pursues open policy, it is not surprising to see that the diffusion speed of Android is much faster than that of iPhone. OEMs started to manufacturer more android products, average sales price went down, and the number of complementary products grew faster. Figure 3-4 is quarterly sales of each device after the initial launch of Android in Oct 2008 and iOS in June 2007. To compare diffusion speed accurately, both operating system's sales data were brought together to start at

[14] http://en.wikipedia.org/wiki/Iphone

[15] http://www.gartner.com/newsroom/archive/

Q0, launching time. Android shows steeper exponential growth than iPhone. One can observe fast diffusion speed of Android OS, reaching 50 million units per quarter sales at 11Q after the initial launching. The diffusion speed of iOS is slower than Android, and one can see that the diffusion depends on new product launching schedule. Two peaks in 19Q and 23Q coincide with the launch of iPhone 4S and iPhone 5.

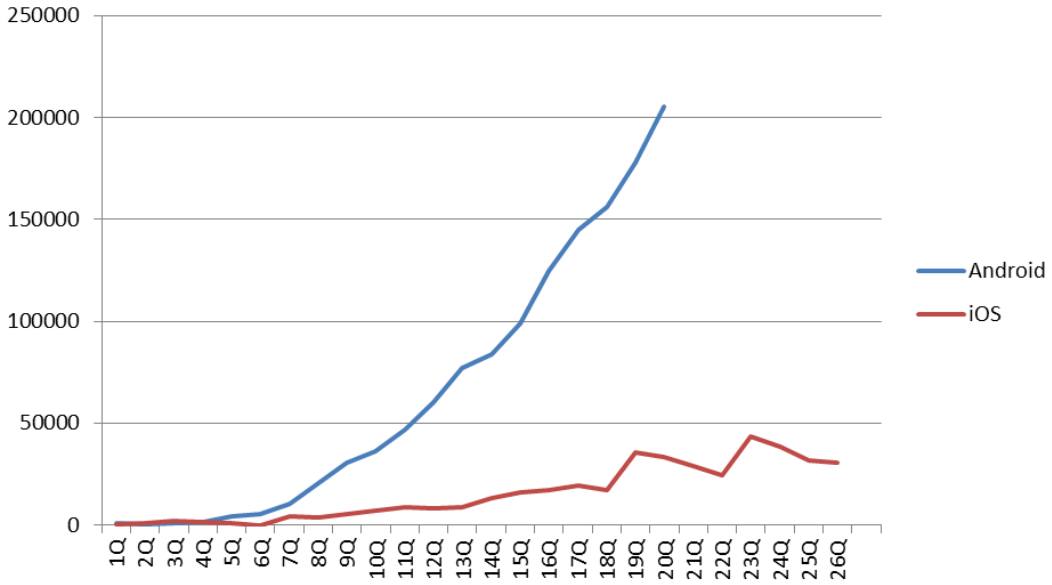


Figure 3-4. Diffusion of iOS and Android[16] (Unit: thousands)

3.4.2. iOS and Android complementary products

The most powerful complementary product or service for mobile platform would be applications and application stores, Appstore and Google Play. AppStore was first launched in July 2008. The transition from iTunes music service to application store resulted in a huge success. Also, this displaced telecommunication operators from central role in disseminating mobile applications to platform participants. Following the Apple's success, other companies started to create mobile application stores, sometimes operated by OEMs (Samsung app store) and sometimes operated by network operators (Vodafone 360). However, only Google succeeded in building up comparable application store. As was the case with VHS and Betamax (Cusumano, Mylonadis, and Rosenbloom 2014), Google gathered

[16] http://www.gartner.com/newsroom/archive/

supporting group via Open Handset Alliance. Over 20 mobile OEMs promised to participate in the value network. Android Google play started operation in October 2008. With the open policy and relatively easy application registration process, Android application started to outnumber iOS applications by mid-2013 (Figure 3-5).

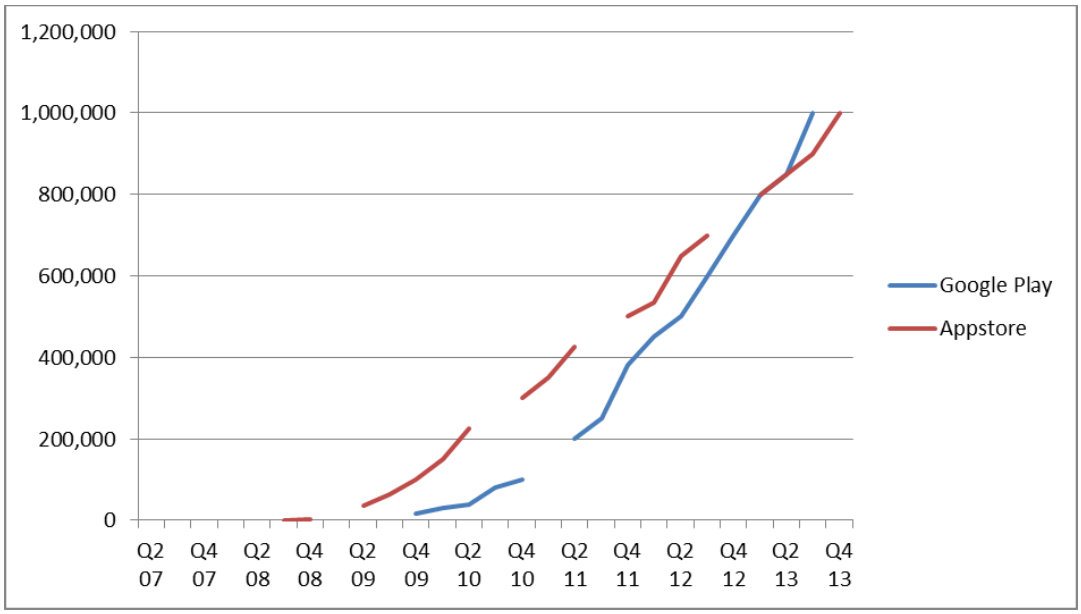


Figure 3-5. iOS and Android number of applications on application stores[17]

Both application stores charge registration fee to developers to maintain minimum application quality. There're differences in the process of application publication. Therefore, while Appstore takes a week for approval, Google play takes only few hours for registration (Figure 3-6).

[17] http://www.statista.com/search/?q=application+store

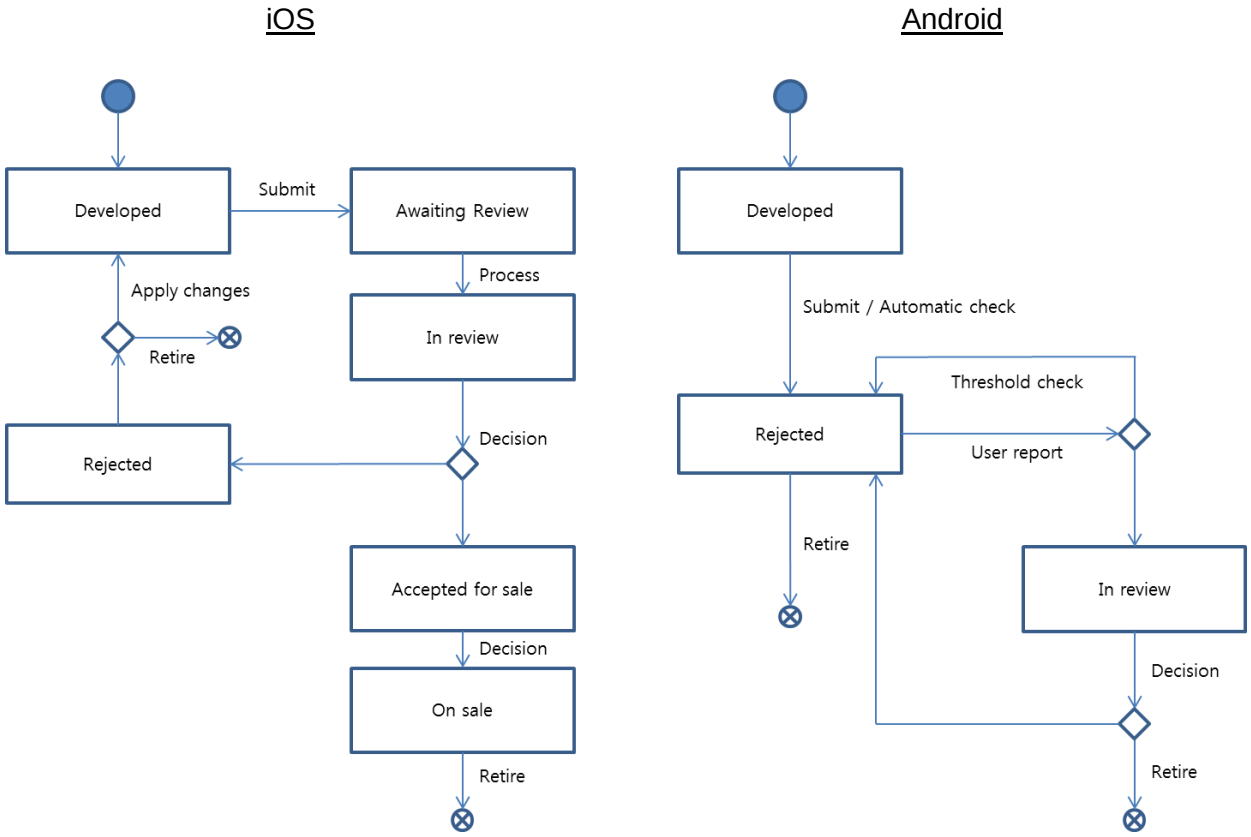


Figure 3-6. iOS (left) and Android (right) application registration process (Cuadrado 2012)

4. Systems Model of Mobile Platform Diffusion

4.1. Introduction to system dynamics

System Dynamics is a field of study that is based on the philosophy of systems thinking. In system dynamics, researchers see the world as complex system where everything is connected to everything else. This connection is represented by feedback process, either positive (or self-reinforcing) or negative (or self-correcting) loop (Sterman 2001). The behavior of such systems consists of those feedback loops, stocks and flows, and nonlinearities. The feedback structure of a system is first represented by casual loop diagrams such as an example in Figure 4-1. Causal loop diagram captures the basic hypothesis about the causes and effect of dynamics and elicits mental models of researchers. This is the initial step to communicate an important feedback responsible for the issue. In causal loop diagram, variables are interconnected by causal links (shown as arrows) with polarity denoting whether the loop is positive (increasing, or reinforcing) or negative (decreasing, or balancing). Figure 4-1 captures how population is increased or decreased according to birth rate, which is dependent upon fractional birth rate, and death rate, which is dependent upon average lifetime.

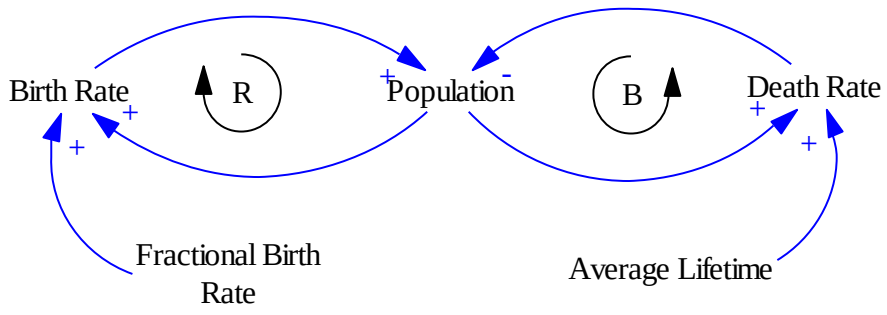


Figure 4-1. An example of causal loop diagram (Sterman 2001)

Finalized causal loop diagram is then modeled into stocks and flows, along with feedbacks. Stocks are represented by rectangles, and inflow and outflow are represented by piped arrow. The concept of stocks and flows can be understood using hydraulic metaphor. Stocks and flows can be compared to reservoirs and the flow of water into and out of it. The quantity of any stock is determined by the accumulation of the flows of material into the stock less the flow of material out of it (Figure 4-2). Therefore, while stock is represented by a certain unit, flows are represented by units per time period. The hydraulic metaphor and stock and flow diagram can be expressed in following equations, either integral equation or differential equation.

$$\text{Stock}(t) = \int_{t_0}^{t} [\text{Inflow}(s) - \text{Outflow}(s) - ds + \text{Stock}(t_0)$$

$$\frac{d(\text{Stock})}{dt} = \text{Net change in Stock} = \text{Inflow}(t) - \text{Outflow}(t)$$

When there are auxiliary variables that have effect on stocks and flows, it is also implemented into the model. In system dynamics modeling process, it is helpful to follow disciplined process (Sterman 2001): (1) articulate the problem to be addressed, (2) formulate dynamic hypothesis or theory about the causes of the problem, (3) formulate a simulation model to test the dynamic hypothesis, (4) test the model, (5) design and evaluate policies for improvement (Sterman 2001).

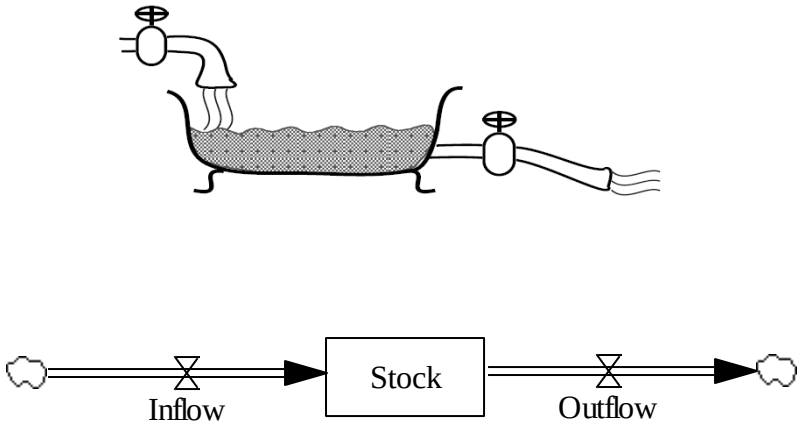


Figure 4-2. Hydraulic Metaphor and Stock and Flow diagram (Sterman 2001)

4.2. Systems model of diffusion

The foremost model of diffusion of new technology would be the Bass Model (Bass 2004). According to the model, adopters can be divided into two groups, innovators and imitators. While innovators adopt new technology regardless of the adoptions of others, imitators are influenced by the contact with social groups. Rogers (2010) further divides these consumer groups into 5 groups according to socioeconomic characteristics; (1) innovators, (2) early adopters, (3) early majority, (4) late majority, and (5) laggards. Bass (2004) aggregated latter four groups into imitators and statistically proved that there are similar adoption pattern among various technologies. The adoption pattern can be expressed by mathematical equations as below.

$$S(T) = pm + (q - p)Y(T) - q/mY^2(T)$$

$$S(T) = p\big(m - Y(T)\big) + q/mY(T)(m - Y(T))$$

Here, T is time, S(T) is the number of adopters, p is the probability that innovators adopt the technology, m is the initial purchases of the product, q is the probability that imitators adopt the technology, Y(T) is the number of previous adopters. Therefore, it can be interpreted as follows.

Total demand = Innovative demand + Imitative demand

Innovative demand = p × market potential

Imitative demand = q/m × adopters × market potential

This non-linear differential expression of diffusion has been well implemented in system dynamics (Figure 4-3). Here, Alpha is equal to p and Beta is equal to q in equations above.

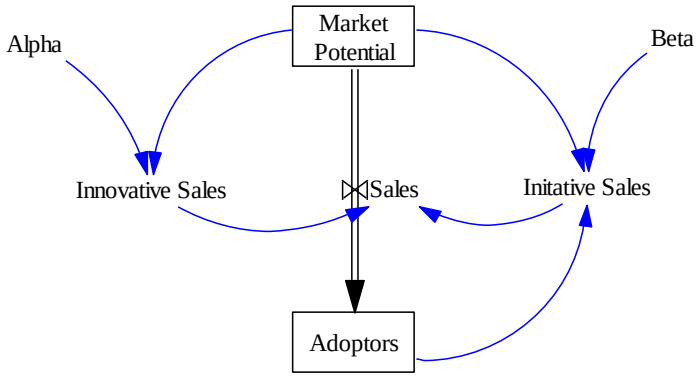


Figure 4-3. Systems model of Bass diffusion model

4.3. Expansion of diffusion model

The innovation diffusion model was further expanded to include more decision variables, such as pricing, advertising, and product capabilities (Maier 1998). There have been series of researches to combine managerial decision variables to explain the pattern of technology diffusion using system dynamics. By incorporating diverse decision variables, the diffusion model became a useful tool to find causality between dependent variables and diffusion of innovation. Maier (1995) categorized these decision variables into four groups as follows and in Figure 4-4.

(1) market structure: monopolistic, oligopolistic, or competitive market structure

(2) factors directly influenced by managerial decision: pricing, advertising strategy, quality, technology, delay, capacity investment

(3) general innovation diffusion process: word of mouth, substitution of successive product generation, repeated purchase

(4) process off innovation diffusion itself: carry over effects from earlier periods

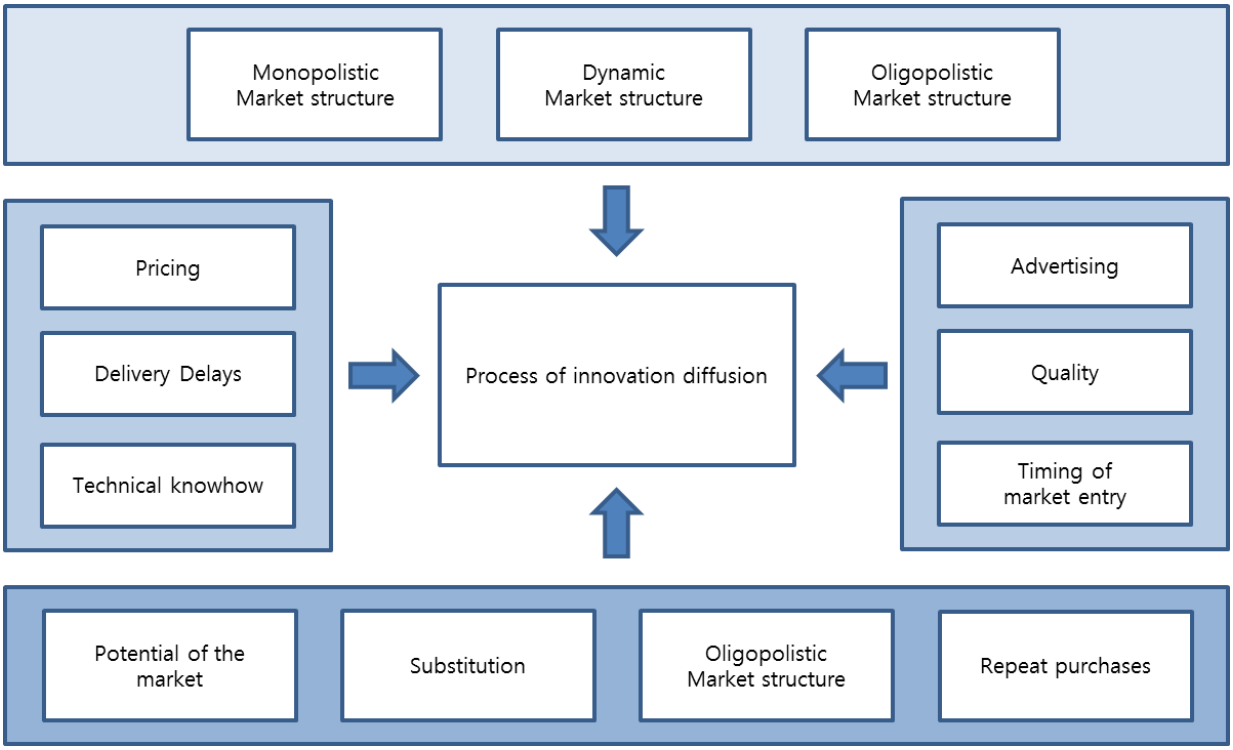


Figure 4-4. Decision variables around the process of innovation diffusion (Maier 1995)

These variables can be expressed in functional form as below equation.

Sales = f(pricing, advertising, quality, delivery delays, productions,
inventory, technical capability, adopters, market potential, …)

When converted to causal loop diagram, these variables can be easily implemented in system dynamics model. Exemplary diagram is depicted in Figure 4-5.

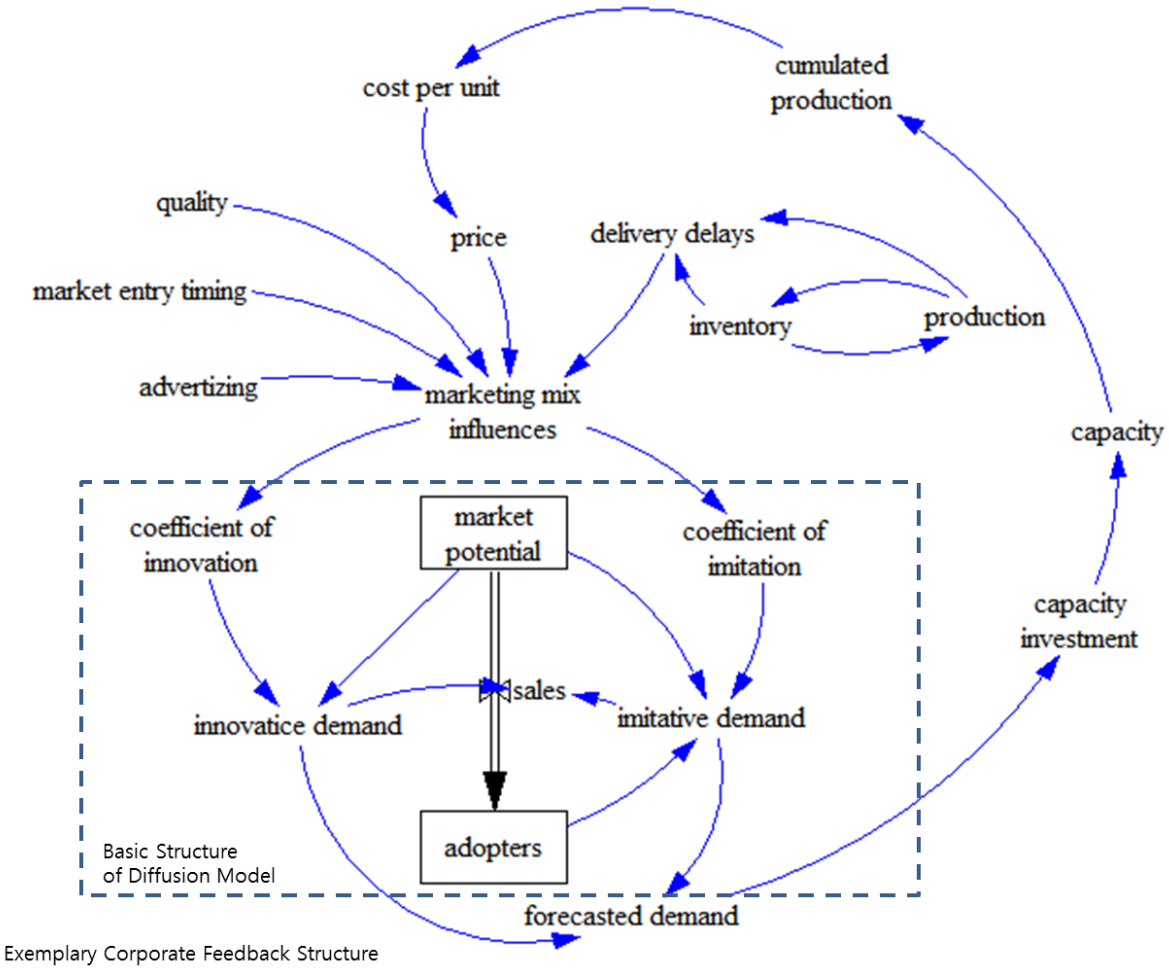


Figure 4-5. Causal loop diagram of innovation diffusion model for managerial decision support (Maier 1995)

The diffusion model was also expanded to explain successive diffusion of innovation. While Bass Model only accounts for the diffusion of a single technology, the model proposed by Norton and Bass (1987) successfully demonstrated a capability to emulate the successive generations of technology diffusion. The basic concept of the model was as follows: with the continuing technology innovation, technologies are replaced by newer ones and this generates successive generation of diffusion. Norton and Bass (1987) modeled this kind of replacement through system dynamics model. Although they successfully fitted the model to real data, the model had limitation in that they had to make an arbitrary guess on maximum number of customers for each generation of technologies.

4.4. Platform diffusion model

Consumers make adoption decision according to the value or utility of a platform. As the value of the platform increases, the probability of adoption also increases. There are several determinants of the utility of the platform. According to study done by Haile and Altmann (2013), usability, service variety, and the number of personal connection were most important ones. Haile and Altmann (2013) indicate that consumers of mobile platform feel most value from the number of participants and number of services. The Study of Zhu and Iansiti (2007) indicates that indirect network effects and consumer's expectation are key drivers for the success of a platform. Since larger installed base of consumer attracts more application suppliers, indirect network effect helps platform stand between both sides of market, consumers and application developers, and prosper even when a quality of the network is inferior to competitors.

Thun, Größler, and Milling (2000) further extended the diffusion model to include network effect. When the utility of a product depends on the number of customers, i.e. platform, network effects exist. By incorporating utility that arises from bandwagon effect and penguin effect, they constructed a technology diffusion model that includes platform utility function. They derived utility function based on number of connection between users (equation below; B represents the number of customers and r represents the number of connection). By incorporating the utility function in the diffusion model, they demonstrated the network effects, where adoption speed is slower at first stage (penguin effect) and becomes faster at later stage (bandwagon effect).

$$U = \sum_{k=2}^{n} \frac{B!}{r!\,(B-r)!}$$

Study done by Casey and Töyli (2012) incorporated diverse value network components in the diffusion of mobile voice services. They used mobile operator's licensing policy as an exogenous variable and found out that network operator's policy has an effect on mobile service's diffusion speed.

Rahmandad and Sibdari (2012) modeled the effect openness and pricing decision on strength of network effect and technology diffusion of software market. By incorporating utility function given by feature richness, price, market share, and openness decision in the dynamics of installed base, they found out that degree of openness has positive effect on the diffusion rate. Also, the study suggested late mover's strategy in platform industry; start-up firms have to lower price and open platform to gain the early advantage. In this study, utility function of Rahmandad and Sibdari (2012) was used to estimate the value

of the platform. While the utility function contains feature richness, model proposed in this study does not include it, since the number of features is nearly same across competing mobile operating systems and not considered critical in the adoption of mobile platform.

4.5. Modeling

4.4.1. Conceptual structure of proposed model.

Conceptual structure of diffusion modeling is given in Figure 4-6. As depicted in the diagram, platform owner releases new version of operating systems, such as Android 2.0, 3.0, and 4.0. Among people of smartphone users, there exist customers who purchase or download released versions. As the number of platform adopters increases, this increases platform value and in turn increases adopters of newer versions. In addition to the number of platform adopters, complementary products features, and average selling price has effect on the platform value.

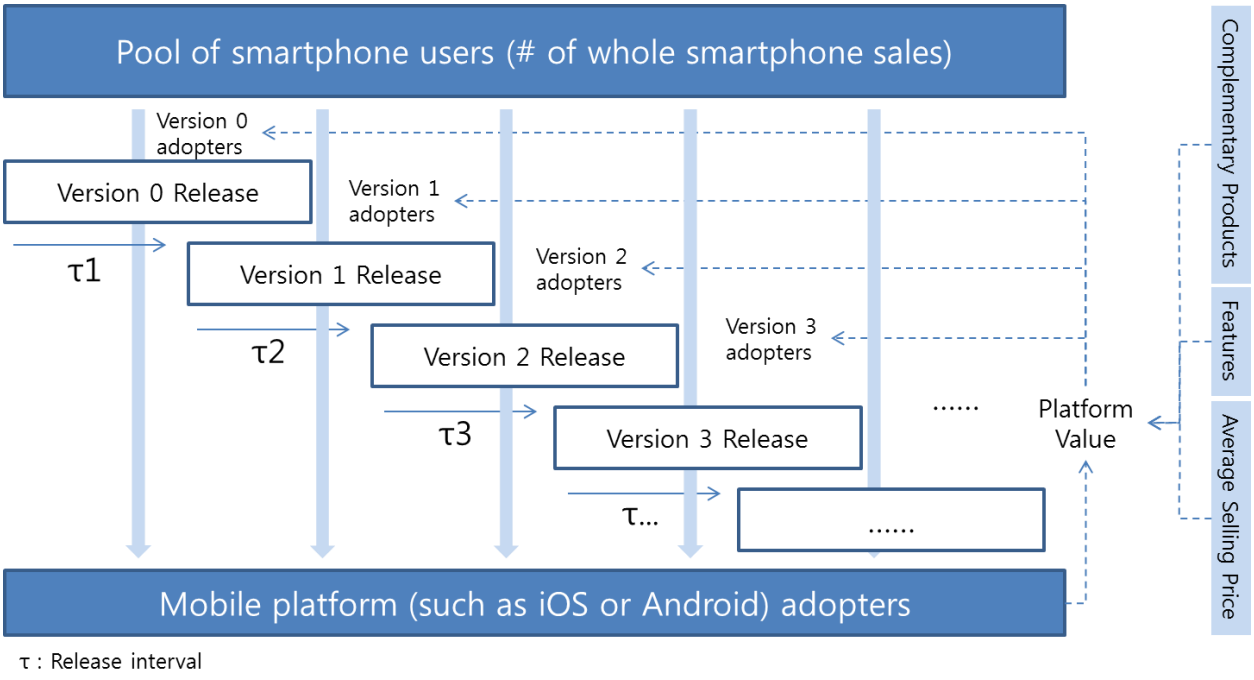


Figure 4-6. Conceptual structure of diffusion model

Descriptions for each component are as follows.

(1) pool of smartphone users (# of whole smartphone sales): this represents smartphone demand. In the

modeling, monthly sales of smartphone in the whole world[18] were used as simulation data.

(2) version 0 ~ 3 release: this represents availability duration for each version. In Figure 4-6, only version 0~3 were drawn, but it can be extended according to the number of releases.

(3) τ: this represents time interval between each version release. It is variable, not constant.

(4) version 0 ~ 3 adopters: this represents the number of adopters for each version. Adopters for each version also can be extended as far as new version is released.

(5) mobile platform (such as iOS or Android) adopters: this represents whole aggregation of adopters of each version.

(6) complementary products, features, average selling price: along with mobile platform adopters, these variables have effects on the platform value.

(7) platform value: this represents the aggregate utility that attracts smartphone users to certain platform. The functional form of utility function for platform value is as follows; S: relative portion of the number of mobile platform adopters among the whole smartphone users, C: the number of complementary products, and P: average selling price. α is weight for each variables and β is sensitivity. We set the summation of weights be 1 and maintained the value of S, C, and P between 0 and 1 to normalize the whole utility function between 0 and 1.

$$U = \alpha_s S^{\beta s} + \alpha_c C^{\beta c} - \alpha_p P^{\beta p}$$

4.5.2. Specific modeling

The proposed model basically starts from defining the diffusion of overall smartphone users. This was done by utilizing bass diffusion model. There are three stocks, Population, Potential End Users, and Active End Users, and other variables that connects these stocks (Figure 4-7). Description for each stocks, flows, and variable are as follows.

(1) Potential End Users: this stock represents the number of potential end users who did not buy

[18] http://www.statista.com

smartphones yet. This stock represents the number of demands in the market. The smartphone market size has been increased with the population increase or finding of new market.

(2) Population: the model employed population from global census data[19]. This stock increases the pool of Potential End Users.

(3) Active End Users: this stock represents the number of smartphone users who bought smartphones.

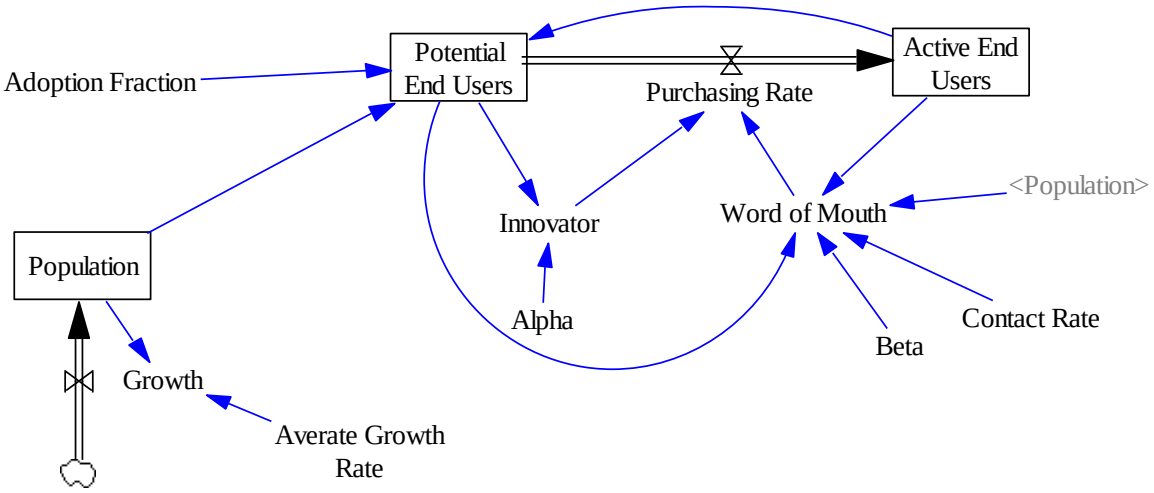


Figure 4-7. Diffusion of overall smartphone users

This basic diffusion model (overall smartphone users) was put into Vensim optimization function to get parameters that best represents the real data. Following values are optimization result, and Figure 4-8 represents the fitted model (blue line).

Alpha: 0.00086

Beta: 0.0061

Adoption Fraction: 0.99

Contact Rate: 7 People/Month

[19] http://www.census.gov

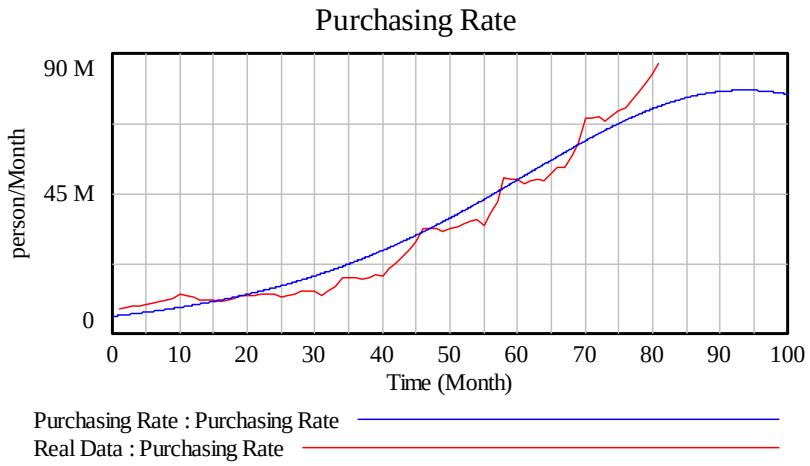


Figure 4-8. Optimization of overall smartphone users model

As a next step, adoption model for installed base of each version was constructed. The main variable here is utility function. Among the pool of whole smartphone sales, only certain fraction of customers adopts particular operating system (or platform). The fraction of such adoption is decided by the utility (or value) of platform. The utility of platform influences the amount of sales of mobile devices using the platform's operating system. The accumulation of such sales of certain platform is called installed base.

The model for adoption of each version was constructed as depicted in Figure 4-9. The difference between initial version (top) and later versions (bottom) is the variable called S/W Update Rate. It represents the number of update from prior version of installed base. Since initial sales (or adoption of initial version) depend only on the sales of new device, adoption model for initial version does not include S/W Update Rate. From second version and later, adoption model contains the variable S/W Update Rate.

Meanwhile, after a certain period of usage (product life cycle), consumers stop using mobile devices. As customers churn away from old devices, the amount of install base starts to decline and this flow is called as Attrition Rate in the model. There's another origin of decline of installed base; S/W upgrade rate. After new version is released, certain proportion of customers upgrades the software into new version, and this causes installed base of older version decrease.

At last, the open and close policy influences the whole diffusion model. To incorporate the policy in the model, the framework proposed by Jansen, Brinkkemper, and Cusumano (2013) was employed. The openness of the platform was evaluated based on the framework and transformed into value ranging from 0 to 1. 0 means perfect openness and 1 means perfect closeness. This numerically converted value was put into the model according to following causalities.

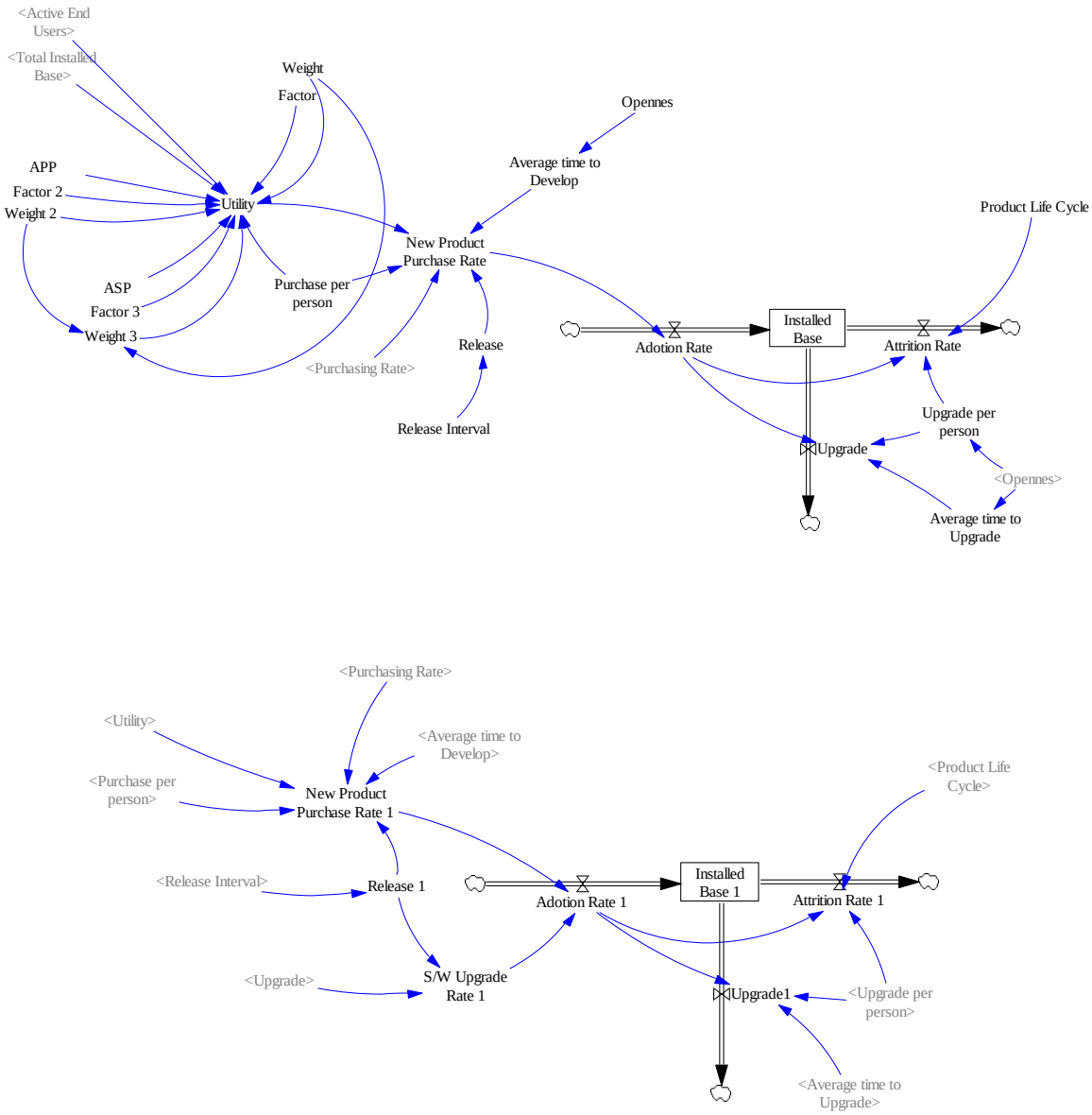


Figure 4-9. Adoption model for each version (top: initial version, bottom: later versions)

(1) the open policy attracts more OEMs and accelerate the diffusion of device sales (not modelled)

(2) the open policy enables OEMs manufacture diverse products and drives average sales price down (not modelled)

(3) the open policy attract more complementary products around the platform (not modelled)

(4) the open policy retards the OEM's development time, since firms have to develop own software.

(5) the open policy retards the customer's upgrade into new firmware, since it takes time for OEMs to support upgrade service for old products

(6) the open policy lowers the proportion of upgrade, since majority of device's software are not upgraded on time.

According to above causalities, the variable Openness was put into the model. Since the model used real data for the number of complementary products and average sales price, causalities (2) and (3) were not modelled in the modelling process, and real data was used in place of them (The study assumed the effect of openness is already reflected in the real data). Causality (1) was also not explicitly modelled, since it was hard to capture every OEM's number of products into the model. Even without explicitly including causality (1), proposed model emulated the real pattern of diffusion relatively well.

The detailed descriptions for main variable are as follows.

(1) Active End Users: this represents the number of smartphone users.

(2) Total Installed Base: the accumulation of whole installed bases (versions). By dividing the total installed base by the active end users, one can get relative percentage of users of specific platform. This relative percentage was put into utility function and represents the relative portion of the number of mobile platform adopters among the whole smartphone users.

(3) Weight: the weight α given to the relative portion of the number of mobile platform adopters among the whole smartphone users.

(4) Factor: the sensitivity β given to the relative portion of the number of mobile platform adopters among the whole smartphone users.

(5) App: the number of applications (or complementary products) registered on the application store.

(6) Weight2: the weight α given to the number of complementary products.

(7) Factor2: the sensitivity β given to the number of complementary products.

(8) ASP: average sales price of devices using particular mobile platform.

(9) Weight3: the weight α given to average sales price.

(10) Factor3: the sensitivity β given to average sales price.

(11) Release Interval: this represents time interval τ between each version release

(12) Average time to develop: this represents average time delay that OEMs introduce new product based on newly released operating system.

(13) Upgrade per person: the percentage of customers who upgrade operating system.

(14) Average time to Upgrade: the average time delay for customers to upgrade operating system.

(15) S/W Upgrade Rate: the rate at which customers upgrade operating system.

After model was set up, predictability test was conducted to verify the viability of the model. As a next step, sensitivity test of the effect of decision variables on software fragmentation was conducted. The results of prediction tests are described in section 4.6.1 and 4.6.2. The sensitivity test result is described in section 4.6.3. To optimize the model into real data, monthly publicized installed base report of both Android developers' site and iOS was gathered from Gartner report. According to Gartner, the data was gathered by counting the number of device logging onto application store. While the data does not include exact number of installed base, it contains share of each versions of firmware. Therefore, our model was modified to calculate relative share of each installed base, i.e. each version, as an output.

4.6. Modeling result

4.6.1. Modeling result for iOS

To see how well proposed model reflects the dynamics of the real world, the model was trained with older installed base data and used to predict the future diffusion of installed base. And then predicted data was compared with real world data to verify the validity of the proposed model. First case study was done on iOS. For training data, the shares of installed base of iOS version 3.x and 4.x were used (data from Jul 2010 to Sep 2012). After being trained, the model predicted the shares of installed base of iOS version 5.x and 6.x (data from Sep 2011 to Sep 2012). Since publically available data for installed base was only share, not exact number of each installed base, the proposed model was modified to predict the share of

each share of installed base version.

Figure 4-10 is the result of training. One can see that simulation data (blue line) fits well with real world data (red line).

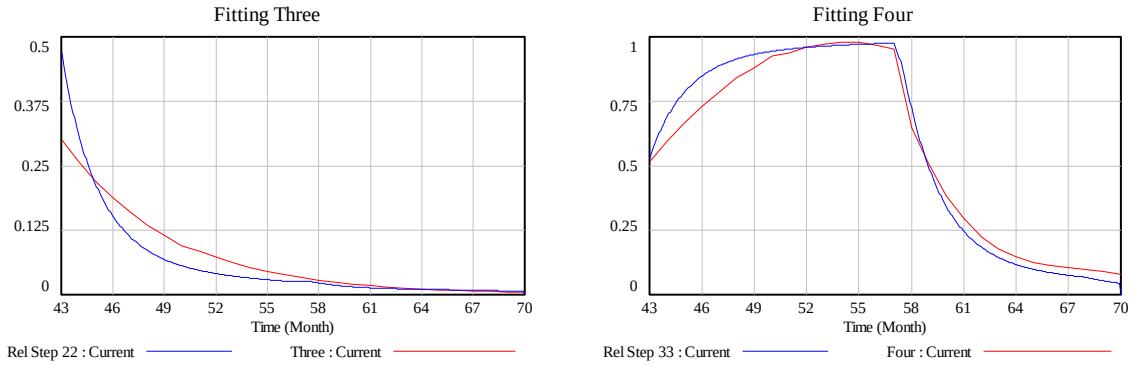


Figure 4-10. Model training (left: iOS 3.x, right: iOS 4.x)

Trained model was utility function with parameters of weights and sensitivities as follows.

$$U = 0.5\,S^{0.1} + 0.17\,C^{3} - 0.33\,P^{3}$$

The degree of openness was 0.99, which means that the platform was highly closed platform.

Using the utility function and degree of openness above, prediction for share of iOS 5.x and 6.x was conducted. Figure 4-11 shows the result, and one can see that model relatively well predicts the real world data (simulation data: blue line, real world data: red line).

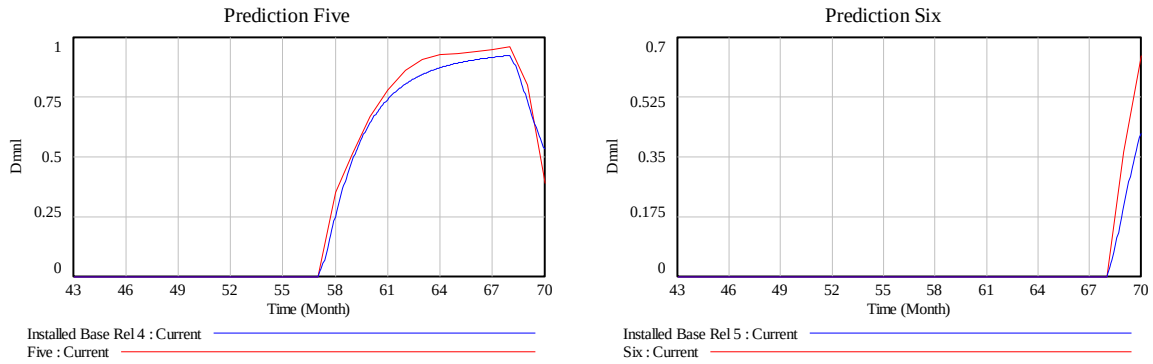


Figure 4-11. Diffusion prediction (left: iOS 5.x, right: iOS 6.x)

Table 4-1 below is comparison between simulation chart and real world data for share of whole versions (3.x through 6.x). Although the simulation does not match exactly with the real world data, it reasonably emulates the real world.

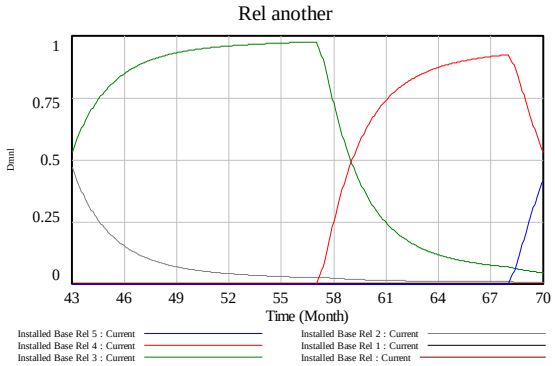
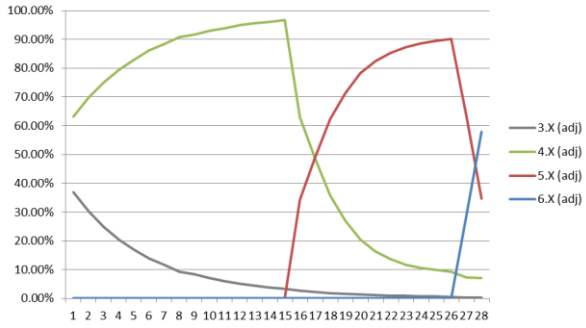
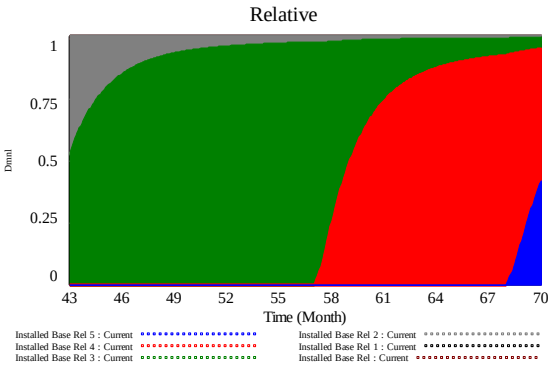
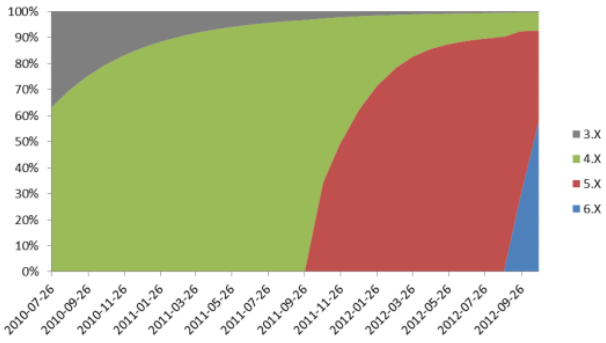
Share (Relative Percentage)	
Simulated Data	**Real World Data**
	
Cumulative Share (Cumulative relative percentage)	
Simulated Data	**Real World Data**
	

Table 4-1. Comparison between simulation data and real world data; iOS

As a final step of prediction test, a prediction of S/W fragmentation on specific date was conducted. Here, the share data of August 2012 was compared between simulation and real world data and one can see that simulated data roughly matches the data of real world (Figure 4-12)

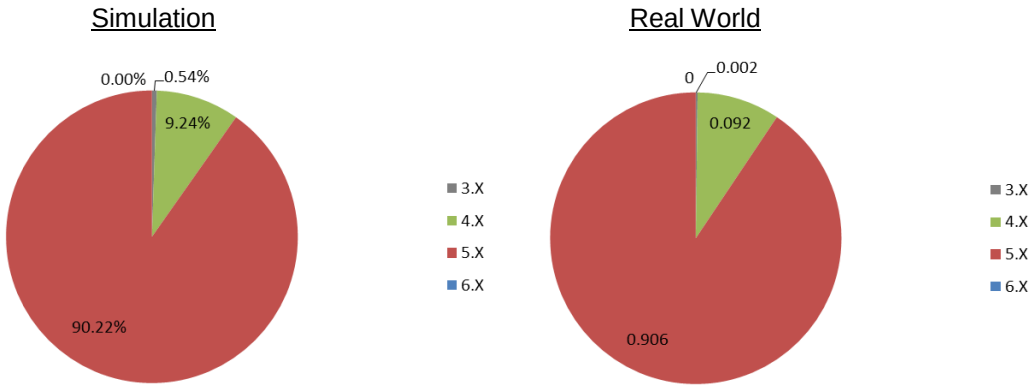


Figure 4-12. Simulation on specific date (August 2012, left: simulation, right: real world)

4.6.2. Modeling Result for Android

As a second step, case study was done on Android. For training data, the shares of installed base of Android Frozen Yogurt and Gingerbread were used. After being trained, the model predicted the shares of installed base of later versions, Android Ice Cream Sandwich and Jellybean. Since publically available data for installed base was only share, not exact number of each installed base, the proposed model was modified to predict the share of each share of installed base version.

Figure 4-13 is the result of training. One can see that simulation data (blue line) fits well with real world data (red line).

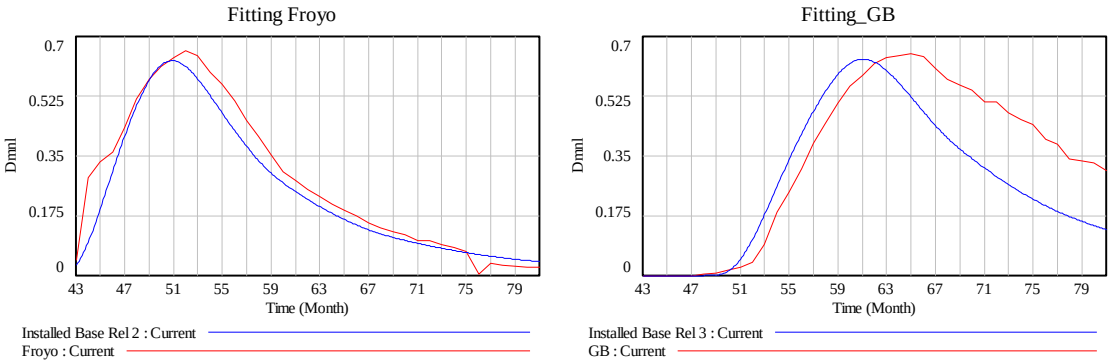


Figure 4-13. Model training (left: Android Froyo, right: Android GB)

Trained model was utility function with parameters of weights and sensitivities as follows.

$$U = 0.5\,S^{0.1} + 0.1\,C^{0.38} - 0.4\,P^{3}$$

The degree of openness was 0.1, which means that the platform was highly opened platform.

Using the utility function and degree of openness above, prediction for share of Android Ice Cream Sandwich and Jelly Bean was conducted. Figure 4-14 shows the result, and one can see that model relatively well predicts the real world data (simulation data: blue line, real world data: red line). The prediction for ICS is not so accurate, since ICS was accepted by many OEMs, because short release interval between ICS and JB made OEMs skip ICS and go directly to JB.

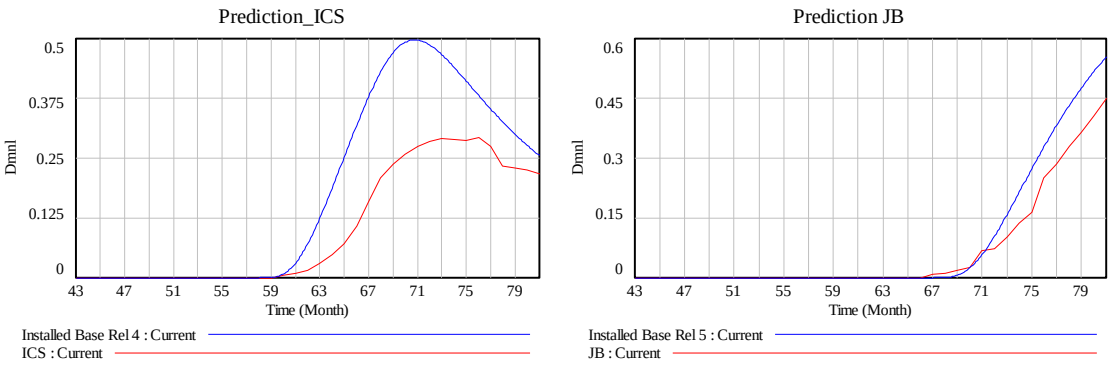
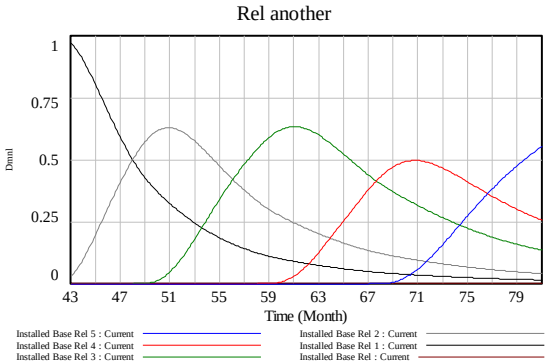
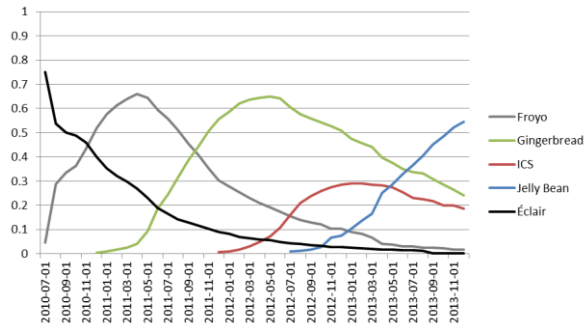


Figure 4-14. Diffusion prediction (left: Android ICS, right: Android JB)

Table 4-2 below is comparison between simulation chart and real world data for share of whole versions (Froyo through JB). Although the simulation does not match exactly with the real world data, it reasonably emulates the real world.

Share (Relative Percentage)

Simulated Data	**Real World Data**
	

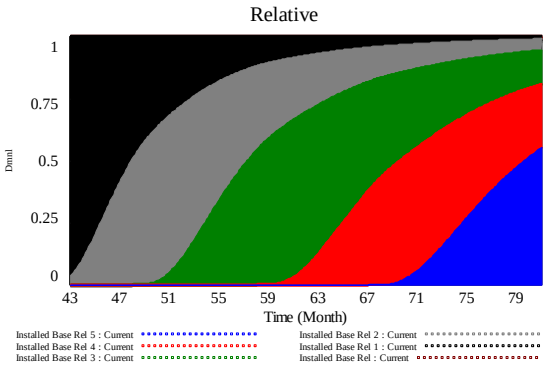
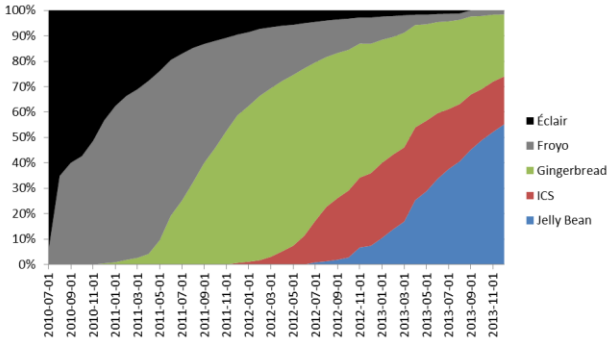
Cumulative Share (Cumulative relative percentage)	
Simulated Data	**Real World Data**
	

Table 4-2. Comparison between simulation data and real world data; Android

As a final step of prediction test, a prediction of S/W fragmentation on specific date was conducted. Here, the share data of July 2013 was compared between simulation and real world data and one can see that simulated data roughly matches the data of real world (Figure 4-15)

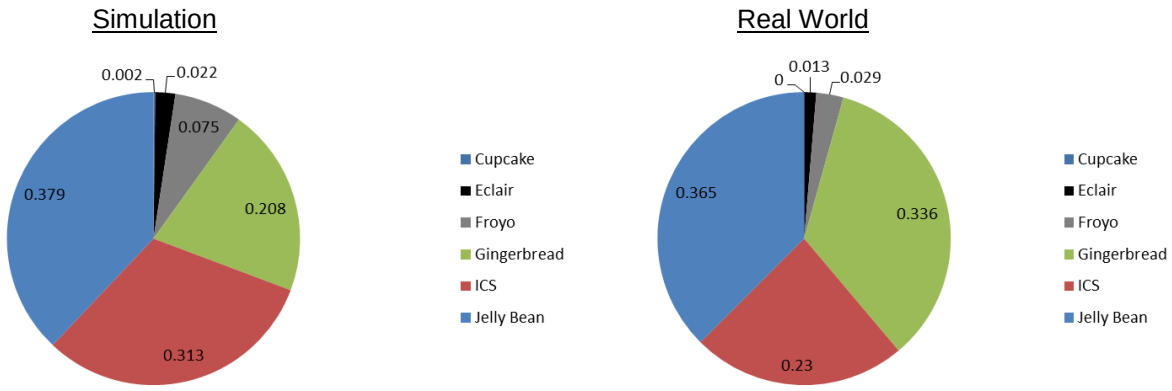


Figure 4-15. Simulation on specific date (July 2013, left: simulation, right: real world)

4.6.3. The effect of open or closed policy on the platform diffusion

After setting up a model and verifying the viability of the model through prediction test, sensitivity analysis was conducted to see how open or closed policy influences the diffusion of platform. Sensitivity analysis was done on Monte Carlo simulation provided by system dynamics simulation software, Vensim.

Vensim runs simulation several times according to parameter setting and give out results for each multiple run. It is easy to see how the change of certain variable influences another variable.

As discussed in section 2.4 and 4.5.2, the proposed model incorporates the open or closed policy based on the estimated degree of openness of platform. Value 1 means closed policy and value 0 means open policy. In our modelling, the case of open policy (Android platform) showed value of 0.1 and the case of closed policy showed value of 0.99. Also, the case study results for iOS and Android showed that the diffusion speed was faster for open platform than for closed platform. Therefore, once the degree of openness is evaluated, the model can predict the diffusion speed of the platform.

In this section, by varying the openness parameter from 0.1 (open platform policy) to 0.99 (closed platform policy), the diffusion speed and the share of installed base were tested on proposed model. As per utility function, parameters of Android case were used. Since S/W fragmentation is more frequently observed in open platform, sensitivity test was conducted one the Android case model, rather than iOS case model. Maintaining all other things equal, only the degree of openness was varied. As was discussed in section 2.4., Openness of 0.5 means the platform's extendibility and modifiability is in between those of Android and iOS. The result is shown in Table 4-3.

The result in table 4-3 shows that as platform policy moves towards opened policy, the diffusion speed of install base increases (the slope of graphs become steeper), and share of each installed base increases.

To better see how the degree of openness changes the diffusion curve, Monte Carlo test was conducted on particular version. Figure 4-16 is the result of Monte Carlo test (the degree of openness was varied from 0.1 to 0.99) conducted on the diffusion of fifth version release. The result showed consistent pattern with Table 4-3; as platform policy moves towards opened policy, the diffusion speed of install base increases, and share of each installed base increases.

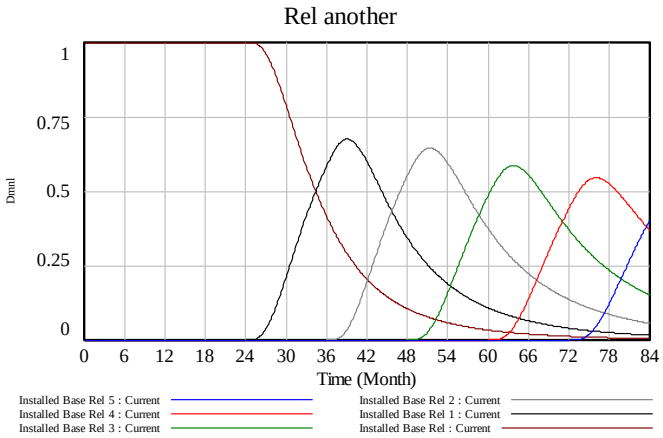
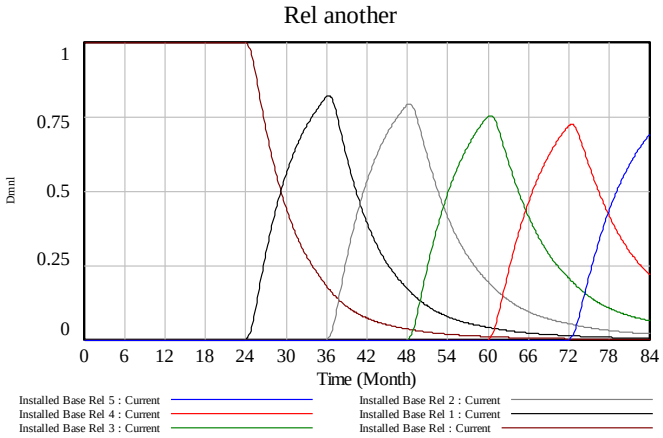
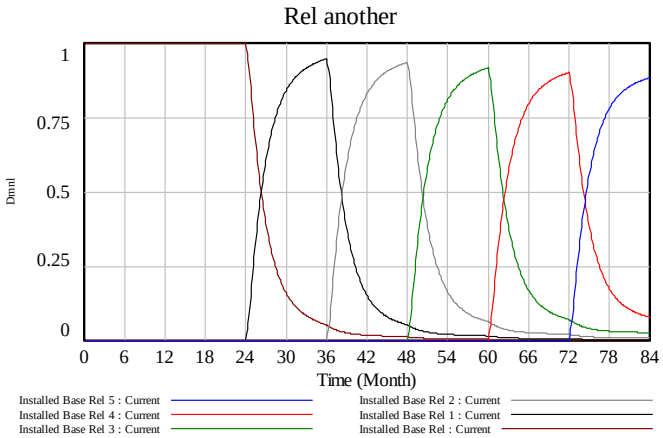
	Sensitivity Test Result
Openness = 0.1 **(open policy)**	
Openness = 0.5 **(half open policy)**	
Openness = 0.99 **(closed policy)**	

Table 4-3. The effect of platform policy on the diffusion of installed base

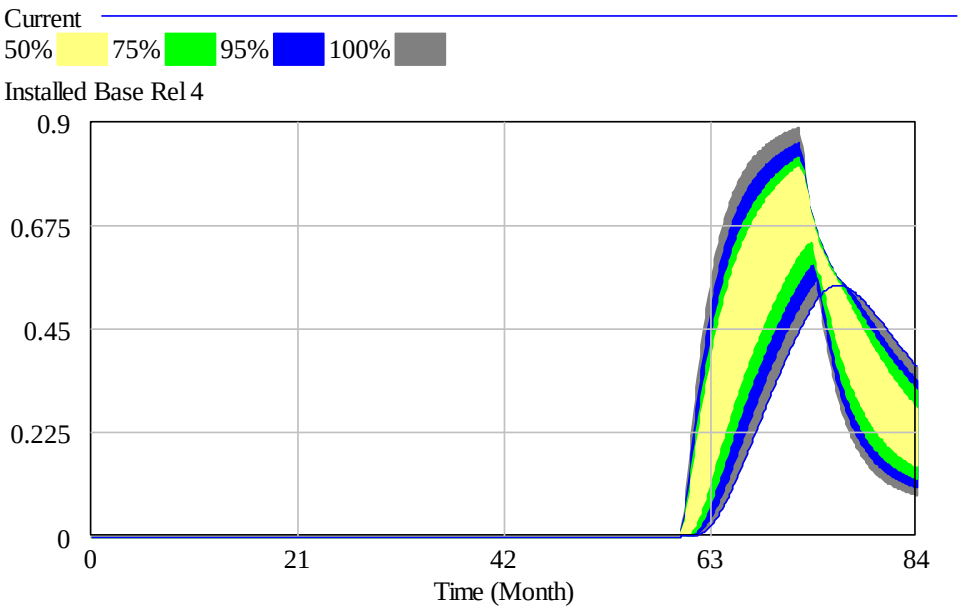


Figure 4-16. Monte Carlo test result for diffusion of selected version

Since proposed model was constructed so that researchers can manipulate release interval, sensitivity test was conducted on release interval, too. By varying the release interval from 8 months to 16months, the diffusion speed and the share of installed base were tested on proposed model. Table 4-4 demonstrates the simulation result. One can observe that short release interval lowers the share of each installed base, slows the diffusion of newly released version, and increases software fragmentation.

To better see how the degree of release interval influences the diffusion curve, Monte Carlo test was conducted on particular version. Figure 4-17 is the result of Monte Carlo test (the release interval was varied from 8 months to 16 months) conducted on the diffusion of fifth version release. The result showed consistent pattern with Table 4-4; as the release interval increases, the diffusion speed of install base increases and share of each installed base increases. This is somewhat intrinsic result, since as the release interval gets shorter, newer version starts to be distributed in the market even before older version is fully distributed and decreases the maximum share of older version. On the contrary, as the release interval gets longer, older version can be distributed in the market as far as it can until new version is introduced and take up the market share.

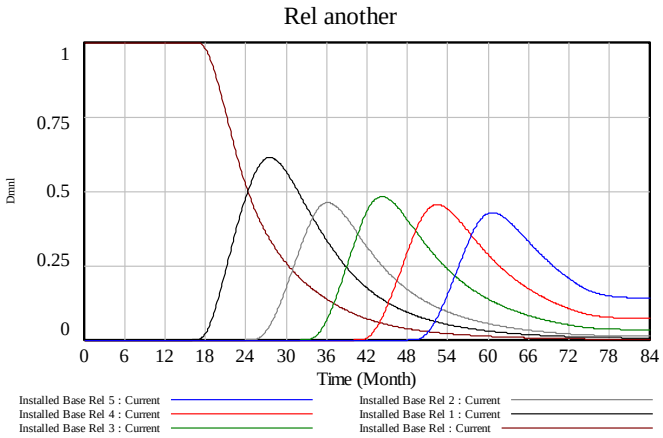
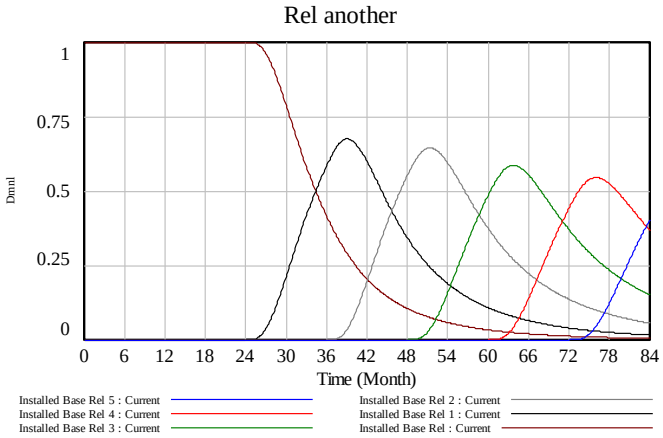
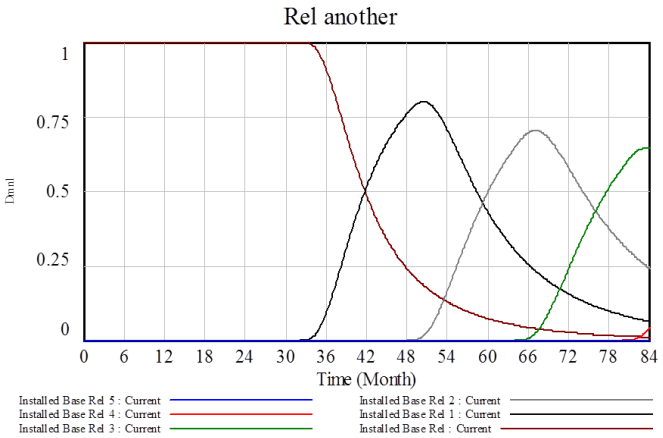
	Sensitivity Test Result
Release Interval = 8 months	
Release Interval = 12 months	
Release Interval = 16 months	

Table 4-4. The effect of release interval on the diffusion of installed base

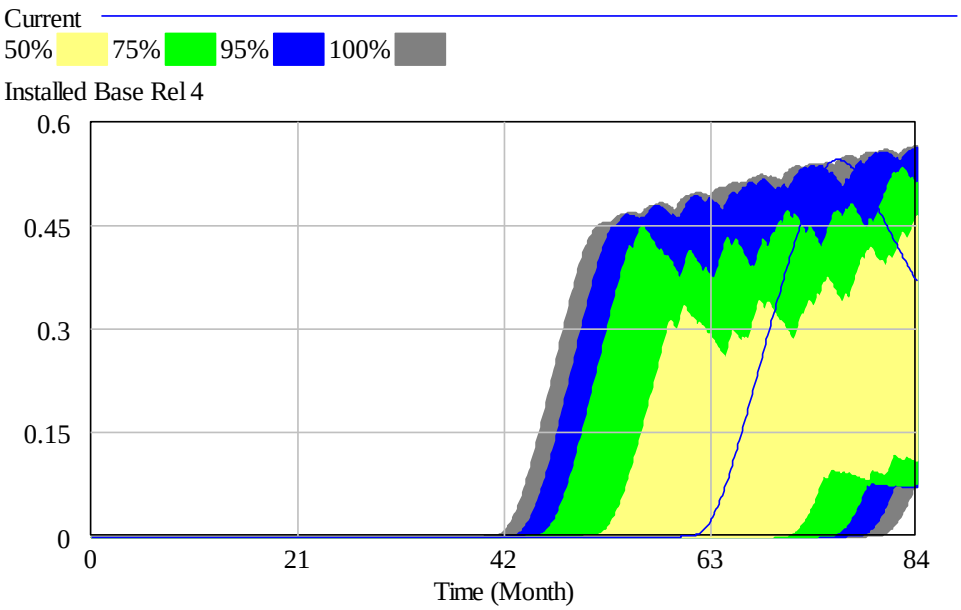


Figure 4-17. Monte Carlo test result for diffusion of selected version

5. Conclusion

5.1. Modelling result interpretation

The study focused on revealing the relationship between platform policy, whether to open or close, and the diffusion pattern of mobile platform. Since the sluggish diffusion of mobile platform results in software fragmentation, that is an issue among platform participants, such as OEMs and complementary product providers. To unravel the causality between platform policy and platform diffusion, as a first step, system dynamics model was setup. Next, to verify how well the proposed model emulates the real situation, the model was employed to predict the diffusion of mobile platform. After verifying the validity of the model, as a final step, sensitivity analysis was conducted. Sensitivity analysis showed how platform policy influences the diffusion of mobile platform and thereby software fragmentation. The test brought about several insightful results.

(1) the platform policy whether to open or close the platform influences the diffusion pattern of a platform. According to the case study done on open policy (iOS) and closed policy (Android), platform deploying proprietary policy resulted in faster diffusion rate (not speed) and less software fragmentation. Meanwhile, platform employing opened policy resulted in slower diffusion rate (not speed) and more software fragmentation. Main reason for the difference is that the degree of openness brings about delays between new firmware release and actual adoption from customers.

First delay is the development time, which is incurred by OEMs who customizes and manufactures handsets onto new operating system. Whenever platform intermediary publishes new version, OEMs require time to develop and test products. The time takes for new product launch is different from each OEMs and each product complexities.

Second delay is related with software upgrade time. Nowadays it is industry norm to provide free software upgrade service for post sales products. However, even between product lines produced by same OEMs get different priorities according to product management policy: flagship products get most of resources for software upgrade and release schedule for them are ahead of other less important products. The overlap of such difference in upgrade service readiness further slows down the diffusion of new operating system and increases software fragmentation.

Thirdly, average upgrade rate is also affected by platform policy. Open policy slows down the diffusion of devices with up-to-date software. As customers do not get proper upgrade service and remain using older version, the share of older installed base do not decrease fast, giving rise to software fragmentation.

(2) the release interval between each version of operating system influenced the diffusion of platform. Sensitivity test result showed that release interval does not influence the diffusion speed. Rather, it influenced the share of installed base of released software. As the release interval becomes larger, the share of new version increased, and vice versa. One can see that too frequent version release results increases the software fragmentation, since newer version in continuously released even before majorities of customers adopt pre-existing software.

5.2. Managerial implication

The difference in the diffusion pattern of open and closed platforms was manifested by the model proposed in this study. Open platform does not depend on OEMs' development time or S/W upgrade time. Therefore, the diffusion pattern of open platform is faster and fragmentation of it is less observed. For platform intermediaries with open policy, it is important to increase the utility of a platform to accelerate diffusion speed. Meanwhile, the diffusion pattern of closed platform depends on the OEMs' ability to introduce products fast and provide upgrade service early. As demonstrated in the modeling, to maximize the diffusion of newest version in the market, it is important to shorten such delays in the diffusion of platform.

There's conflict between platform intermediaries and platform participants regarding the delay. Assume that both of them are concerned about software fragmentation. Platform intermediaries' main concern would be reducing the fragmentation and fasten the diffusion of new operating system. They would want OEMs' delay in customization that delays the schedule of new product launching or upgrade service. Also, increased fragmentation makes OEM's effort to support every software version harder. However, for OEMs, fragmentation is like flip side of the coin. Without customizing the software, it is hard to differentiate their devices from other OEM's devices. Also, OEMs are under pressure to customize software according to requirements from network operators. Therefore, for platform intermediaries with open policy, it is important to control the factors that delay OEMs' product or service development time, such as customization or device development with older software.

The power of proposed model is that it can predict the diffusion speed and share of each software versions. Using the model, product life cycle of each version can be estimated. Currently, upgrade policy of each OEMs are different and sometimes the policy is applied differently according to product segment. The model will provide a systematic base in situations where managers have to decide on to what extent they have to provide service, looking at predicted product life cycle and market share of each installed base.

Also, complementary service or product providers would able to foresee how many versions they have to support based on the proposed model.

As demonstrated by the modeling and sensitivity analysis, it is hard to dramatically reduce fragmentation of open platform without discouraging the usage of customer's preference for older firmware or OEM's dependence on older firmware. Granting the nature of open platform, managers of OEMs or application developers can avoid unnecessary discussion about fragmentation itself and turn to the discussion of how many versions they have to support. Without concrete market forecast, it is hard to make decision on how many times they have to provide services. The tool will help those managers' decision making and provide clear view about market adoption.

To handle the fragmentation issue, Google started to set the deadline for the submission of OEM's own customized operating system. Once the deadline is passed, OEM cannot get certification to preload google mobile service on their devices. Google hopes the policy help reduce fragmentation by impeding OEM's excessive dependence on older versions. However, to reduce delays incurred by OEMs, it would be necessary to enact direct controls, such as limiting the number of preloaded OEM's own applications.

6. Further Study

6.1. Limitations of the study

The study still has several limitations. The proposed model does not consider intra platform competition. Customer's choice of mobile operating system not only depends on the value of the platform, but also depends on other platform. However, the model proposed in this study focuses on the value of one platform, not the interaction among competing operating systems.

Also, the model does not include feature or quality of each version. We see customers do not voluntarily upgrade firmware because of price barrier, satisfaction with current version, safety, quality, or discomfort of upgrade. Sometimes these characteristics impede or accelerate customer's adoption of newer versions. However, proposed model assumes that each version has same degree of feature, satisfactoriness, and quality. To make the model more accurate, these factors need to be incorporated.

The model did not incorporate the causality among the platform and complementary product (the number of application) and average sales price. There exists indirect or direct network effect between the value of platform and the number of applications on the app store. Also, there is network effect between the value of platform and the average sales price of devices. For simplification, the model proposed here used real data as an input, and did not fully emulate causality between those variables.

Finally, the study mostly represents product and service providers' viewpoint. While fragmented operating system is an issue for platform intermediaries and OEMs, it has positive aspects also. Fragmentation itself broadens customer's choice. Not only that, huge portion of smartphone users favors low end devices made by small OEMs. And these devices are only run on previous old and light operating system whose H/W and S/W requirement is lower than newer operating system.

6.2. Proposal for future study

The proposed model did not internalize the dynamics between the value of platform and platform participants. Rather, the model used historical data to get the number of complementary products, as a representative of platform participants. There must be causal relationship between the value of platform and entry of complementary product or service providers. Incorporating such relationship would make the model more concrete and enable researchers find more insights from it.

Also, the model did not incorporate causal relationships running through organizations, such as experience curve, pricing, and capacity improvement. Including these factors would let researchers figure out how intra organizational decisions influences the diffusion of mobile platforms. It is also possible to extend the model to predict the revenues of platform participants. To illustrate, once average selling price of applications and number of downloads per user are given, total sales revenue from each version can be predicted from the model.

In addition, the study only discusses the cases of two successful platforms. It would be beneficial to test the model on other operating system's diffusion to verify the model's applicability. In that the proposed model has endogenous variable that incorporates the degree of openness, once the degree of openness of specific platform is evaluated, the proposed model would be able to simulate the diffusion and fragmentation pattern of the platform.

Lastly, it would be a good practice to apply the model in examining the diffusion pattern of other mobile platforms with different degree of openness, such as Blackberry and Linux Mobile. One can decide the degree of openness by analyzing the modifiability or extendibility of each software layers. It is also possible to get real world data for the number of applications and average selling price. Once necessary data is gathered, it is possible to test the diffusion pattern of the platform using the proposed model.

Reference

Baldwin, Carliss Young, and Eric Von Hippel. *Modeling a paradigm shift: From producer innovation to user and open collaborative innovation*. Harvard Business School, 2009.

Basole, Rahul C. "Visualization of Interfirm Relations in a Converging Mobile Ecosystem." *Journal of Information Technology* 24.2 (2009): 144–159.

Bass, Frank M. "Comments on "a new product growth for model consumer durables the bass model"." *Management science* 50.12_supplement (2004): 1833-1840.

Boudreau, Kevin. "Open Platform Strategies and Innovation: Granting Access vs. Devolving Control." *Management Science* 56.10 (2010): 1849–1872.

Butler, Margaret. "Android: Changing the Mobile Landscape." *IEEE Pervasive Computing* 10.1 (2011): 4–7.

Casey, Thomas R., and Juuso Töyli. "Mobile Voice Diffusion and Service Competition: A System Dynamic Analysis of Regulatory Policy." *Telecommunications Policy* 36.3 (2012): 162–174.

Cho, Yun Chan, and Jae Wook Jeon. "Current software platforms on mobile phone." *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*. IEEE, 2007.

Cuadrado, Felix, and Juan C. Duenas. "Mobile application stores: success factors, existing approaches, and future developments." *Communications Magazine, IEEE* 50.11 (2012): 160-167.

Cusumano, Michael A. *Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Uncertain World (lessons from Microsoft, Apple, Intel, Google, Toyota and More)*. Oxford University Press, 2010.

Cusumano, Michael A., Yiorgos Mylonadis, and Richard S. Rosenbloom. "Strategic maneuvering and mass-market dynamics: The triumph of VHS over Beta." *Business history review* 66.01 (1992): 51-94.

Eisenmann, Thomas, Geoffrey Parker, and Marshall W. Van Alstyne. "Strategies for two-sided markets." *Harvard business review* 84.10 (2006): 92.

Gandhewar, Nisarg, and Rahila Sheikh. "Google Android: An Emerging Software Platform For Mobile Devices." *International Journal on Computer Science & Engineering* (2011).

Haile, Netsanet, and Jörn Altmann. "Estimating the Value Obtained from Using a Software Service Platform." *Economics of Grids, Clouds, Systems, and Services*. Springer International Publishing, 2013. 244-255.

Hall, Sharon P., and Eric Anderson. "Operating Systems for Mobile Computing." *Journal of Computing Sciences in Colleges* (2009): 64–71.

Jansen, Slinger, Michael A. Cusumano, and Sjaak Brinkkemper, eds. *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Edward Elgar Publishing, 2013.

Katz, Michael L., and Carl Shapiro. "Network externalities, competition, and compatibility." *The American economic review* (1985): 424-440.

Katz, Michael L., and Carl Shapiro. "Systems competition and network effects." *Journal of economic perspectives* 8 (1994): 93-93.

Kenney, Martin, and Bryan Pon. "Structuring the Smartphone Industry: Is the Mobile Internet OS Platform the Key?" *Journal of Industry, Competition and Trade* 11.3 (2011): 239–61.

Lin, Feida, and Weiguo Ye. 2009. "Operating System Battle in the Ecosystem of Smartphone Industry." *2009 International Symposium on Information Engineering and Electronic Commerce* (2004): 617–621.

Maier, Frank H. "Innovation Diffusion Models for Decision Support in Innovation Management" *System Dynamics* 2 (1995): 656–665.

Maier, Frank H. "New product diffusion models in innovation management—A system dynamics perspective." *System Dynamics Review* 14.4 (1998): 285-308.

Müller, Roland M., Bjorn Kijl, and Josef KJ Martens. "A comparison of inter-organizational business models of mobile App Stores: there is more than open vs. closed." *Journal of theoretical and applied electronic commerce research* 6.2 (2011): 63-76.

Nikou, Shahrokh, and Harry Bouwman. "Mobile service platform competition." (2012).

Norton, John A., and Frank M. Bass. "A diffusion theory model of adoption and substitution for successive generations of high-technology products." *Management science* 33.9 (1987): 1069-1086.

Pagani, Margherita, and Charles H. Fine. "Value network dynamics in 3G–4G wireless communications: A systems thinking approach to strategic value assessment." *Journal of Business Research* 61.11 (2008): 1102-1112.

Parker, Geoffrey G., and Marshall W. Van Alstyne. "Two-Sided Network Effects: A Theory of Information Product Design." *Management Science* 51.10 (2005): 1494–1504.

Rahmandad, Hazhir, and Soheil Sibdari. "Joint Pricing and Openness Decisions in Software Markets with Reinforcing Loops." *System Dynamics Review* 28.3(2012): 209–229.

Raymond, Eric. "The cathedral and the bazaar." *Knowledge, Technology & Policy* 12.3 (1999): 23-49.

Rogers, Everett M. *Diffusion of innovations.* Simon and Schuster, 2010.

Sterman, John D. "System Dynamics Modeling: TOOLS FOR LEARNING IN A COMPLEX WORLD." *California management review* 43.4 (2001).

Sun, Mingchun, and Edison Tse. "When does the winner take all in two-sided markets?." *Review of Network Economics* 6.1 (2007).

Thun, Jörn-Henrik, Andreas Größler, and Peter M. Milling. "The diffusion of goods considering network externalities: a system dynamics-based approach." *Sustainability in the Third Millennium. Systems Dynamics Society, Albany* (2000): 204-1.

Zhu, Feng, and Marco Iansiti. *Dynamics of platform competition: Exploring the role of installed base, platform quality and consumer expectations.* Division of Research, Harvard Business School, 2007.

Appendix: Model documentation

Active End Users= INTEG (Purchasing Rate,0)
Units: person

Adoption Fraction=0.99024
Units: Dmnl

Adoption Rate=New Product Purchase Rate
Units: unit/Month

Adoption Rate 1=(New Product Purchase Rate 1+"S/W Upgrade Rate 1")
Units: unit/Month

Adoption Rate 2=(New Product Purchase Rate 2+"S/W Upgrade Rate 2")
Units: unit/Month

Adoption Rate 3=(New Product Purchase Rate 3+"S/W Upgrade Rate 3")
Units: unit/Month

Adoption Rate 4=(New Product Purchase Rate 4+"S/W Upgrade Rate 4")
Units: unit/Month

Adoption Rate 5=(New Product Purchase Rate 5+"S/W Upgrade Rate 5")
Units: unit/Month

Alpha=0.000857785
Units: 1/Month

APP:INTERPOLATE::=GET XLS DATA('APP.xls', 'APP', '1', 'B2')
Units: Dmnl

ASP:INTERPOLATE::=GET XLS DATA('ASP.xls', 'ASP', '1', 'B2')
Units: Dmnl

Attrition Rate=DELAY1(Adoption Rate*(1-Upgrade per person),Product Life Cycle)
Units: unit/Month

Attrition Rate 1=DELAY1(Adoption Rate 1*(1-Upgrade per person),Product Life Cycle)
Units: unit/Month

Attrition Rate 2=DELAY1(Adoption Rate 2*(1-Upgrade per person),Product Life Cycle)
Units: unit/Month

Attrition Rate 3=DELAY1(Adoption Rate 3*(1-Upgrade per person),Product Life Cycle)
Units: unit/Month

Attrition Rate 4=DELAY1(Adoption Rate 4*(1-Upgrade per person),Product Life Cycle)
Units: unit/Month

Attrition Rate 5=DELAY1(Adoption Rate 5*(1-Upgrade per person),Product Life Cycle)
Units: unit/Month

Average time to Develop=0.5/Openness
Units: Month

Average time to Upgrade=5/Openness
Units: Month

Average Growth Rate=0.0115/12
Units: 1/Month

Beta=0.0061331
Units: Dmnl

Contact Rate=7
Units: 1/Month

Factor=0.102448
Units: Dmnl

Factor 2=0.375033
Units: Dmnl

Factor 3=3
Units: Dmnl

Growth=Average Growth Rate*Population
Units: person/Month

Innovator=Potential End Users*Alpha
Units: person/Month

Installed Base= INTEG (Adoption Rate-Attrition Rate-Upgrade,1)
Units: unit

Installed Base 1= INTEG (Adoption Rate 1-Attrition Rate 1-Upgrade1,0)
Units: unit

Installed Base 2= INTEG (Adoption Rate 2-Attrition Rate 2-Upgrade2,0)
Units: unit

Installed Base 3= INTEG (Adoption Rate 3-Attrition Rate 3-Upgrade3,0)
Units: unit

Installed Base 4= INTEG (Adoption Rate 4-Attrition Rate 4-Upgrade4,0)
Units: unit

Installed Base 5= INTEG (Adoption Rate 5-Attrition Rate 5-Upgrade5,0)
Units: unit

Installed Base Rel=Installed Base/Total Installed Base
Units: Dmnl

Installed Base Rel 1=Installed Base 1/Total Installed Base
Units: Dmnl

Installed Base Rel 2=Installed Base 2/Total Installed Base
Units: Dmnl

Installed Base Rel 3=Installed Base 3/Total Installed Base
Units: Dmnl

Installed Base Rel 4=Installed Base 4/Total Installed Base
Units: Dmnl

Installed Base Rel 5=Installed Base 5/Total Installed Base
Units: Dmnl

New Product Purchase Rate=DELAY3(Purchasing Rate*Purchase per person*Utility*Release,Average time to Develop)
Units: unit/Month

New Product Purchase Rate 1=DELAY3(Utility*Purchase per person*Purchasing Rate*Release 1,Average time to Develop)
Units: unit/Month

New Product Purchase Rate 2=DELAY3(Utility*Purchase per person*Purchasing Rate*Release 2,Average time to Develop)
Units: unit/Month

New Product Purchase Rate 3=DELAY3(Utility*Purchase per person*Purchasing Rate*Release 3,Average time to Develop)
Units: unit/Month

New Product Purchase Rate 4=DELAY3(Utility*Purchase per person*Purchasing Rate*Release 4,Average time to Develop)
Units: unit/Month

New Product Purchase Rate 5=DELAY3(Utility*Purchase per person*Purchasing Rate*Release 5,Average time to Develop)
Units: unit/Month

Openness=0.1
Units: Dmnl

Population= INTEG (Growth,6.5521e+009)
Units: person

Potential End Users=MAX (0,Population*Adoption Fraction-Active End Users)
Units: person

Product Life Cycle=18
Units: Month

Purchase per person=1
Units: unit/person

Purchasing Rate= Innovator+Word of Mouth
Units: person/Month

Release=STEP(1,1*Release Interval)-STEP(1,2*Release Interval)
Units: Dmnl

Release 1=STEP(1,2*Release Interval)-STEP(1,3*Release Interval)
Units: Dmnl

Release 2=STEP(1,3*Release Interval)-STEP(1,4*Release Interval)
Units: Dmnl

Release 3=STEP(1,4*Release Interval)-STEP(1,5*Release Interval)
Units: Dmnl

Release 4=STEP(1,5*Release Interval)-STEP(1,6*Release Interval)
Units: Dmnl

Release 5=STEP(1,6*Release Interval)-STEP(1,7*Release Interval)
Units: Dmnl

Release Interval=12
Units: Month

S/W Upgrade Rate 1=Upgrade*Release 1
Units: unit/Month

S/W Upgrade Rate 2=Release 2*Upgrade1
Units: unit/Month

S/W Upgrade Rate 3=Release 3*Upgrade2
Units: unit/Month

S/W Upgrade Rate 4=Release 4*Upgrade3
Units: unit/Month

S/W Upgrade Rate 5=Release 5*Upgrade4
Units: unit/Month

Total Installed Base=Installed Base+Installed Base 1+Installed Base 2+Installed Base 3+Installed Base 4 +Installed Base 5
Units: unit

Upgrade=DELAY3(Adoption Rate*Upgrade per person, Average time to Upgrade)

Units: unit/Month

Upgrade per person=0.9*Openness
Units: 1/Month

Upgrade1=DELAY3(Adoption Rate 1*Upgrade per person, Average time to Upgrade)
Units: unit/Month

Upgrade2=DELAY3(Adoption Rate 2*Upgrade per person, Average time to Upgrade)
Units: unit/Month

Upgrade3=DELAY3(Adoption Rate 3*Upgrade per person, Average time to Upgrade)
Units: unit/Month

Upgrade4=DELAY3(Adoption Rate 4*Upgrade per person, Average time to Upgrade)
Units: unit/Month

Upgrade5=DELAY3(Adoption Rate 5*Upgrade per person, Average time to Upgrade)
Units: unit/Month

Utility=MAX(Weight*(XIDZ(Total Installed Base,Active End Users*Purchase per person,0))^Factor+Weight 2*APP^Factor 2-Weight 3*ASP^Factor 3,0)
Units: Dmnl

Weight= 0.5
Units: Dmnl

Weight 2=0.1
Units: Dmnl

Weight 3=1-Weight-Weight 2
Units: Dmnl

Word of Mouth=Active End Users*Beta*Contact Rate*Potential End Users/Population
Units: person/Month