Strategic Management Surrounding Firm Specialization and Competitiveness in the Mobile Phone Industry: Impact of Firms' Software Contributions on Time to Market

By
Talha Mushtaq

****

Submitted in Partial Fulfillment
Of the requirements for
Honors in the Department of Economics and for
Honors in the Department of Computer Science

Union College
June 2023

Abstract

Acknowledgements: I would like to thank my advisors, Dr. Ercan Karadas and Dr. John Rieffel, for their guidance throughout the thesis process.

Google's Android Operating System (OS) is open-source software available for public use. Several smartphone manufacturers (Original Equipment Manufacturers or OEMs) like Samsung and LG use and contribute to the Android Open-Source Project (AOSP). These contributions can be tracked using their commits to the AOSP, a public data set. This research will explore the relationship between a firm's or an OEM's contributions to AOSP and marketing a product based on the contribution type and date, which could ultimately contribute to the success of a phone model and the firm. These contributions to AOSP can be in the form of a firm's exploratory and exploitative learning, which can be related to the Time to Market (TTM) of a product, which would be a phone model in this research. I will use the mobile telecom data on Android commits (software contributions) to understand the difference between exploratory and exploitative learning of these Android OEMs. The dataset consists of the complete population of Android commits written by software engineers worldwide between 2008-2022 (approximately 16 million observations), which Prof. Dallas and his colleagues have collected over a period. Because Android is open source, this is free and open for public use. The data set highlights some handset (mobile phone) companies contributing to Android and to which release (different releases of Android), and how quickly they come up with a handset (or phone) after the new version is released that version of the Android OS. I will be researching the time and type (exploratory or exploitative) of these contributions and exploring what affects their TTM. This research could potentially lead to discovering strategic management techniques and timed marketing in the Information and Communications Technology (ICT) industry, focusing on firms that use and contribute to the Android OS.

Contents

1 Introduction

Researchers have observed that the pace of technological progress has not only accelerated but has also been altering at an accelerated rate in recent decades. Researchers had previously used the evolutionary biology phrase 'punctuated equilibrium' to describe the pace of technological progress, in which short bursts of radical technical change were alternated by extended periods of gradual change [45, 56]. In the 1990s, however, it became clear that technological change had no periods of relative stability and was always rapid, requiring businesses to adapt their routines and organizational structures [17, 36, 54]. Strategic management scholars developed a slew of new ideas about the speed of strategic decision-making [18], pacing strategic change [21], accelerating adaptive processes [19], organizational response, and a heap of new ideas about structured chaos and time [20, 8].

According to research, the largest earnings are made via routine and periodic innovation. This is sometimes referred to as "generative appropriability" [2], and it entails the continuous introduction of generational product improvements that maintain long-term consumer appeal, resulting in an accumulation of short-term advances with significant implications for a company's long-term success. Companies pursue innovative business ventures to enter new market categories with first-mover advantages to achieve competitive advantages [44]. This research will explore these first-mover advantages and strategic management by mobile handset manufacturers (called original equipment manufacturers, or OEMs), which utilize and contribute to the open-source Android Operating System (OS).

GTC (Generational Technological Change) refers to significant changes in the environment that other businesses confront [29]. Generational Product Innovation (GPI), on the other hand, is typically defined as discrete inventions that companies introduce on a regular basis. Android's OEMs are the original equipment manufacturers who utilize the Android OS software such as Samsung, LG, and NUU Mobile. These Android OEMs often contribute to Android Open-Source Project (AOSP), which is an open-source operating system development project maintained by Google. Being open source, anyone is free to review and contribute code and fixes to the project repository [53]. This research will focus on exploring the relationship between the success of a new handset model (a GPI) based on Android's OEMs changes in strategies in contributing to AOSP, whenever Google introduces a new-found version of the Android OS (a GTC), which occurs approximately once a year. The Android OEMs strategies in contributing to AOSP can be explored using exploitative and exploratory learning based on an OEMs git commits to AOSP. Exploitative learning in this case is when the firms or OEMs leverage their resources and knowledge to improve performance and fix bugs in the current or old generation of the Android version. Whereas exploratory learning is when the firms focus on the new generation of the Android version and try to learn about the new technology [29, 51]. Exploitative and exploratory learning concerning the new release of the Android OS (a GTC) can affect a firm's Time to Market (TTM) and its new product (a GPI), which could eventually contribute to the success of the product and the firm. This research will try to investigate a relationship between exploitative

and exploratory learning, using the Android commits data, with TTM.

2 Literature Review

2.1 Time to Market

Time to Market (TTM) is a well-established concept in business and product development that emphasizes the importance of bringing a product or service to market quickly. Various studies and articles have demonstrated the significant impact of TTM on a company's success and competitive advantage.

According to research by Cooper and Kleinschmidt (1995), reducing TTM is crucial for achieving higher market share and profitability [14]. Their study analyzed a large sample of product development projects and found a clear correlation between shorter TTM and improved business performance. Another study conducted by Griffin and Page (1993) further supports the notion that TTM is a critical factor for success [24]. Their research examined new product introductions in the consumer electronics industry and revealed that companies with faster TTM had a higher likelihood of achieving market leadership and superior financial performance. The study emphasized that speed to market is not merely about launching a product quickly but also about effective execution and meeting customer needs in a timely manner. Furthermore, an article by Smith and Reinertsen (1998) highlights the strategic significance of TTM [49]. They argue that in today's fast-paced and competitive markets, the ability to deliver products rapidly and respond swiftly to customer demands is a key driver of success. The article also emphasizes the importance of cross-functional collaboration, efficient decision-making processes, and agile development methodologies to achieve shorter TTM.

These studies and articles collectively provide empirical evidence and industry insights that substantiate TTM as a proven concept. They underline the positive impact of reducing TTM on market share, profitability, and overall business performance. Implementing strategies and practices to expedite product development, streamline processes, and foster innovation can empower companies to gain a competitive edge and capitalize on market opportunities. By embracing TTM as a core principle, organizations can enhance their ability to adapt to changing market dynamics, meet customer expectations, and stay ahead of the competition.

My research will build on the previous and present research, calculating exploitative and exploratory learning from the up-to-date Android commits data set, and ultimately investigating the effect of exploitative and exploratory learning on TTM, which is defined as the speed and productivity of product development [7, 19, 32] and is a well-supported concept.

2.2 Defining Generational Technology Shifts

There are previous and present research on the concept of GTC and GPI, and how they effect a firm's performance. Lawless and Anderson's (L&A's) research in the area has focused on defining GTC as "a significant advance in technical performance within a technological regime"; their study looked at how generational shift affects firm performance, as well as the evolution of niches, or firm positions in a product-market sector [36]. LA observed that, in general, performance depends more on how companies maneuver in relation to local rivals (competitors within a niche) than on discovering and occupying a suitable niche in fast-paced innovation. This implies that success in a technologically dynamic environment is not solely determined by finding the right niche, but also by effectively maneuvering within the competitive landscape and establishing advantageous positions relative to local rivals. Understanding the dynamics of local rivalry becomes crucial for firms aiming to capitalize on the opportunities brought by a GTC. By considering LA's study, we can further explore the effects of GTC and its implications for firm performance.

L&A's definition of a GTC has been used by others, but the basic concept has also been extended to adjacent concepts with similar terminology, like "Generational Product Innovation" (GPI). In numerous GPI studies, authors directly quote L&A and define GPIs in ways that are almost exact with GTC. As an illustration, Chen et al. (2021: 793) "[focuses] on generational product innovation (GPI), defined as the improvement of an established product that results in significant functional and technical advances within a technological regime" In addition, according to Turner et al. (2013: 1853) "generational product innovations are substantial advances in the technical performance of existing products within technological regimes" [11, 26, 55, 36]

GPIs are essential not only for value capture from "generative appropriability" but also because each new GPI has an interactive impact on demand for prior generation releases, and sometimes new releases can alienate customers by disrupting their ingrained behaviors and devaluing their habitual product knowledge [11, 2, 3, 31, 46]. The significance of GPIs has given rise to management literature on how to optimize product launches, the value of a workforce with experience in production generation, and how to balance internal rhythms and external events when innovating and releasing GPIs [38, 40, 55].

We contend, however, that Lawless and Anderson's original notion of GTC contains significant and underappreciated distinctions from seemingly similar concepts like GPI, despite the apparent significance of GPIs. The main cause of the "significant advances in technical performance" is one important distinction between them. L&A's empirical research focuses on the comparative performance of PC manufacturers between 1982 and 1991. However, PC producers do not lead technical advancements. Instead, according to L&A, PC makers must operate "within the technological regime defined by Intel microprocessors and Microsoft operating systems." According to L&A's framework, Intel and Microsoft are the companies who establish the technology regime (environmental circumstances) and advance the GTC. As opposed to this, a different group of companies (PC manufacturers) carefully select "different technology vintages [that] coexist at different price points" to offer GPIs inside this developing regime. These "technology vintages"

refer to the GTCs that Microsoft and Intel have unveiled [36].

The more current GPI research stream seems to have forgotten this distinction. For example, in the research of mobile gaming apps, there are no companies like Intel and Microsoft that structure the technological regime or externally propel the GTC forward [11]. Instead, it is up to each individual app developer to decide whether to release a GPI — a new version of their software. They specifically discover that app developers must strike a balance between the risk of alienating their users and the tension between releasing a new GPI to capture more value (generative appropriability) because the new GPI may destroy consumer value by upsetting their established behavioral patterns and imposing learning costs. According to Chen et al. (2021), more significant alterations to the app developer environment, such as Android or Apple's decision to routinely upgrade their OS (a GTC), have no bearing on the strategic decisions made by their app developers. As a result, OS upgrades are treated as background variables in their regressions.

Given their study questions, Chen et al. (2021) are not incorrect in their decision to ignore OS upgrades. Instead, their GPI is different from L&A's GTC because a GTC is a significant technological advancement that establishes the technological regime and acts as the catalyst for change. Given that a single app developer does not constitute a technological regime, Chen et al. (2021)'s GPI should be conceptually distinguished from GTCs. Despite using L&A, other studies have collapsed GPIs and GTCs together or back-grounded GTC events (such as OS upgrades) [55]. In a definitional manner, GPIs are best viewed as distinct product innovations created by the focal firms of interest to the researcher (PC makers in L&A or app developers in Chen et al. 2021). GTCs, on the other hand, are best viewed as relatively separate occurrences that the research's focal enterprises (such as Intel and Microsoft in L&A or OS system changes) are affected by. Despite this, GTCs and GPIs both take place inside the same technological environment [29].

Should GTC and GPI always be distinguished clearly? In the end, we think this is a decision made by the researchers based on their research questions. We do, however, think that the degree of flexibility in the strategic decisions made by focal businesses relative to GPIs is a crucial indicator of when a GTC should be distinguished from a GPI, and that by failing to do so, researchers run the risk of misinterpreting the strategic environment of GPIs.

In truth, the literature has a very diverse view of how to differentiate GTCs from GPIs. According to Chen et al. (2021: 794) "innovation introduction is a deliberate choice made by firms," they are clear in putting their entire attention on the strategic decision of their focal firms (and so back-grounding any GTC).

Other GPI research, such as that by Turner et al. (2010) on PC software companies, does take the external environment into account [55]. They view the degree of industry concentration as the external environment, which is a product of dynamics within the PC software companies themselves. Operating system generational changes, in contrast, are ignored. However, despite their invocation of L&A, a GTC is still missing from their work because industrial concentration does not drive the technical regime on its own.

As opposed to L&A, where the focal firms (PC manufacturers) are the ones driving the important environmental changes, Intel and Microsoft are external companies that operate in separate sectors. However,

since their target companies are independent of Intel or Microsoft, each generation of microprocessors does not limit or restrict the options available to them. In reality, new microprocessor generations expand the range of options for PC manufacturers who may now combine microprocessors at various price points, further complicating the market and fostering the emergence of new niches. However, Intel's latest chip generation is a GTC that establishes each new technological regime, affecting PC makers. Although L&A does not use the word GPI, the growing number of microprocessor options (GTCs) forces PC manufacturers to reevaluate their product plans (GPIs). Since GTCs and GPIs are discrete processes, it is best to distinguish between the two ideas.

In our study, GTC and GPI differ even more sharply, having a significant effect on how organizations innovate and learn. When Android publishes a new version of its operating system, a GTC happens. As a result, OEMs start a new round of GPIs and create a new handset model based on the updated OS. The GPI comprises studying and improving the open-source software code that forms the basis of Android as well as testing and optimizing the numerous dependencies between the Android OS, the OEMs' hardware, and their proprietary firmware. This is a common characteristic of various platform ecosystems, however, empirical work can be focused on only one platform (i.e., Android OS) [29].

However, with a platform environment like Android, a new OS version "forces" all major OEMs to start new GPIs, in contrast to L&A where a GTC boosts alternatives for the PC makers' GPIs. This is partially due to the fact that Android is free and open-source, unlike Intel microprocessors, hence both newer and older Android versions do not provide performance and price trade-offs [29]. Instead, the most recent version is vastly superior (and cost-free), making Android's GTC the crucial window of opportunity for OEMs to engage in generative appropriability. Android unilaterally decides when to release a new GTC, which appears to OEMs as though a GPI has been imposed on them. In order to achieve a quicker time to market (TTM) while introducing new phone models, OEMs must concentrate on how quickly and effectively they can leverage the free open-source resources. Importantly, the time, style, and organization of learning for complementors are significantly altered by Google's institutionalization of each new GTC (on the Android Open-Source Project, or AOSP), as well as by the rigorous timing of OS releases. OEMs must develop sequential ambidexterity, which we define as ambidexterity that is temporally rigid. Routinizing the learning process and perfecting the timing of exploitative and exploratory learning are crucial for OEMs using the Android platform [58].

2.3 Sequential Ambidexterity for GPC: Exploitative and Exploratory Learning

In order to achieve a quicker TTM while introducing new phone models, OEMs must concentrate on how quickly and effectively they can leverage the free open-source resources [29]. Importantly, the time, style, and organization of learning for complementors are significantly altered by Google's institutionalization of each new GTC (on the Android Open-Source Project, or AOSP), as well as by the rigorous timing of OS releases. OEMs must develop sequential ambidexterity, which we define as ambidexterity that is temporally

rigid. Routinizing the learning process and perfecting the timing of exploitative and exploratory learning are crucial for OEMs using the Android platform. According to Wadhwa and Kotha (2006), local and distant search are related to exploratory and exploitative learning, respectively. For the purpose of this study, Android's GTC calls for complementors to order and time the two learning modes in order to seamlessly integrate the technology of the previous and current generations [29].

The ambidexterity of complementors depends on the properties of GTCs. In the platform ecosystem, complementors are required to understand how platforms' technologies are interdependent with one another and to invest in platform-specific resources for their product development [1, 10, 30]. Interdependencies provide more difficult problems during GTCs, particularly if they happen quickly and continuously. Even with modularity, GTCs are not formulaic changes from one technology to another, and they frequently necessitate adjustments to the product systems' architectural layout [4, 60]. Kapoor and Agarwal draw attention to the numerous technological dependencies that mobile app developers encounter when switching between Android versions, such as the need to adapt their apps for various screen sizes and the vastly varying screen and camera resolutions used by OEMs [30].

The GTC provides complementors with a window of opportunity to understand technology interdependencies and improve their GPIs. For instance, OEMs will first start with exploitative learning by utilizing their accumulating knowledge from the old-generation technology to improve performance and correct problems before a new version of Android OS is officially published. Backward compatibility over the GTC can also be secured by OEMs' nefarious understanding of the older generation of technology [34, 36]. The technical interdependence rises as a new generation of technology is launched, necessitating exploratory learning on the part of complementors. This is due to OEMs' desire to experiment with and improve hardware-OS interdependence in order to gain a competitive advantage. In order to improve their TTM and deal with the technological interdependence of GTC, complementors must think about how to perform sequential ambidexterity.

3 Hypothesis

A platform provider's GTC is a technological advancement that poses problems for complementors' GPI and makes it more difficult to achieve faster TTM. Complementors must expend more effort understanding the GTC of a platform, which creates extra uncertainties and complications for them. Complementors can start learning as soon as the latest generation technology is accessible, which will hasten the development of their new products. According to earlier research, a platform's accessibility and openness can improve complementors' abilities to produce new products. As an illustration, Boudreau (2010) contends that PDA operating system (OS) providers give independent hardware developer firms increased levels of access, which causes a fivefold acceleration of new portable device development [6].

The Android ecosystem is more open than the iOS platform, enabling content producers to operate more

technically effectively [37]. The openness of Android affects application developers' plans to continue participate in the Android ecosystem [12]. As a result, a platform's openness and accessibility have an impact on complementors' success in GPIs; complementors must coordinate their learning with the platform's release schedule.

Typically, the platform provider starts the process by releasing a "developer version" of software, giving complementors the chance to learn about the technological characteristics of a new generation of platform technology. The complementors begin integrating this developer-version software with their hardware architecture, testing the quality, fixing issues, and improving performance. Complementors must think about how to more effectively combine hardware and software because the developer version may still have issues and some features' performance has to be improved. To put it another way, when a novel solution is used, unexpected side effects may have an impact on the technological dependency between software and hardware. Due to the ongoing definition of technical dependency, complementors can have trouble determining the exact origin of a defect. In these situations, complementors not only work closely with inside divisions to seek solutions, but they also do so with the platform provider, outside vendors, and even rival companies. In other words, complementors may investigate solutions from the platform provider's upgrading developer version [29]. As an alternative, complementors can investigate solutions using various platform functionalities for distant search. This is especially true given that developers can find solutions to issues from various knowledge domains. In conclusion, it is essential for complementors to conduct remote search through exploratory learning to uncover solutions and boost the effectiveness of problem-solving activities when technological interdependence is strong and changing during the platform's developer version software [29].

Hypothesis 1:

In the period of a GTC, the higher the degree of exploratory learning of the complementors on the new-generation platform (or the developer version), the greater it contributes to the TTM of the complementors' GPI.

The "maintenance version" software, which attempts to fix small issues and boost the performance of the old-generation platform, is instead released by a platform provider during a GTC before the development version software. The older platform has established design guidelines, conventions, and interfaces that enable complementors to make advantage of the body of existing expertise to enhance performance and address hardware-software interoperability issues. In other words, as technological dependency becomes more solid, complementors will be able to solve issues by applying their existing skills and knowledge to more commonplace tasks. For instance, when complementors solve a problem, they might be more adept at predicting the adverse consequences and choose the best testing method to save testing time and improve problem-solving effectiveness. Additionally, complementors can conduct local search from their current solutions to enhance product performance because technical interdependence is constant. Complementors learn how to boost problem-solving efficiency when they use this exploitative learning on the old-generation platform. In other words, complementors are able to amass more information and experience the more exploitative learning they embrace.

In some circumstances, future technological performance improvements may make older technology obsolete [10]. When creating a new generation of platforms, the platform provider will also take backward compatibility with the older platforms into account [34, 36]. When this occurs, complementors may be able to integrate both old and new features of the new generation platform with the information they have gained through exploitative learning on the old generation platform. According to complementors' experience, backward compatibility of some specific capabilities, including Bluetooth and WiFi, occasionally necessitates certification testing for complementors and can be implemented via add-on testing suits. Moreover, complementors can use user feedback to prioritize the interaction between existing and new features while implementing existing functions in a new-generation platform. To prepare or allocate development resources for the adoption of the new generation platform in advance, the platform provider may add some new features to the maintenance version software, and complementors may further enhance the effects of exploitative learning on the old generation platform.

Hypothesis 2:

In the period of a GTC, the higher the degree of exploitative learning of the complementors on the old-generation platform (or the maintenance version), the greater it contributes to the TTM of the complementors' GPI.

According to several studies, firms engaging in exploitative and exploratory learning have an interaction effect on innovative activities[5, 9, 22, 32, 56, 25, 59]. In other words, exploitative learning makes it easier for a company to make future investments in exploratory learning. In order to help businesses increase the effectiveness of discovering new knowledge in new product development, exploitative learning, for instance, improves complementors' absorptive capacity[13]. Additionally, exploitative learning supports knowledge exploration throughout the development of new products and helps businesses recognize and use new external knowledge[13, 61].

In other words, in sequential ambidexterity, exploitative and exploratory learning are interacting and complimentary. Complementors who engage in exploitative learning on the old-generation platform improve their exploratory learning on the new-generation platform in a sequential fashion. Product managers can provide postmortem reports to formalize the troubleshooting, bug fixes, testing methods, and quality controls they have utilized after choosing the old-generation platform for their GPI. To lessen uncertainty for future product development, these postmortem reports can be used as reference materials and distributed to other development teams[16, 35, 52]. Problems in new projects can also be reduced through other forms of knowledge exchange, like project-to-project knowledge transfers[52]. As a result, when complementors explore their new GPI, there will be less issues and more opportunities for exploratory learning on the new-generation platform the more exploitative learning that engages the old-generation platform.

Hypothesis 3:

In the period of a GTC, the higher the degree of exploitative learning of the complementors on the old-generation platform (or maintenance version), the more efficient it is to improve the effects of exploratory learning on the new-generation platform (or developer version) and contribute to the TTM of complementors' GPI.

4 Methodology

4.1 Data Collection and Processing

The objects of analysis in this study are Nokia, Sony Ericsson, Motorola, Samsung, Xiaomi, ZTE, HTC, ASUS, and Acer - nine smartphone OEMs. In 2019, these eleven OEMs possessed 77% of the global Android smartphone market share[1]. Android is an open-source operating system (OS), and OEMs can participate in the Android Open Source Project (AOSP) and use Android OS to develop Android smartphones. Google releases a new version about once per year, which constitutes the GTCs in our study. As shown in Figure 1, we observe how the OEMs conduct exploitative and exploratory learning during nine GTCs of the Android OS, from June 6, 2010, to August 15, 2022.

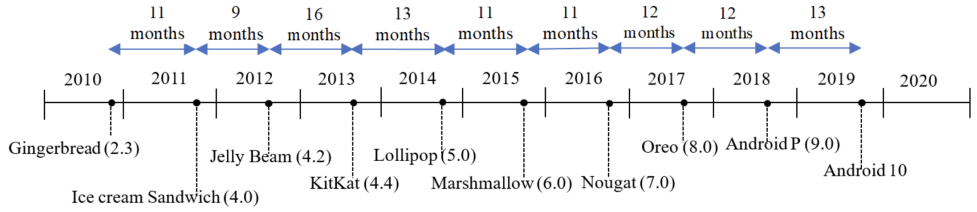


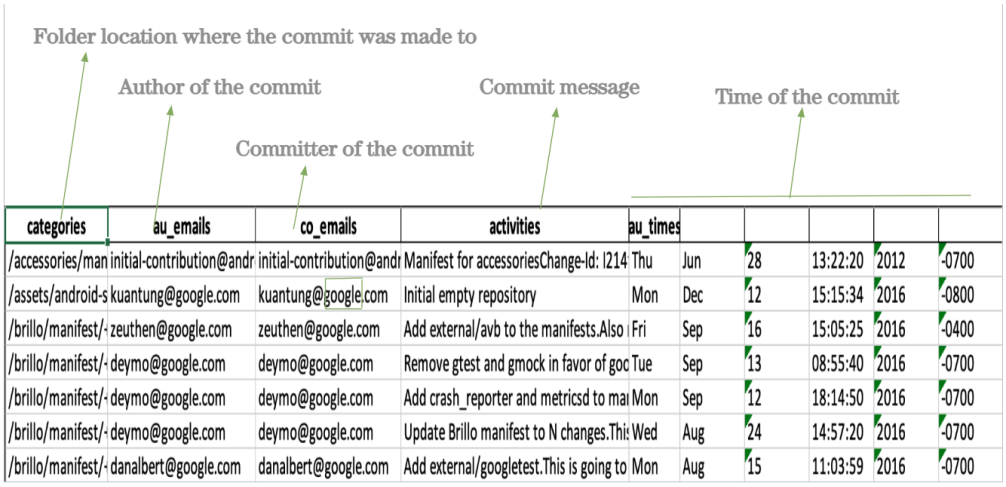
Figure 1: Example of Android OS Release Dates

Each GTC consists of a "maintenance version" of the old Android OS, during which exploitative learning occurs, and the new "developer version" where exploratory learning occurs to experiment on the upcoming OS. In Figure 1, the first GTC occurred during the transition from Gingerbread (Android 2.3) to Ice Cream Sandwich (Android 4.0), and the second GTC occurred from Ice Cream Sandwich (Android 4.0) to Jelly Beam (Android 4.2). On average, each GTC takes 12 months. We refer to the Android developer website[2] and define the former half period of GTC as the old-generation platform (maintenance version) and the latter half period of GTC as the new-generation platform (developer version). The center point of a GTC exists in between the maintenance and developer versions [29].

Companies that give their software code to the open-source software community can pick up technological knowledge from other companies [27, 33]. This is what Nagle (2018) refers to as "learning by contributing," meaning that businesses pick up important knowledge through software contributions, which helps them produce more software more quickly and add more value to it. Software contributions, such as enhancing the effectiveness of software debugging and swiftly integrating new features in their products, are also viewed by Sowe and Stamelos (2008) as a kind of company learning [41, 50]. According to these research, we use contributions—also known as "commits"—as a measure of OEM learning throughout the GTC. We extracted and downloaded a comprehensive dataset of 15,083,526 commits from the Android

[1]https://gs.statcounter.com/vendor-market-share/mobile/worldwide/2019
[2]https://developer.android.com/about/versions

source code website[3] using Python in July 2022 by working alongside Professor Dallas and his research colleagues in Taiwan [29]. Figure 2 shows a snapshot of the commits dataset.



categories	au_emails	co_emails	activities	au_times					
/accessories/man	initial-contribution@andr	initial-contribution@andr	Manifest for accessoriesChange-Id: I214	Thu	Jun	28	13:22:20	2012	-0700
/assets/android-s	kuantung@google.com	kuantung@google.com	Initial empty repository	Mon	Dec	12	15:15:34	2016	-0800
/brillo/manifest/-	zeuthen@google.com	zeuthen@google.com	Add external/avb to the manifests.Also	Fri	Sep	16	15:05:25	2016	-0400
/brillo/manifest/-	deymo@google.com	deymo@google.com	Remove gtest and gmock in favor of goc	Tue	Sep	13	08:55:40	2016	-0700
/brillo/manifest/-	deymo@google.com	deymo@google.com	Add crash_reporter and metricsd to mai	Mon	Sep	12	18:14:50	2016	-0700
/brillo/manifest/-	deymo@google.com	deymo@google.com	Update Brillo manifest to N changes.This	Wed	Aug	24	14:57:20	2016	-0700
/brillo/manifest/-	danalbert@google.com	danalbert@google.com	Add external/googletest.This is going to	Mon	Aug	15	11:03:59	2016	-0700

Figure 2: Snapshot of the Commits dataset

There are 9,436,657 commits (62.5%) where the committer and author companies do not match. In the context of version control systems like Git, including the Android Open Source Project (AOSP), there is a distinction between the author and the committer of a commit. The author of a commit refers to the person who originally created the content or made the changes. The author is typically the developer who wrote the code or made the modifications. The committer of a commit refers to the person who applies the changes to the codebase. The committer is responsible for committing the changes into the version control system. The distinction between the author and the committer is important for giving credit to the original contributor of the code and also for tracking the person responsible for applying the changes into the codebase. It allows for proper attribution and helps maintain a clear history of the project's development.

In the context of the Android Open Source Project (AOSP), the distinction between the author and committer of a commit can be observed due to the collaborative nature of open-source development. In open-source projects, multiple individuals or contributors actively participate in the development and enhancement of the software. The author of a commit refers to the person who initially creates or authors the code changes, representing the original work or idea. On the other hand, the committer is responsible for applying the changes to the project's codebase, ensuring that they meet the project's standards and guidelines before merging them. The committer may perform code reviews, testing, and integration of the code changes.

Various factors contribute to the author and committer of a commit being different in AOSP. Firstly, the code review process plays a vital role in open-source projects. Code changes often go through rigorous reviews, where reviewers suggest modifications or improvements. The original author may not always

[3]https://android.googlesource.com

be the one who incorporates these suggestions into the final commit, leading to a distinction between the author and committer. Secondly, collaborative workflows within open-source projects involve multiple contributors working simultaneously on different aspects of the codebase. These contributors may have diverse roles, responsibilities, and areas of expertise. Consequently, the person who initially authored the code changes may not be the same person who commits them to the repository.

Moreover, in larger open-source projects like AOSP, designated maintainers or core developers are responsible for merging code changes. These maintainers review and modify the code changes before integrating them into the main codebase. As a result, the committer, who is often a maintainer or core developer, may differ from the original author. This differentiation between authorship and commit can be observed as a testament to the collaborative and distributed nature of open-source development. Hence, to capture the authorship aspect and understand the contributions of individual authors, we decided to use the author email instead of the committer email in our analysis.

Developers upload software program files known as commits, which also serve as a sign of updated software code [28, 57]. Commits are seen as a sign of open source software's technical advancements, and as a measure of a project's success in terms of software development and knowledge generation. The email addresses of OEM commits allowed us to locate them [23, 47, 48]. As an example, the commit record associated with Nokia can be identified by the email address format xxxxx@nokia.com, while for Samsung, it is xxxxx@samsung.com. To determine the OEM names, we extract the portion between the "@" symbol and the "." symbol in the email addresses. Figure 3 shows the distribution of the commits made by the nine OEMs in comparison to each other.

Android OEM	# of Commits	% share of commits
samsung	117232	69.82%
nokia	38687	23.04%
motorola	4868	2.90%
xiaomi	1925	1.15%
zte	1725	1.03%
htc	1500	0.89%
sonyericsson	839	0.50%
acer	643	0.38%
asus	497	0.30%

Figure 3: OEMs Share of Commits to AOSP

4.2 Model and Variables

This study examines the manner in which OEMs launched their smartphones during Android OS GTCs. In each GTC, we identify the developer version of the new Android OS as the period of exploratory learning for OEMs while the maintenance version of the old Android OS is referred to as the period of exploitative learning for OEMs. The window of opportunity for OEMs to undertake exploitation on the t-1st version of Android OS opens upon its release, and exploratory learning starts with the release of the developer version for integrating the tst version of Android OS. Exploratory and exploitative learning are hence the time-dependent covariate with OEMs' TTM.

This study is a follow-up to Puranam et al. (2006), who examined the impact of exploitative and exploratory learning at various phases on a firm's new product launches using the Cox proportional hazards model (Cox, 1972). The hazard (instantaneous likelihood) in this mode represents OEM smartphone launches following the introduction of a given Android OS, and the hazard rate is a time-dependent covariate with effects from both exploitative and exploratory learning. We created a longitudinal dataset of the number and timing of product releases by OEMs. The window of opportunity for learning is the same for every OEM because it is impossible for an OEM to release a smartphone with the new version of Android OS before it has finished the GTC [15, 43]. For simplicity, we mark a GTC's start date as the beginning of the smartphone launch observation period. The period of smartphone launch observations is terminated when the next new version of Android OS is released.

Our dependent variable, TTM, is characterized as the rate and output of product development [2, 7, 19]. It is connected to how well processes work when resolving conflicts and transferring data. The more possibility an OEM has to profit from the product market, the closer the time between a GTC start date and the introduction of a focus OEM's smartphone for the same GTC period. Throughout the course of the nine GTCs, we represent exploitative and exploratory learning using the Android commits data for the old and new generations of Android, respectively. According to Grewal et al. (2006) and MacCormack et al. (2012), OEM commits to different software folders within Android OS represents different types of learning [23, 39]. Android OS is organized within hundreds of nested folders as show in Figure 4 below. Folders in Android OS comprise the technological dependencies that have an impact on how OEMs learn and perform during product development. They each represent a particular capability.

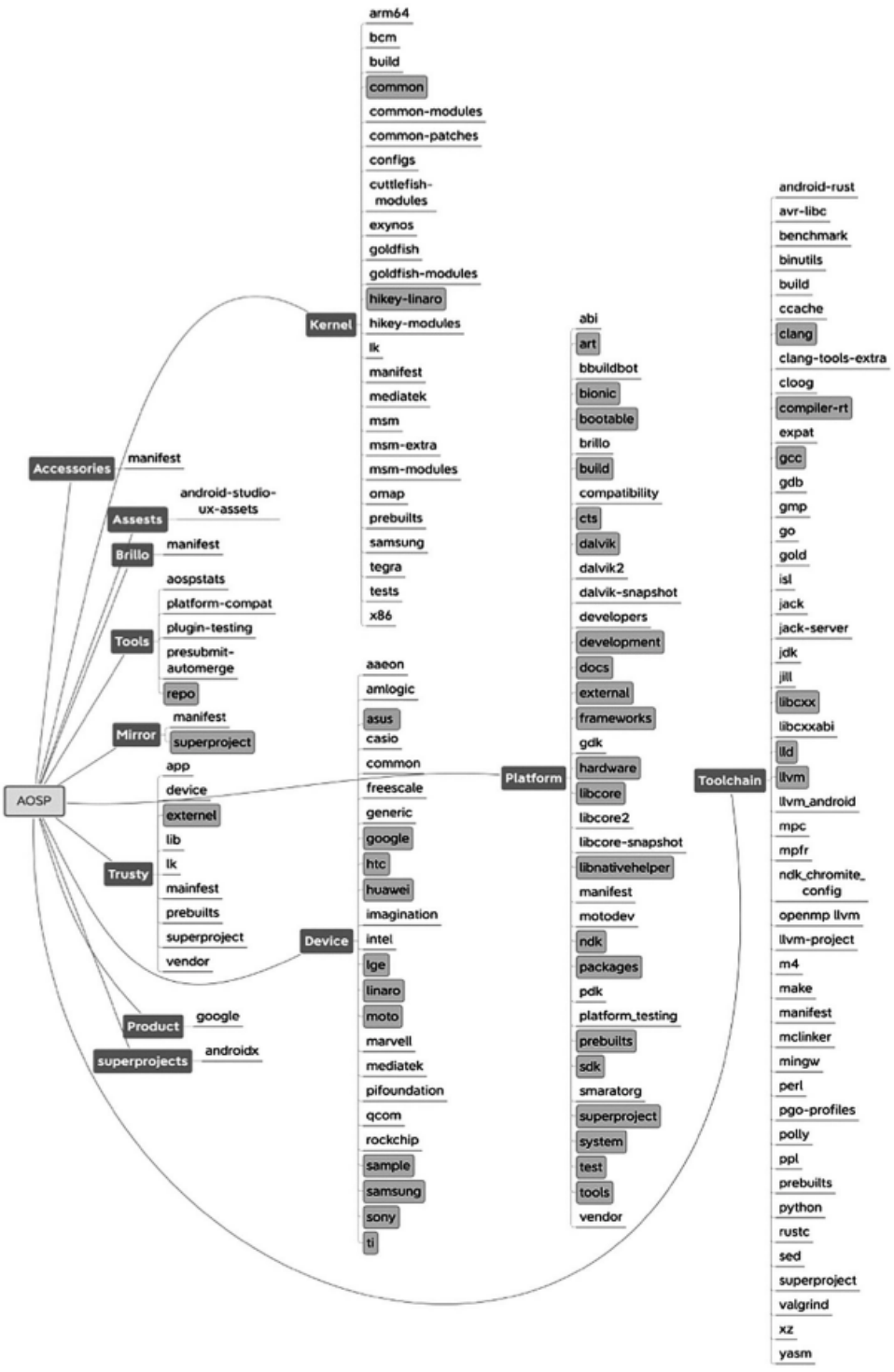


Figure 4: AOSP Folder Structure

Platform folder is the central location where OEMs devote their resources to integrate the Android OS with their hardware design [29]. The 37 second-tier subfolders in the Platform folder all function in conjunction with one another. For instance, the software in the Framework folder (a subdirectory of the Platform folder) supports several Android mobile apps in addition to the user interface, telephony, resources, locations, content providers, and package managers. Contrarily, software in the External folder interacts with browser, database, font, and media formats, whereas hardware in the Hardware folder drives components like the camera, sensors, and display, and integration testing is implemented in the CTS folder to ensure proper software-hardware interaction. These and other Platform folder-related sub-folders have technology dependencies that OEMs must address. In total, we found 12,004,739 commits in the Platform folders in our data.

We utilized the date of each commit to differentiate all 118,717 commits into seventeen GTCs. Moreover, within each GTC, we further divided commits between the maintenance version of the old-generation platform (the first half of the GTC), and the developer version of the new-generation platform (the second half of the GTC).

We compute the Simpson Index-based Diversity (Page, 2010) of OEM commits dispersed throughout all subfolders within the Platform folder during the developer version period to assess *exploratory learning*. As can be seen in formula (1), M is the subfolder number beneath the Platform folder where an OEM has made a promise, and ni is the degree of an OEM's commitment in the particular subfolder i [42]. N represents the total number of commits made by an OEM during the developer version period to the Platform folder. The Simpson index is constructed such that we capture the notion of diversity or *exploration* of commits; hence, exploratory learning.

$$\text{exploratory learning} = 1 - \sum_{i=1}^{M} \frac{n_i(n_i - 1)}{N(N-1)} \tag{1}$$

Lastly, the logarithm of the number of commits in the Hardware folder, a sub-folder of the Platform folder in the original Android OS, is used to measure *exploitative learning*. In our discussions with Taiwanese Google software developers, we discovered that the Hardware folder is where OEMs concentrate their efforts to improve the performance of the smartphone on the older platform. There are 82,610 commits from our eleven OEMs in the Hardware folder, with almost 70% of those being in the Platform folder. OEMs use exploitative learning, as opposed to exploratory learning, to enhance hardware performance, address issues, and update the outdated Android OS (maintenance version).

Since different technical aspects of CPUs may need varying levels of learning that have an impact on TTM, we account for difference in CPUs of smartphones in this study. Additionally, CPU vendors might serve as platform suppliers that have an impact on the design of smartphones [29]. In order to account for variances between Qualcomm, MediaTek, Samsung, and Texas Instruments, which may affect OEMs' TTM, the dummy variables of CPUs were created.

5 Results

We used 1,425 Android smartphones from nine OEMs as our sample. Over the course of approximately eight GTCs, one OEM released an average of about 120 Android smartphones. At the extremes, Samsung aggressively released approximately 300 Android smartphones over the course of nine GTCs, while Nokia produced only about 30 Android devices while contributing to AOSP over three GTCs.

Table 1 displays the findings from a Cox regression in which the hazard of TTM, or the speed and output of nine Android smartphone OEMs, served as the dependent variable. The coefficients are β, which can be exponentiated as $e^{-\beta}$ to represent the hazard ratios as the shortened TTM. The hazard of product introduction depends on the occurrence and the timing of product introductions [43]. The greater the hazard ratio, the greater the TTM advantage OEMs have after the GTC start date or Android release date.

We examined all three hypotheses using the Cox proportional hazards model. We also include dummy variables to control the effect on TTM caused by major CPUs: Samsung, Qualcomm, MediaTek, and Texas Instruments. In Model 1, we only included the dummy and control variables. All the CPUs have significant negative impacts on the TTM, i.e., decreasing the TTM. When Samsung increased one unit of its CPU_Samsung, to develop Android smartphones, it can accelerate the TTM by about 1.34 times.

Table 1: Cox Regression for the Hazard of TTM.

Variables	Model 1	Model 2	Model 3	Model 4
Exploratory learning		$-0.289(0.468)$***		$-0.103(0.734)$
Exploitative learning			$-0.051(0.117)$*	$0.496(0.478)$***
Exploitative*Exploratory learning				$-0.585(0.316)$***
CPU_Samsung	$-0.295(0.102)$**	$-0.291(0.118)$**	$-0.255(0.968)$**	$-0.289(0.341)$**
CPU_Qualcomm	$-0.272(0.826)$***	$-0.299(0.252)$***	$-0.285(0.239)$**	$-0.286(0.228)$**
CPU_MediaTek	$-0.401(0.170)$***	$-0.489(0.839)$***	$-0.396(0.138)$*	$-0.361(0.249)$*
CPU_TI	$0.639(0.252)$***	$1.99(0.896)$***	$0.686(0.797)$**	$0.561(0.170)$**

* $p < .05$; ** $p < .01$; *** $p < .001$.

Models 2, 3, and 4 are related to our three hypothesis. Firstly, we hypothesized that a higher degree of exploratory learning among OEMS on developer version would decrease the TTM. Again, exploratory learning represents the diversity or exploratory nature of the commits made by an OEM during the developer version of an Android version, ranging from 0 to 1.

Model 2 shows the coefficient for exploratory learning is -0.289, suggesting that higher levels of exploratory learning are associated with a lower hazard or shorter time to market. The coefficient is statistically significant, indicating that there is no strong evidence to support a significant association between

exploratory learning and time to market. The hazard ratio (exp(-coef)) for exploratory learning is 1.34, indicating that a one-unit increase in exploratory learning is associated with a decrease in the hazard of releasing a phone after a new Android version by a factor of 1.34.

Secondly, we hypothesized that a higher degree of exploitative learning among OEMs on the maintenance version would make the TTM faster.

Model 3 shows that coefficient for exploitative learning is -0.051. Since the coefficient is negative, it suggests that higher levels of exploitative learning are associated with a lower hazard or shorter time to market. The exponentiated coefficient (exp(-coef)) is 1.052, indicating that for a one-unit increase in exploitative learning, the hazard of TTM decreases by a factor of 1.052. The p-value of 0.0117 indicates that the relationship between exploitative learning and time to market is statistically significant for $p< 0.01$.

Lastly, we hypothesized that a higher degree of exploitative learning on the maintenance version would enhance the effects of exploratory learning on the developer version and further contribute to the TTM of an OEM's GPI or phone model.

We measured this effect by using the interaction term of exploitative and exploratory Learning. The results in Model 4 shows that the coefficient for the interaction term between exploitative learning and exploratory learning is -0.585, and it is statistically significant ($p < 0.001$). The negative coefficient suggests that the combined effect of exploitative learning and exploratory learning quickens TTM. The hazard ratio is 1.80, indicating that as the interaction between exploitative and exploratory learning increases by one unit, the hazard of releasing a phone after a new Android version decreases by by a factor of 1.80.

All three hypothesis are supported by the results, controlling for the CPU. However, several other factors like an OEM's financial health, resources, nationality, etc. are not taken into account for the models above.

6 Discussion and Conclusion

According to this study, there are various types of generational technology transitions. It concentrates on GTCs and GPIs, both of which, according to Dosi's (1982) framework, remain within a single technical trajectory and paradigm yet involve continual and extremely quick technological advancements. Despite these similarities, we conclude that GTCs and GPIs should be distinguished even though the literature frequently conflates them. A GTC is more comprehensive since it establishes technical regimes and drives them forward, frequently through isolated individuals or occasions that have an impact on the technological environment. Platform leaders who advance technological generations in microprocessors, operating systems, or hegemonic equipment companies are examples of GTCs. A GPI is more specialized and limited to the introduction of new items that have no bearing on the technical environment that the researcher is researching. This study used actual evidence to demonstrate how each new GTC Android introduced imposed timing and sequencing requirements on how complementors organized their learning and when they released GPIs. As a result, OEMs must embrace sequential ambidexterity when using Android's open-source learning engine. In other words, the yearly introduction of a new OS for Android compelled OEMs to undergo new GPIs and imposed a strict order of exploratory learning, followed by exploitative learning. Our empirical work offers several insights: exploitative learning towards maintenance, exploratory learning towards the developer version, as well as the interaction of exploitative and exploratory learning, all improve TTM for OEMs phone models.

These findings could have significant ramifications. First, variations in GTCs have an effect on the decisions and tactics used by businesses involved in GPIs. In the instance of Android, the platform enforced rigidity and order on OEMs, limiting their ability to make strategic decisions on the time of GPI, the method and timing of learning and innovation, and perhaps other areas that may merit further study. In contrast, GTCs performed the exact opposite in other research (like L&A) by boosting the diversity of intermediary products (like new-generation microprocessors), which leads to a more complicated strategic environment and the emergence of new market niches and product combinations. While Kapoor and Agarwal (2017) do not investigate the characteristics of their GTCs (generations of video game consoles), it is possible that the relative rigidity of those platforms played a role in their discovery that GTCs can negatively impact the performance of the GPIs of top complementors (video game developers). Therefore, rather of just determining whether a GTC exists or not, researchers would do better to look into its nature [30].

'Sequential' ambidexterity is used in this work to explain the sequential nature of exploitative and exploratory learning during GTCs. To put it another way, organizational ambidexterity differs from architectural ambidexterity or contextual ambidexterity in that it involves more than just engaging in exploitative and exploratory learning at the same time [3]. Instead, time-pacing demands imposed by platform providers drive OEMs to match their innovation clocks and GPI releases to the GTC in fast-paced industries with well-structured platforms. When the window of opportunity opens, businesses must carefully

pivot between exploratory and exploitative learning.

However, it is important to acknowledge that our study has certain limitations and assumptions. One notable limitation is that we did not consider the financial health or competitive advantage of each OEM in our analysis. The financial health of an OEM can play a crucial role in its ability to invest in research and development, allocate resources efficiently, and ultimately affect its time to market. Companies with stronger financial positions may have more resources at their disposal to invest in exploratory and exploitative learning, potentially leading to shorter time to market. On the other hand, OEMs facing financial constraints may have limited capabilities to engage in extensive learning activities, which could result in longer time to market. Similarly, competitive advantage can significantly impact an OEM's time to market. OEMs with a strong competitive position, such as established market leaders or companies with unique technological capabilities, may have a better understanding of customer needs, access to critical resources, and established relationships with suppliers and partners. These advantages can enable them to expedite product development processes and reduce time to market compared to their competitors.

Future research could delve into the role of financial health and competitive advantage to gain a more comprehensive understanding of the dynamics between learning strategies and time to market in the context of OEMs. It is important to note that while our study focused on the specific relationship between exploratory and exploitative learning and time to market, these additional factors could provide valuable insights and further enhance our understanding of the complexities involved in new product development and market entry for OEMs. In order to demonstrate learning through contributing, this study used the quantity and distribution of commits across Android OS folders [41]. However, there are several research opportunities in the large and complicated open-source Android ecosystem. For instance, by looking at firm commits within the same folders, one can pinpoint the patterns of interaction and collaborative problem-solving activities amongst firms [23]. Researchers might also look into the technological dependencies between files and show how those patterns change over time. In order to accomplish this, researchers may need to collaborate with software engineers who can offer technical details on these more intricate interdependencies. Researchers can also detail the many activities that OEMs engage in inside the Android open-source community. Our understanding of how OEMs carry out exploitative and exploratory learning should be improved as a result. Finally, it is possible that this study overlooked hardware design and concentrated too much on software development. Future studies could address how sequential ambidexterity is used by integrating hardware and software, testing, quality assurance, and other procedures using qualitative approaches. The general insights that GTCs and GPIs are different and that platform-specific learning is not only essential to modularity but can also take on very specific modalities, like sequential ambidexerity, are all ideas that can easily migrate to other platform environments even though they are all narrowly focused on Android OS.

References

[1] Ron Adner and Rahul Kapoor. "Value creation in innovation ecosystems: How the structure of technological interdependence affects firm performance in new technology generations". In: *Strategic Management Journal* 31.3 (2010), pp. 306–333.

[2] Gautam Ahuja, Curba Morris Lampert, and Elena Novelli. "The second face of appropriability: Generative appropriability and its determinants". In: *Academy of Management Review* 38.2 (2013), pp. 248–269.

[3] Ankur Anand et al. "An interactive approach to determine optimal launch time of successive generational product". In: *International Journal of Technology Marketing* 9.4 (2014), pp. 392–407.

[4] Shahzad Ansari and Raghu Garud. "Inter-generational transitions in sociotechnical systems: The case of mobile communications". In: *Research Policy* 38.2 (2009), pp. 382–392.

[5] Seigyoung Auh and Bulent Menguc. "Balancing exploration and exploitation: The moderating role of competitive intensity". In: *Journal of Business Research* 58.12 (2005), pp. 1652–1661.

[6] Kevin J. Boudreau. "Open platform strategies and innovation: Granting access vs. devolving control". In: *Management Science* 56.10 (2010), pp. 1849–1872.

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

[8] Shona L Brown et al. *Competing on the edge: Strategy as structured chaos*. Harvard Business Press, 1998.

[9] Qing Cao, Eric Gedajlovic, and Hongping Zhang. "Unpacking organizational ambidexterity: Dimensions, contingencies, and synergistic effects". In: *Organization Science* 20.4 (2009), pp. 781–796.

[10] Carmelo Cennamo. "Building the value of next-generation platforms: the paradox of diminishing returns". In: *Journal of Management* 44.8 (2018), pp. 3038–3069.

[11] L. Chen et al. "Growing pains: The effect of generational product innovation on mobile games performance". In: *Strategic Management Journal* 43.4 (2021), pp. 792–821.

[12] Gyewan Choi, Changi Nam, and Seungjae Kim. "The impacts of technology platform openness on application developers' intention to continuously use a platform: From an ecosystem perspective". In: *Telecommunications Policy* 43.2 (2019), pp. 140–153.

[13] Wesley M Cohen and Daniel A Levinthal. "Absorptive capacity: A new perspective on learning and innovation". In: *Administrative Science Quarterly* 15.1 (1990), pp. 128–152.

[14] Robert G. Cooper and Elko J. Kleinschmidt. "Reducing Time to Market: A Strategic Imperative". In: *The Journal of Product Innovation Management* 12.1 (1995), pp. 3–14.

[15] D. R. Cox. "Regression models and life-tables". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 34.2 (1972), pp. 187–202.

[16] Michael Cusumano and Richard Selby. *Microsoft secrets*. The Free Press, 1995.

[17] RA D'Aveni. *Hypercompetition–Managing the Dynamics of Strategic Maneuvering, New York/Toronto*. 1994.

[18] Kathleen M Eisenhardt. "Making fast strategic decisions in high-velocity environments". In: *Academy of Management journal* 32.3 (1989), pp. 543–576.

[19] Kathleen M Eisenhardt and Behnam N Tabrizi. "Accelerating adaptive processes: Product innovation in the global computer industry". In: *Administrative science quarterly* (1995), pp. 84–110.

[20] D Charles Galunic and Kathleen M Eisenhardt. "The evolution of intracorporate domains: Divisional charter losses in high-technology, multidivisional corporations". In: *Organization Science* 7.3 (1996), pp. 255–282.

[21] Connie JG Gersick. "Pacing strategic change: The case of a new venture". In: *Academy of management journal* 37.1 (1994), pp. 9–45.

[22] Cristina Gibson and Julian Birkinshaw. "The antecedents, consequences, and mediating role of organizational ambidexterity". In: *Academy of Management Journal* 47.2 (2004), pp. 209–226.

[23] Rajdeep Grewal, Gary L Lilien, and Girish Mallapragada. "Location, location, location: How network embeddedness affects project success in open source systems". In: *Management science* 52.7 (2006), pp. 1043–1056.

[24] Abbie Griffin and Alan L. Page. "Time to Market, Sequential Investment, and Pricing Policy for New Industrial Products". In: *The Journal of Marketing Research* 30.2 (1993), pp. 220–233.

[25] Zhiwen L. He and Poh Kam Wong. "Exploration vs. exploitation: An empirical test of the ambidexterity hypothesis". In: *Organization Science* 15.4 (2004), pp. 481–494.

[26] Constance E. Helfat and Margaret A. Peteraf. "Understanding dynamic capabilities: Progress along a developmental path". In: *Strategic Organization* 7.1 (2009), pp. 91–102.

[27] Pekka Himanen, Linus Torvalds, and Manuel Castells. *The Hacker Ethic*. Random House, 2001.

[28] Eric von Hippel and Georg von Krogh. "Open source software and the "private-collective" innovation model: Issues for organization science". In: *Organization science* 14.2 (2003), pp. 209–223.

[29] Zih-Rong Wang Jing-Ming Shiu Mark P. Dallas. "Varieties of Generational Technology Transitions: The Android Platform, Complementors and Sequential Ambidexterity in Organizational Learning". Unpublished paper. 2022.

[30] Rahul Kapoor and Sumit Agarwal. "Sustaining superior performance in business ecosystems: Evidence from application software developers in the iOS and Android smartphone ecosystems". In: *Organization Science* 28.3 (2017), pp. 531–551.

[31] Praveen Kapur et al. “Multi-generational innovation diffusion modelling: A two dimensional approach”. In: *International Journal of Applied Management Science* 7.1 (2015), pp. 1–18.

[32] Riitta Katila and Gautam Ahuja. “Something old, something new: A longitudinal study of search behavior and new product introduction”. In: *Academy of management journal* 45.6 (2002), pp. 1183–1194.

[33] David Kohanski. *Moths in the Machine*. St. Martin’s Griffin, 1998.

[34] Tobias Kretschmer and Jörg Claussen. “Generational transitions in platform markets—The role of backward compatibility”. In: *Strategy Science* 1.2 (2016), pp. 90–104.

[35] Sameer Kumar and David Terpstra. “The post mortem of a complex product development—lessons learned”. In: *Technovation* 24.10 (2004), pp. 805–818.

[36] Michael W Lawless and Philip C Anderson. “Generational technological change: Effects of innovation and local rivalry on performance”. In: *Academy of Management Journal* 39.5 (1996), pp. 1185–1217.

[37] Changhui Lee, Dong-Joon Lee, and Junseok Hwang. “Platform openness and the productivity of content providers: A meta-frontier analysis”. In: *Telecommunications Policy* 39.7 (2015), pp. 553–562.

[38] Ilan Lobel et al. “Optimizing product launches in the presence of strategic consumers”. In: *Management Science* 62.6 (2016), pp. 1778–1799.

[39] Alan MacCormack, Carliss Y. Baldwin, and John Rusnak. “Exploring the duality between product and organizational architectures: A test of the ”mirroring” hypothesis”. In: *Research Policy* 41.8 (2012), pp. 1309–1324.

[40] Alan MacCormack, Roberto Verganti, and Marco Iansiti. “Developing products on ”Internet time”: The anatomy of a flexible development process”. In: *Management Science* 47.1 (2001), pp. 133–150.

[41] Frank Nagle. “Learning by contributing: Gaining competitive advantage through contribution to crowdsourced public goods”. In: *Organization Science* 29.4 (2018), pp. 569–587.

[42] Scott E. Page. *Diversity and Complexity*. Princeton University Press, 2010.

[43] Phanish Puranam, Harbir Singh, and Maurizio Zollo. “Organizing for innovation: Managing the coordination-autonomy dilemma in technology acquisitions”. In: *Academy of Management Journal* 49.2 (2006), pp. 263–280.

[44] Rajagopal. “Innovation and Technology Generations”. In: *Transgenerational Marketing*. Cham: Palgrave Macmillan, 2020. DOI: 10.1007/978-3-030-33926-5_3. URL: https://doi.org/10.1007/978-3-030-33926-5_3.

[45] Lori Rosenkopf and Michael L Tushman. “The coevolution of community networks and technology: Lessons from the flight simulation industry”. In: *Industrial and corporate change* 7.2 (1998), pp. 311–346.

[46] Xiaomeng Shi, Kiran Fernandes, and Pattarawan Chumnumpan. “Diffusion of multi-generational high-technology products”. In: *Technovation* 34.3 (2014), pp. 162–176.

[47] Param Vir Singh. "The small-world effect: The influence of macro-level properties of developer collaboration networks on open-source project success". In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 20.2 (2010), pp. 1–27.

[48] Param Vir Singh, Yong Tan, and Vijay Mookerjee. "Network effects: The influence of structural capital on open source project success". In: *Mis Quarterly* (2011), pp. 813–829.

[49] Preston G. Smith and Donald G. Reinertsen. "Time-Based Competition: Strategy and Tactics for a Changing Environment". In: *The Journal of Product Innovation Management* 15.2 (1998), pp. 125–141.

[50] Sulayman K Sowe, Ioannis Stamelos, and Lefteris Angelis. "Understanding knowledge sharing activities in free/open source software projects: An empirical study". In: *Journal of Systems and Software* 81.3 (2008), pp. 431–446.

[51] Zhongfeng Su et al. "Exploratory learning and exploitative learning in different organizational structures". In: *Asia Pacific Journal of Management* 28.4 (2011), pp. 697–714.

[52] Stefan Thomke and Takahiro Fujimoto. "The Effect of "Front-loading" Problem-Solving on Product Development Performance". In: *Journal of Product Innovation Management* 17.2 (2000), pp. 128–142.

[53] Robert Triggs. *What is AOSP? Everything you need to know*. https://www.androidauthority.com/aosp-explained-1093505/. Accessed: 20/05/2022. URL: `https://www.androidauthority.com/aosp-explained-1093505/`.

[54] Mary Tripsas. "Unraveling the process of creative destruction: Complementary assets and incumbent survival in the typesetter industry". In: *Strategic management journal* 18.S1 (1997), pp. 119–142.

[55] Simon Fraser Turner, Will Mitchell, and Richard A. Bettis. "Strategic momentum: How experience shapes temporal consistency of ongoing innovation". In: *Journal of Management* 39.7 (2013), pp. 1855–1890.

[56] Michael L Tushman and Philip Anderson. "Technological discontinuities and organizational environments". In: *Administrative science quarterly* (1986), pp. 439–465.

[57] Georg Von Krogh, Sebastian Spaeth, and Karim R Lakhani. "Community, joining, and specialization in open source software innovation: a case study". In: *Research policy* 32.7 (2003), pp. 1217–1241.

[58] Anu Wadhwa and Suresh Kotha. "Knowledge creation through external venturing: Evidence from the telecommunications equipment manufacturing industry". In: *Academy of Management Journal* 49.4 (2006), pp. 819–835.

[59] Zhaohui Wei, Yunfei Yi, and Hong Guo. "Organizational learning ambidexterity, strategic flexibility, and new product development". In: *Journal of Product Innovation Management* 31.4 (2014), pp. 832–847.

[60] Xiaojing Xu et al. "Model of migration and use of platforms: Role of hierarchy, current generation, and complementarities in consumer settings". In: *Management science* 56.8 (2010), pp. 1304–1323.

[61] Shaker A Zahra and Gerard George. “The net-enabled business innovation cycle and the evolution of dynamic capabilities”. In: *Information Systems Research* 13.2 (2002), pp. 147–150.