Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Karhu, Kimmo; Gustafsson, Robin; Lyytinen, Kalle

Exploiting and defending open digital platforms with boundary resources

Published in:
Information Systems Research

DOI:
10.1287/isre.2018.0786

Published: 01/06/2018

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY-NC-SA

Please cite the original version:
Karhu, K., Gustafsson, R., & Lyytinen, K. (2018). Exploiting and defending open digital platforms with boundary resources: Android's five platform forks. *Information Systems Research*, *29*(2), 479-497. https://doi.org/10.1287/isre.2018.0786

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

This article was downloaded by: [130.233.216.209] On: 12 July 2018, At: 01:44
Publisher: Institute for Operations Research and the Management Sciences (INFORMS)
INFORMS is located in Maryland, USA

Information Systems Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Exploiting and Defending Open Digital Platforms with Boundary Resources: Android's Five Platform Forks

Kimmo Karhu, Robin Gustafsson, Kalle Lyytinen

To cite this article:
Kimmo Karhu, Robin Gustafsson, Kalle Lyytinen (2018) Exploiting and Defending Open Digital Platforms with Boundary Resources: Android's Five Platform Forks. Information Systems Research 29(2):479-497. https://doi.org/10.1287/isre.2018.0786

Full terms and conditions of use: http://pubsonline.informs.org/page/terms-and-conditions

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2018, The Author(s)

Please scroll down for article—it is on subsequent pages

INFORMS is the largest professional society in the world for professionals in the fields of operations research, management science, and analytics.
For more information on INFORMS, its publications, membership, or meetings visit http://www.informs.org

http://pubsonline.informs.org/journal/isre/

INFORMATION SYSTEMS RESEARCH
Vol. 29, No. 2, June 2018, pp. 479–497
ISSN 1047-7047 (print), ISSN 1526-5536 (online)

Exploiting and Defending Open Digital Platforms with Boundary Resources: Android's Five Platform Forks

Kimmo Karhu,[a] **Robin Gustafsson,**[b] **Kalle Lyytinen**[c]

[a] Department of Computer Science, Aalto University, FI-00076 Aalto, Finland; [b] Department of Industrial Engineering and Management, Aalto University, FI-00076 Aalto, Finland; [c] Department of Design and Innovation, Weatherhead School of Management, Case Western Reserve University, Cleveland, Ohio 44106

Contact: kimmo.karhu@aalto.fi, http://orcid.org/0000-0002-0026-4466 (KK); robin.gustafsson@aalto.fi (RG); kalle@case.edu (KL)

Received: September 1, 2016
Revised: June 22, 2017; December 18, 2017; January 31, 2018
Accepted: February 13, 2018
Published Online in Articles in Advance: May 14, 2018

https://doi.org/10.1287/isre.2018.0786

Copyright: © 2018 The Author(s)

Abstract. Digital platforms can be opened in two ways to promote innovation and value generation. A platform owner can open access for third-party participants by establishing boundary resources, such as APIs and an app store, to allow complements to be developed and shared for the platform. Furthermore, to foster cooperation with the complementors, the platform owner can use an open-source license boundary resource to open and share the platform's core resources. However, openness that is too wide renders the platform and its shared resources vulnerable to strategic exploitation. To our knowledge, platform strategies that promote such negative outcomes have remained unexplored in past research. We identify and analyze a prominent form of strategic exploitation called platform forking in which a hostile firm, i.e., a forker, bypasses the host's controlling boundary resources and exploits the platform's shared resources, core and complements, to create a competing platform business. We investigate platform forking on Google's Android platform, a successful open digital platform, by analyzing the fate of five Android forks and related exploitative activities. We observe several strategies that illustrate alternative ways of bundling a platform fork from a set of host, forker, and other resources. We also scrutinize Google's responses, which modified Android's boundary resources to curb exploitation and retain control. In this paper, we make two contributions. First, we present a theorization of the competitive advantage of open digital platforms and specifically expose platform forking as an exploitative and competitive platform strategy. Second, we extend platform governance literature by showing how boundary resources, which are mainly viewed as cooperative governance mechanisms, are also used to combat platform forking and thus sustain a platform's competitive advantage.

History: Panos Constantinides, Ola Henfridsson, and Geoffrey Parker, Special Issue Editors; Rikard Lindgren, Associate Editor. This paper has been accepted for the *Information Systems Research* Special Issue on Digital Infrastructure and Platforms.

Open Access Statement: This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. You are free to download this work and share with others for any purpose, except commercially, if you distribute your contributions under the same license as the original, and you must attribute this work as "*Information Systems Research*. Copyright © 2018 The Author(s). https://doi.org/10.1287/isre.2018.0786, used under a Creative Commons Attribution License: https://creativecommons.org/licenses/by-nc-sa/4.0/."

Funding: This work received partial support from the Digital Disruption of Industry project that the Strategic Research Council [grant 292889] of Finland funded.

Keywords: boundary resources • competitive advantage • digital platform • exploitation • open platform • platform competition • platform forking

Introduction

Digital platforms can be opened in two ways to promote innovation and value generation (Boudreau 2010). A platform owner can grant access to third-party participants by establishing boundary resources (Ghazawneh and Henfridsson 2013), such as APIs and an app store, to allow complements to be developed for the platform. Furthermore, to foster cooperation with the complementors, the platform owner can use an open-source license boundary resource to open its core platform resources (West 2003). An open digital platform (ODP) can thus be defined as an extensible digital core that is opened for third parties to contribute improvements or add complements (de Reuver et al. 2017). The complements can be physical resources (such as a new device) or digital and intangible resources (such as a new application, hereafter referred to as apps). When complements are combined with the platform's core, the value of the platform "bundle" to the user becomes greater than its separate parts (Gawer and Henderson 2007, Nalebuff and Brandenburger 1996). Google Android is a great example of a successful ODP. It currently has, depending on the estimate, a dominant 80%–90% share of the mobile phone market and its app store features over 3 million apps that generate more than 100 billion downloads per year.

However, just as shared resources are exploited in alliances (Lavie 2006), an ODP's openness can render the platform and its shared resources vulnerable to outside exploitative activities. An ODP that has been opened in two ways has two kinds of shared resources: complements that are shared for distribution and platform core resources whose intellectual property rights (IPR) are shared. Because of their digital nature, these resources can be easily copied, reverse engineered or breached and can invite competitors to exploit them as part of a hostile strategy. For example, Amazon has created its proprietary Fire OS platform, which appropriated the open platform core of the Android Open-Source Project (AOSP). Furthermore, Amazon has not only copied Android's core but has expanded the exploitation to its app complements that are shared for distribution. We conceptualize this form of platform-level exploitation as *platform forking* and call the resulting competing platform stack *a platform fork*. In management research, the term "exploitation" commonly refers to an organization's mode of learning related to efficient use of resources (March 1991), but we use the term "exploit" in its literary meaning (Merriam-Webster): "to make use of meanly or unfairly for one's own advantage."

Technically, a platform fork "forks" the platform's core, but strategically the technical fork is a means to create a new platform that directly competes with the host platform while maintaining compatibility with it. Compatibility offers a means to exploit the host platform's complementarities, especially its apps. Because of maintained compatibility, developers' multi-homing costs are marginal and they are, therefore, motivated to "multi-home" (Armstrong 2006, Rochet and Tirole 2003) to the fork. Overall, platform forking allows one to "jump-start" the platform with an existing developer community and to forgo high start-up costs and risks associated with igniting a new platform. Platform forking is a hostile, competitive strategy because the forker contributes little or nothing back to the host platform, does not provide monetary rewards to the host, and does not add value to the host platform's users. By contrast, a platform fork potentially transfers the host's value-generating assets, including the complementary apps, the user base, and data assets, to the fork without accruing significant investments, risks or penalties.

Platform forking is a relatively recent phenomenon and is largely unique to ODPs. It has become common since the emergence of widely used open digital platforms such as Android. Smartphones that use a forked Android platform comprise between 15% and 25% of Android's market share. Apart from gaining market share, forkers can profit financially. Some estimates (Barnett 2011) suggest that creating a fork of the platform core saves at least between $1 and $2 billion in initial development cost and provides significant additional savings in each subsequent version if the forker manages to maintain compatibility. Moreover, because forks promote app multi-homing, a forker can quickly boost its app store and accrue additional revenues from the apps.

Despite growing evidence of hostile, exploitative activities around ODPs, research on open platform strategies (Boudreau 2010) and open innovation (West and Bogers 2014) has focused primarily on the positive effects of openness. However, Dahlander and Gann (2010, p. 706) have called for research into acting "opportunistically in bad faith," while Yoo et al. (2012, p. 1406) has noted the need for firms "to learn how to compete and thrive" with ODPs. Earlier research has reported exploitation by complementors, such as the creation of meta-platforms (Ghazawneh and Henfridsson 2013) or jailbreaking (Eaton et al. 2015). Separately, it has also discussed multiple potentially exploitative activities, such as copyright infringement (e.g., Lemley and Reese 2003), hacking (e.g., Boudreau and Jeppesen 2015), and forking (e.g., Lerner and Tirole 2002, Simcoe and Watson 2016). However, to our knowledge, it has not considered these activities as elements of orchestrated strategies to build a substitute platform that seeks to threaten the host's overall platform business. Pon et al. (2014, p. 988) recognized the "emergence of new proprietary platforms built on the Android operating system," but their focus is on shifts in control points rather than the overall exploitative strategy of platform forking. As a result, we know little of how exploitation is exercised, why and under what conditions platform forks are likely to emerge, how successful forks are constructed or how hosts can successfully respond to platform forking. Specifically, what are the different kinds of shared and boundary resources involved and what is their role in exploitation? To address this research gap, we pose the following research question: How do openness and related governance decisions render an ODP and its resources vulnerable to platform forking and how can the host use its resources to defend against it?

Because of a lack of explicit theorizing and empirics in addressing the research question, we conduct an exploratory multiple case study of Google's Android platform and its forks. In particular, we examine the emergence and evolution of five Android forks and Google's subsequent responses. In analyzing the platforms, we focus on shared and boundary resources to understand their role in platform forking and in defending against it. The study covers a period between November 2007 and March 2017 and relies on an extensive web-based data set, which captures the technical and market evolution of the Android platform and its five platform forks and contains recorded interpretations and rationales for platform changes. The public data set is augmented with executive interviews and an opinion poll solicited from case firms.

Our study shifts attention on open platform strategizing from a cooperative to a competitive perspective and from within a platform ecosystem to between competing platforms. We make two contributions. First, we present a theorization for a resource-based view of an ODP's competitive advantage and specifically expose platform forking as an exploitative platform-level competitive strategy in which a forker directly exploits the host's shared resources. Second, we extend platform governance literature by exposing how governance that is too loose causes an ODP to become vulnerable to platform forking and how the host can combat this activity. As to governance means, we show in particular how boundary resources are not only versatile resources for cultivating a multitude of complementors within an ODP but also offer means with which to defend against exploitation. As such, we expose how boundary resources are critical in extracting (or inhibiting) all four types of rents and are therefore a key determinant of an ODP's competitive advantage.

Openness in Digital Platforms and Platform Forking

Platform openness is about easing the restrictions on the use, development, and commercialization of platform technologies (Boudreau 2010). Decisions concerning platform openness are key to deciding on any ODP architecture and governance. The initial decisions on a platform's openness form part of the ODP's architecture, while the continued adjustment of a platform's openness constitutes part of governance (Schreieck et al. 2016). Next, we briefly review the main elements of ODP architecture and conceptualize two forms of platform openness: *access openness* and *resource openness*. This allows us to synthesize how platform governance influences different forms of platform openness. We conclude by reviewing the consequences of exploiting platform openness as part of a hostile platform forking strategy.

ODP Architecture

An ODP relies on a modular design (Baldwin and Clark 2000); it consists of a set of modules with standardized interfaces that can be combined in multiple ways to generate alternative products or platform features. Baldwin and Clark (2000) present six modular operators for creating alternative designs. These operators (including their combinations) are splitting, substituting, augmenting, excluding, inverting, and porting. For example, Android is built on top of the Linux kernel by integrating additional functions via kernel header files (i.e., augmenting the platform). Each device manufacturer has further adapted the Android stack by introducing additional features. For example, Samsung has integrated its TouchWiz module into its Android stack to differentiate it from competitors' products. Apart from being a modular design with fixed product boundaries, an ODP is also a layered modular architecture that allows fluid product boundaries in which innovation can independently spark at any layer (Yoo et al. 2010). Generally, an ODP's architecture can be modeled as a "stack" (i.e., a layered structure) consisting of three primary components that all involve possibly multiple modules. These components are a (stable) digital core, assets of (varying) complements on top of and around it, and interfaces connecting the two (Baldwin and Woodard 2008, Tiwana et al. 2010).

Two Forms of Platform Openness: Access and Resources

ODP's openness assumes that new modules can be added or existing modules modified within the current platform stack. Openness is multidimensional in that it can be granted to multiple participant groups in varying ways. Traditionally, platform openness has been approached as an activity carried out by the platform host to share the platform's resources, i.e., its IPR. The most common example of such activity is the open-source licensing of the platform's core (West 2003). Eisenmann et al. (2008) broaden the concept of openness by recognizing that a platform host, such as Google, can separately open its platform to distinct participant groups, including users, complementors, and platform providers. In this definition, openness toward complementors focuses on opening access through interfaces, while openness toward platform providers, such as Samsung in the case of the Android platform, is more concerned with openness toward the platform's core resources. In line with this, Boudreau (2010) distinguishes two forms of openness: (1) granting access and (2) devolving control. We adopt his concept and define two forms of ODP openness as follows. *Access openness* refers to the granting of access to external complementors to participate and conduct business on a platform by providing them with dedicated resources to interact with the platform. For example, a host can provide interfaces, such as APIs, to allow outside developers to create new apps on top of the platform. Here, access refers to the access to participate. By contrast, *resource openness* refers to opening the platform's valuable resources by forfeiting the IPR of the resource. For example, a platform owner can open-source a platform's codebase. These two forms are not mutually exclusive and can be used in combination and in distinct ways toward different participant groups.

The rationale for access openness is to spark innovation within the platform ecosystem and induce complementors to use the platform to create additional value that invokes positive network effects (Parker and Van Alstyne 2005, Rochet and Tirole 2003). In access openness, the platform host can extract part of the created value through revenue sharing (such as the percentage of sales or service use) or through other mechanisms (such as selling in-app ads). Such access can also be separately opened to each participant group (Karhu et al. 2014). For example, on the smartphone platforms, the platform host, in addition to granting access to developers, can separately grant access to third-party app store providers. To open access, a platform host can create multiple interfaces that range from technical to regulatory (Baldwin and Woodard 2008, Farrell and Saloner 1992, West and O'Mahony 2008). For example, in addition to APIs, the Android platform contains a hardware abstraction layer (HAL) interface for integrating device complements.

In resource openness, the host sees it as advantageous to open the platform's core resources by forfeiting related IPR (Boudreau 2010, West and Gallagher 2006). For example, in 2007, Google made the Android platform's core open source under the AOSP. Barnett (2011) refers to this form of openness as strategic forfeiture because the platform host voluntarily forfeits the platform core's IPR while seeking to retain control of other parts of the platform to recover the costs of developing the core. In line with this, Google generates most of its revenues within the Android ecosystem from advertisements powered through the use of its search engine, YouTube, and other Google services.

The two forms of platform openness create two distinct types of *shared resources* between the host and other platform participants. In access openness, the platform host grants access to outside complementors who can then submit complements, such as apps, to be run on the platform. Complements become the first type of shared resources that are *shared for distribution* on the platform. They are shared for distribution only because the complementors grant the platform host a restricted right to "possess" and distribute the apps while holding the IPR over the resource. Resource openness creates a second type of shared resource that comes with *shared IPR*. Here, the host forfeits the IPR of the platform's resources by using, for example, open-source licensing. This conceptualization of two types of shared resources is consistent with the use of shared resources "that are intentionally committed and jointly possessed" in alliance studies (Lavie 2006, p. 645).

Governance of ODP

Platform governance refers to the mechanisms through which a platform owner exerts influence over other actors in the ecosystem, such as app developers (Tiwana 2013). It consists of a plethora of hosts' activities, such as the stipulation of decision rights, the selection of formal and informal control mechanisms, and decisions about pricing schemes (Tiwana 2013). The goal of platform governance should be to orchestrate, rather than direct (Williamson and Meyer 2012), and cultivate an ecosystem of complementors for innovation (Wareham et al. 2014). In our study context, which involves the platform host and forkers, governance centers on the host's activities to open the platform and manage its resources and participants within the overall ecosystem.

A central question in platform governance is how to manage the tension between control and openness (Tilson et al. 2010, Wareham et al. 2014). In the ODP context, a platform host needs to balance stability and quality requirements with a need to promote innovation and invite outside contributions (Eaton et al. 2015, Ghazawneh and Henfridsson 2013). Ghazawneh and Henfridsson (2013) present boundary resources as concrete tools and regulations that can govern participants' behaviors and contributions at arm's length. Boundary resources include access-openness-related software tools, such as APIs and software development kits (SDKs), as well as dedicated regulations that govern resource openness, such as open-source licenses. Boundary resources are essential in resourcing and controlling outside contributions on the platform (Ghazawneh and Henfridsson 2013) and, as such, form control points (Tilson et al. 2010, 2012). To promote platform growth, the platform host must proactively engage with the ODP's boundary resources to create leverage while reactively responding to outsiders' diversifying or even competitive approaches to using the resources (Ghazawneh and Henfridsson 2013). Several studies have noted the importance of the host's control of the platform's boundary resources compared to just controlling the platform's core (Baldwin and Woodward 2009, Pon et al. 2014, Schilling 2000). Because of shared use of the platform's boundary resources, the host can never fully control them and their use can be contested by complementors or other actors (Eaton et al. 2015).

Overall, we can extend Ghazawneh and Henfridsson's (2013) developer-focused analysis of boundary resources with a competitive aspect and consider boundary resources as the *software tools and regulations that serve as the interface for resourcing contributions and complements and for governing the resultant shared resources and their producers and consumers at arm's length, including the competitive actions within and between platforms*. Hence, we distinguish between boundary resources, i.e., a means or mechanism that enables resource sharing and permits contributions on the platform, and shared resources, i.e., the valuable resources that provide the contribution. Table 1 summarizes our discussion of the two forms of openness, boundary resources, shared resources, and the host's rationale for promoting them within an ODP.

Platform Forking

In line with Lavie's (2006) theorization in alliance setting, opening an ODP and sharing resources leaves the platform potentially vulnerable to hostile exploitation. Software forking (Robles and González-Barahona 2012) becomes a threat for an ODP if its core resources

Table 1. Two Forms of Platform Openness and Related Resources

Platform openness	Boundary resources	Shared resources	Actor who shares	Type of sharing	Platform owner's rationale
Access openness	API, app store	Complement, e.g., apps	Complementor	Shared for distribution	Generate network effects, and extract value from complementarities
Resource openness	Open-source license	Platform core, e.g., AOSP	Platform owner	Shared IPR	Strategic forfeiture of IPR while recovering costs from somewhere else

have been open sourced. Past research provides multiple, somewhat conflicting, assessments of forking. Technically, a fork is defined as an independent line of development (Robles and González-Barahona 2012) or as a separate pathway that "might remain similar in the code but under control by different programming groups" (Vetter 2016, p. 175). A wealth of economics literature views forking negatively, as it leads to incompatible code bases (Barnett 2011, Lerner and Tirole 2002, Simcoe and Watson 2016, Yoo 2016) and fragments the platform (Parker and Van Alstyne 2009). Wheeler (2015) further clarifies this by pointing out that the intent of creating a competing project defines a fork. From the governance perspective, code forks manifest a challenge (von Krogh and von Hippel 2006) or a failure of cooperation (Viseur 2012), calling for stronger leadership (Fleming and Waguespack 2007). Another, and smaller, stream of research views forking as a means to create derivative works and portrays it positively because it improves the long-term sustainability of the codebase (Gamalielsson and Lundell 2014, Nyman et al. 2012, Nyman and Lindman 2013).

In our study, we combine the technical perspective of forking with the competitive and strategic intent and define platform forking as a process that creates two competing platforms from the same resource base, and which are controlled by different and competing hosts. The definition extends the forking concept into the platform level in that platform forking produces alternative configurations of the full platform stack, not just the technical codebase. On the module level, platform forking involves the forking of software modules, such as the platform's core. However, as Amazon's Fire OS example illustrates, in addition to forking the platform's core, the forker also seeks to "fork" complementary assets. Through these operations, Amazon has built an alternative, competing platform to Android. In this regard, platform forking differs significantly from the reciprocal use of shared resources within alliances. For example, members of the Open Handset Alliance (OHA) use AOSP to build Android variants, but in this setting Google accrues the app store and search revenues generated through all of these variants. By contrast, Google does not receive app revenues from Amazon's Fire, nor does Google have access to users or user-generated data originating within Fire, unless a Fire user uses a Google browser (e.g., Chrome) or launches a dedicated Google search. To prevent such access, the default search engine in Fire has been set to Microsoft's Bing and the default browser is Amazon's Silk. As a result, in terms of business logic and a revenue model, Amazon's Fire platform competes directly with Google's main platform businesses built around Android while, at the same time, Amazon unilaterally exploits Android's core and complementary resources.

We can summarize our argument so far as follows: An ODP can use two forms of openness by establishing boundary resources, which help invite and contribute complements to the platform. However, too-wide openness combined with too loosely established governance mechanisms can render a platform and its shared resources vulnerable to exploitation. Under such conditions, a hostile outside actor, a forker, may circumvent the host's boundary resources and build a competing platform fork by exploiting the ODP's shared resources that cover the core and the complements.

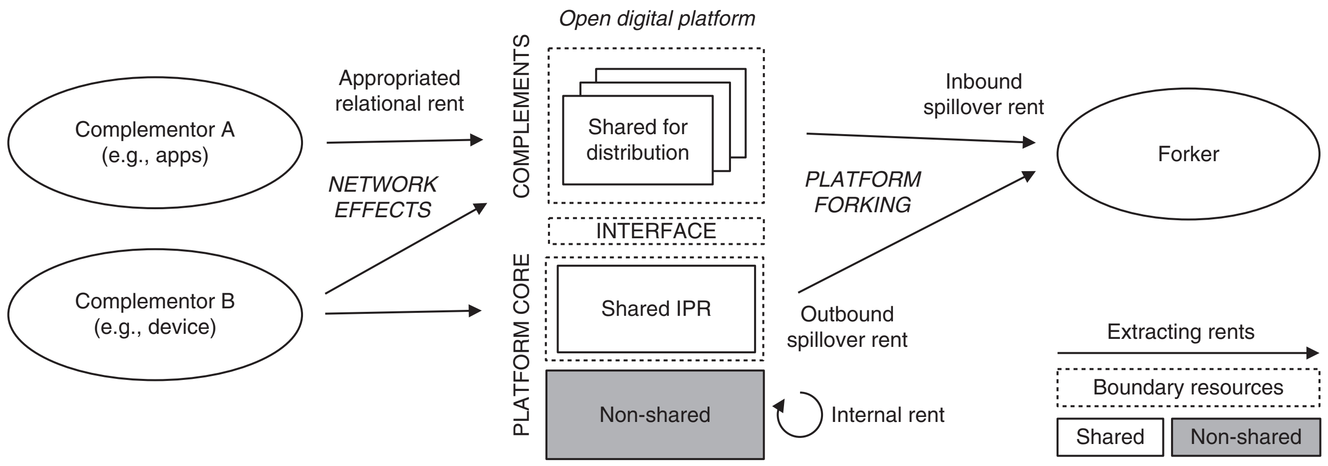
Method, Data, and Analysis

Case Study Goals and Case Selection

To answer our research question, we conducted an exploratory embedded case study (Yin 2009) where the unit of analysis was the platform. Our aim is to reveal platform-forking strategies, their key components, motivations, and outcomes from the forker's and the host's perspective. To this end, we analyzed the original host platform and the resulting forked platforms and their changes as embedded cases.

In selecting the cases, we followed replication logic (Yin 2009) so that each case replicates the platform-forking process and related outcomes. We analyzed all five forks that originated from the Android platform between 2007 and 2017. For each fork, the host remained the same, i.e., the Android platform, whose codebase has been opened through AOSP. None of the studied forkers were members of OHA and did not need to follow the alliance regulations. Thus, each fork potentially imposed a threat and sought to compete directly with Google's business in one form or another. The platform forks studied were (1) the Amazon Fire OS platform, (2) the Xiaomi MIUI platform (initially operating in China only, but which has extended into other Asian markets with an international version of the platform), (3) the Nokia X platform (which was terminated in the spring of 2014, soon after its initial commercial release), (4) CyanogenMOD

Figure 1. ODP and Platform Forking



(a community-based platform that evolved into a start-up firm in 2013), and (5) the Jolla Sailfish platform (built on the Mer open-source project while exploiting some elements of the Android platform).

To reach analytic generalization (Yin 2009), the included cases feature different conditions and contrasting approaches to forking. The cases differ with respect to the economic regions and markets in which the forkers operated (Asia, Europe, the United States). They also differed with respect to the nature of the forker. Some were corporate (Amazon, Xiaomi, Nokia), while others were strongly empowered by a community (CyanogenMOD, Jolla). Finally, Jolla's Sailfish fork is a contrasting case in that it did not fork the whole platform but, rather, only select parts.

ODP Model and Research Approach

We base our analysis on an idea that platform forking can be viewed as similar to the unilateral exploitative use of shared resources among alliance partners (Lavie 2006). A platform ecosystem around ODP can be viewed as a loose alliance governed at arm's length using open-source licenses, APIs, and other boundary resources. We can, accordingly, extend Lavie's (2006) model to the ODP context and platform forking, as illustrated in Figure 1. At the center sits an ODP stack, including the core, interfaces, and complements, where the stack is subjected to access and resource openness. On the left-hand side, we observe positive impacts of these forms of openness as fostering innovation by complementors and leading to network effects. In line with Lavie (2006), the ODP host extracts positive appropriated relational rents from the complements. We extend Lavie's (2006) model to the ODP context and note that the use of two forms of openness generates two types of shared resources: complements that are shared for distribution and the platform core whose IPR is shared. In line with Lavie's (2006) model of exploitation within alliances, on the right-hand side, we observe platform forking as a negative outcome in which the forker unilaterally exploits shared resources of the ODP. In the ODP context, this applies to both types of shared resources. The forker achieves this by smartly using or bypassing the boundary resources established by the host. In Lavie's terminology (2006), this generates negative outbound spillover rents for the host and, conversely, positive inbound spillover rents for the forker.

The above presented analytic model of ODP forking provides us with the conceptual scaffolding to study strategic exploitation of platform forking and defending against it. We operationalized the analytic model to our research setting as follows: For each case, we analyzed strategic activities related to different forms of rent extraction with a focus on platform forking to extract inbound spillover rents, and, correspondingly, defending against it to curtail outbound spillover rent. Our analysis of the platforms' resources focused on shared resources that were exploited and boundary resources that were used as a means to exploit and defend.

Data

The data set covers the evolution of the Android platform, beginning with its launch in November 2007, and its five platform forks from their respective initiation dates through March 2017. We screened published documents on Android's evolution and all five platform forks. After screening the documents, we focused on analyzing thick descriptions of platform changes, their technical details, and legal and regulatory aspects. The available web-based material on the evolution of forks was rich in detail and provided accuracy and reliability and, due to multiple sources, also provided potential for data triangulation. The data set included technology news, case companies' websites (technical information on the platform and company data), developer discussion boards (for hackers' perspective), platforms' histories (e.g., reports on Android

Table 2. Research Process and Sources of Evidence

Analysis step	Data source/evidence ($N = 178$, 1,855 pages)	Analysis outcome	See
1. Explication of activities	Longitudinal data (2007–2017): • platform histories ($N = 23$) ∘ https://www.arstechnica.com ∘ http://www.androidcentral.com ∘ http://www.wikipedia.org, . . . • technology news (60) ∘ http://www.cnet.com ∘ http://www.androidpolice.com ∘ http://techcrunch.com, . . . • executive blog posts (10) ∘ http://officialandroid.blogspot.co.uk/ ∘ https://blog.jolla.com/, . . . • developer discussion boards (25) ∘ https://forum.xda-developers.com ∘ https://www.amazon.com/forum/ ∘ https://together.jolla.com, . . .	Activities database • 66 activities in total • 22 core activities related to exploitation and defense	Appendix A
2. Explication of platform resources involved in activities	Current data (as of 2017): • company web pages ($N = 37$) ∘ https://developer.android.com ∘ https://developer.amazon.com, . . . • legal docs (23) ∘ https://opensource.org/ ∘ platform developer sites	Platform resources database • 5 + 1 platform stacks • 16 boundary resources with five designed functions	Figure 2, Appendix B
3. Identifying contextual conditions	Using the data from earlier steps	• condition for platform forking	2nd ODP Governance section in findings
4. Identifying mechanisms		• platform forking • defense mechanisms	Figure 2
5. Executive corroboration	Interviews ($N = 3$) • CTO of a platform fork (~45 min, face to face) • technology director of a case firm (~45 min, over phone) • regional president of an OHA firm (~45 min, secondary source) Expert opinion poll (5 respondents out of 30 top executives from case platforms)		

and the five platform forks), and legal documents that include terms, conditions, and OSS licenses (see Table 2 for details). As Android is an open-source project with a massive developer community, the technical data and each activity occurring during its evolution are critically monitored and commented on. Included web-based data also cover executive blog posts that comment or report on ongoing forking or defense operations. These documents were reviewed to reveal parties' policy reasoning. Our use of a web-based data set overall recognizes Vesa and Vaara's (2014) call to take online strategizing seriously. Our final database includes 178 documents totaling 1,855 pages, as well as additional interview and survey data among case platform firms. Our data set is similar to recent case studies on iOS evolution and its boundary resources (Eaton et al. 2015, Ghazawneh and Henfridsson 2013).

Data Analysis

Our data analysis consisted of five steps (see Table 2).

The first step involved explicating key development and market activities. The resulting activity database included all activities that were potentially impactful in technical terms, had possible strategic implications, and/or had business implications for the Android platform. We started with open coding (Bryant and Charmaz 2007). For each activity, we next coded the date, the related boundary resource, and the targeted platform-stack elements. We also created a short description of the activity and its rationale. We specifically coded activities that bore exploitative characteristics and looked for subsequent host responses. The database came to include platform and product releases, changes in boundary resources (e.g., modified policies), alliance and collaboration announcements, and legal suits and threats across platforms. In total, we identified 66 such activities. We confirmed the veracity of each activity description by cross-checking the information against another source. Next, we focused on identifying those core strategic activities in which a platform fork exploited the host, the host defended itself or a platform fork accommodated the host's defense. This shortening of the activity list ended with 22 core strategic activities that illustrated the main forms of strategic interplay between the platform host and its forks.

The second step pertained to the identification of those platform resources involved in platform forking or in defending against it. The platform resources database contained information on the platform stacks including shared resources and boundary resources for the host and each platform fork. We focused specifically on identifying those boundary resources within the host platform that were drawn upon by the platform forkers. For each deployed resource in the forks' platform stack, we identified the respective potential source (shared resource) within the host stack and the boundary resource that was potentially used.

As a third step, we studied conditions for platform forking, such as the type of openness exploited and its dimensions. As an output, this step produced the conditions in relation to the two forms of openness and exposed the role of boundary resources in defining them. As a fourth step, we performed a cross-case analysis to identify the common activities and resources used across cases. This led to the identification of separate platform forking strategies. By investigating the timelines of events for each case, we exposed common concepts in the strategic interplay between the host and forkers. During the comparative process, we moved from open coding to selective coding (Bryant and Charmaz 2007). Per this analysis, we synthesized the types of boundary resources and their functions (see Appendix B). The analysis also resulted in the identification of three exploiting activities ("forking," "cloning," and "hacking"). These activities emerged from the data set as common actions that targeted specific boundary resources to exploit the shared resources. As an example of the evolution of coding, our initial coding for boundary resources involved phrases such as "be agnostic of," "isolate from," and "be independent," which were later synthesized under the "loosen couplings" function related to removing tight couplings between the stack layers in question. Another example of coding is a forker "becoming a partner" or "removing an app," which was later phrased, following terminology introduced by Eaton et al. (2015), as "accommodating" the host's policies.

Finally, to validate our analysis findings for veracity and plausibility, we conducted focused semistructured interviews with executives in two of the case firms. To complement the forker perspective, we also analyzed one OHA executive's online interview about Android's future and related strategies. Finally, we conducted an executive opinion poll by reaching out to technology and strategy executives working for, or involved in, case platforms to solicit interpretations of our tentative findings. Using LinkedIn, we obtained five responses for our poll (from requests sent to 30 top executives). Together, these executive perspectives, referred to hereafter as executive corroboration, provided additional substantiation and validation of our analysis.

Findings

We report our findings in four parts. The first section presents the case narratives and the context and exposes the "in vivo" strategic interplay between Google and five forks. The second section disentangles the functions and impact of boundary resources in governing the ODP and identifies the conditions for platform forking. The third section concentrates on identifying successful platform-forking strategies. The final section analyzes how the host can defend against platform forking.

Strategic Interplay between Android and Platform Forks

In total, 22 core strategic activities were identified that illustrated exploitative, defensive or accommodating activities between the host and the forkers or constituted important antecedent activities. These activities are listed in chronological order in Appendix A, which includes a unique identifier (A1–A22) and exact date for each activity. In the Type column, we characterize whether the activity was exploitation by a forker (*Hack, Fork, Clone,* or generic *Exploit*), a host's defense (*Defend*) or a forker accommodating the host's defense (*Accommodate*). For each activity, when relevant, we have marked the boundary resource(s) used (see Appendix B). Furthermore, for each of Google's defensive responses, we mark, in parentheses, the associated exploiting activity by the forker that prompted the response.

On November 5, 2007 (A1), the OHA was announced, and an initial version of Android's codebase and its SDK were published under the AOSP using the Apache open-source license as a boundary resource. Following this announcement, September 23, 2008 (A2), was the official release of the API, SDK, and the Android Market. The first Android phone, HTC Dream/G1, became available on October 20, 2008 (A3). The developer community was quick to hack the platform and the newly released phone. On November 4, 2008 (A4), some developers released instructions on how to root[1] Android and build custom Read Only Memory (ROMs)[2] for it. This event was strategically significant because it technically enabled the subsequent build-up of platform forks. To have a full stack including the apps, on September 21, 2009 (A5), the CyanogenMOD developer community released a hacked package of Google's proprietary apps, including Gmail and Android Market. This event made it possible to run Google apps on any rooted phone. Google responded quickly to this event. By September 24, 2009 (A6), Google sent a cease-and-desist letter to the developer community and threatened legal action. Google explained its legal action in a blog post (September 26, 2009): "Unauthorized distribution of this software harms us just like it would any other business, even if it's done with the best of intentions."

The community temporarily backed off, but it quickly found a way to circumvent the legal threat and separately installed the Google apps package. Subsequently, this became a common practice among custom ROM users. Interestingly, this solution was jointly developed with Google engineers. Steve Kondik, the community leader, revealed this in his blog post on September 30, 2009: "[*a*] lot of people are helping to work many of these issues out, notably the guys from Google [anonymized] who manage the open-source project."

Xiaomi, a Chinese company, was the first commercial fork that benefited from these hacking events. It announced its MIUI platform in late 2010 (A8). Xiaomi released its first phone using the fork in July 2011 (A10). Triggered by the first commercial exploitation of Android outside OHA, Google started to transfer the development of several of its Android apps, e.g., search, music service, calendar, and camera, to a closed-source regime. It effectively ceased to forfeit its IPR related to these apps. This change took place between the release of Android's Froyo on May 20, 2010 (A7), and its Gingerbread version on December 6, 2010 (A9). The latter release made Google's new IPR practices transparent. Consequently, several AOSP versions of the closed-sourced apps remain frozen to the Froyo state.

New threats of exploitation continued to emerge for Android. On September 28, 2011 (A11), Amazon released its first commercial Fire tablet, which runs on a forked Android platform. From Google's perspective, Amazon's move differed significantly from the earlier forks. Xiaomi, at that time, was only a Chinese manufacturer of mobile handsets in a market that Google had already lost due to the Chinese government's stringent policies. By contrast, Amazon operated in the same U.S. and European markets as Google. Moreover, one of Amazon's core businesses is selling content (such as Amazon Prime), which directly competes with Google's YouTube and Google Play business. Furthermore, the default search engine in Fire has been Microsoft Bing since Amazon's Fire HD release on September 9, 2012 (A13). This hampers Google's main search and ad business, which increasingly relies on the user data collected from mobile devices.

The subsequent changes in Android's boundary resources reveal Google's response. On March 6, 2012 (A12), Google rebranded its app store, Android Market, as Google Play. By doing this, Google effectively claimed ownership of the boundary resource. It rebranded the name for that service from a shared Android domain to its own company brand, which it could thereafter fully control. Furthermore, because Amazon's move threatened Google's content business, Google strengthened its competitive position against Amazon, adding music and a bookstore (Amazon's core business) under the rebranded marketplace, which it could now control.

Another defensive act, in the form of a legal move, occurred on September 13, 2012, when Google prevented one OHA member, Acer, from launching an Aliyun phone (an incompatible fork) in partnership with Alibaba (A14, A15). Google issued a related statement (as quoted in Marketing Land): "Compatibility is at the heart of the Android ecosystem and ensures a consistent experience for developers, manufacturers and consumers. Non-compatible versions of Android, like Aliyun, weaken the ecosystem." Alibaba, by contrast, described the exchange in a statement on September 13, 2015 (as quoted in CNET), as: "Our partner was notified by Google that if the product runs Aliyun OS, Google will terminate its Android-related cooperation and other technology licensing with our partner." As Google had contractual power over Acer through the alliance contract boundary resource, it could force Acer (and Alibaba) to yield. Google's swift response highlights how important it was for it to enforce control over the Android ecosystem and ensure the compatibility of the complementors. This situation also reveals hidden tensions within the OHA as to how far other members can "expand" the official Android platform and related resources without incurring a penalty.

Apart from licensing, branding, and legal moves, Google also responded to the continued exploitation through digital means. On September 16, 2012 (A16), Amazon published its clone of the Google Maps API. Soon thereafter, on September 26, 2012 (A17), Google released a new boundary resource, i.e., a client library called Google Play Services. Apart from making Google's own API updates more flexible for developers, this new boundary resource also served a defensive purpose against Amazon's cloned APIs. By using the library, developers become dependent on a proprietary Google library that is only available on official Android releases, making it more difficult for developers to migrate their apps to any forked Android platform. Furthermore, by freeing API development from official platform releases, Google could speed up the development of its APIs. This has made it more difficult for API copycats, such as Amazon, to keep up. For example, it took two years for Amazon to provide interface parity[3] for the Google Maps API v2 update (A22).

Lately, new developments have unfolded within the strategic relationship between Google and a platform forker, Xiaomi. Xiaomi recently launched its devices internationally by making them available in Hong Kong, Singapore, and India (A18). To secure the availability of Google's services in international markets, Xiaomi made a licensing deal with Google; its international devices now have a preinstalled set of Google services, including Google Play and Gmail. This licensing deal has not been confirmed by any formal public announcement, but Hugo Barra, a Xiaomi director, noted on March 3, 2015, at the Mobile World Congress in Barcelona, Spain, that Xiaomi is indeed partnering with Google (as quoted in Techcrunch): "We are a GMS [Google Mobile Services] partner. We have been from the very first day that we sold our first device outside of China."

A platform fork may also sometimes inject its boundary resources back into the host platform. On September 9, 2014 (A19), Amazon released the "Amazon App for Android Phones" (essentially a shopping app and an app store) on Google Play featuring Amazon's tangible offerings and intangible digital content, such as the apps published for Fire OS. Google reacted swiftly (A20), changing its developer distribution agreement on September 25, 2014, from restricting only apps "whose primary purpose is to facilitate the distribution of software applications and games for use on Android devices outside of the Market" to "any Product which has a purpose...". Amazon accommodated Google's new policy (A21), releasing a more limited "Amazon Shopping App" between September 25 and December 11, 2014, and issued a statement saying (as quoted in Android Police on December 11, 2014): "Google subsequently changed their Developer Distribution Agreement on September 25. As a result, we removed the app from Google Play and published the Amazon Shopping app."

ODP Governance Through Boundary Resources and Conditions of Platform Forking

To understand how platform governance decisions affect conditions of platform forking, we next focus on the boundary resources, which are the practical governance means. Overall, our analysis identified 16 distinct boundary resources that served five distinct functions (for a full list, see Appendix A). Earlier research has identified resourcing and securing as two generic functions for boundary resources (Ghazawneh and Henfridsson 2013). Based on our analysis, we further decompose resourcing into four more specific functions: *define openness*, *facilitate*, *loosen couplings*, and *capture value*. Here we term securing as *control*. Note that one boundary resource may serve several functions. Next, we review the boundary resources that we identified in the Android platform with respect to these functions and show how the host's choices with regard to openness and control defined the condition for platform forking.

Boundary resources are primary means that define the level of openness, i.e., they open up specific parts of the codebase or specific interfaces within the platform. Under access openness, Android's APIs offer a set of dedicated interfaces that can be used by app developers. Furthermore, Google has made Android open to potential third-party app stores. To this end, it comes with the Android Package (APK) format and a manifest specification that defines the protocols for distributing Android apps through an app store. As to the second form of openness, i.e., resource openness, the open-source license attached to the code forms the primary boundary resource. It defines the rights related to the use of the open-sourced code. For AOSP, Google has chosen the Apache license. The choice of a specific open-source license has crucial strategic consequences for the ODP. By choosing a reciprocal copyleft license, such as a General Public License (GPL) or a "Lesser" GPL (LGPL), instead of a permissive license, such as Apache, MIT or BSD, the platform owner can force license beneficiaries to share their modifications and improvements to the source code (West and O'Mahony 2008). In the ODP context, complements that link to the platform core through APIs add another layer of complexity to the licensing. For example, GPL's "virality" feature may, under some conditions, enforce the complements to be opened as well. Weaker reciprocal licenses, such as LGPL, restrict enforcement solely to the modifications made to the open-sourced code and explicitly exclude software that links through APIs. As evidence of the uncertainty about complements' treatment under GPL, Linus Torvalds states in the copyright notice for the Linux kernel: "This copyright does *not* cover user programs that use kernel services by normal system calls—this is merely considered normal use of the kernel, and does *not* fall under the heading of 'derived work.' " Google probably chose the permissive Apache license due to uncertainty about complements' future treatment and to allow its alliance partners to keep enhancements proprietary, thereby increasing their interest in investing in Android.

Apart from mere opening, boundary resources facilitate complementors' work, loosen couplings between platform modules, and are involved in value capture. As to facilitation, Android comes with an SDK that facilitates app development and provides an app store for distributing and monetizing apps on Android. In addition, Android has a client-library boundary resource called Google Play Services, which is intended to address Android's version-fragmentation problem by facilitating more frequent Google service and API updates independent of the rigid operating-system releases. This client library also loosens couplings, in this case, between the OS and APIs. Similarly, the API and runtime environment boundary resources loosen couplings between the OS and apps, whereas HAL provides this function between the OS and the hardware. Furthermore, because Google Play Services wrap[4] the APIs and their proprietary client-side implementations, it has, over time, become a critical boundary resource for Google's value-capture logic. All user data powering Google's search and ad business flow through them. Likewise, Android's app store is another boundary resource that serves a value-capture function.

Boundary resources are also used to control the complementor's behaviors. Android comes with a developer distribution agreement that controls developers and the distribution of their apps. Correspondingly, the Compatibility Definition Document (CDD) and Compatibility Test Suite (CTS) boundary resources are used to control device complementors. They ensure that devices remain compatible with the platform and all of the software that runs on top of it. Google exerts rigid control through these boundary resources by requiring devices to pass all of the device tests before Google services such as Google Maps and Google Play are licensed for the device. Finally, Google uses the Mobile Alliance Distribution Agreement (MADA) to control how its business-critical apps, such as Search and Google Play, will be placed on Android devices sold by OHA partners.

How do Google's choices for access and resource openness within Android, by using specific boundary resources, affect the conditions for platform forking? First, by introducing resource openness, Google has inevitably enabled platform forkers to leapfrog upfront investments into the underlying platform technology. Furthermore, by offering a permissive license, Google has little visibility and few possibilities for benefiting from changes made within the forks unless the forker voluntarily shares them. Second, by simultaneously introducing an open APK definition and permissive terms in distribution agreements (i.e., not assuming exclusivity in app distribution), Google has essentially granted access for third-party app store providers. Indeed, Android's distribution policies for developers (as of May 25, 2017) states: "You can distribute your apps through any app marketplace you want or use multiple marketplaces."

Although such policies can be justified as a means of fostering indirect positive network effects, under these conditions, Google enables developers technically (through APK) and legally (due to distribution terms) to submit their apps to any competing fork. Because the app store boundary resource forms a central control point of critical app complementors, to avoid platform forking, it should not be opened to third parties and should remain in exclusive host control. To summarize, simultaneously providing resource openness and granting relatively wide access openness for brokering functions creates conditions sufficient for platform forks to emerge.

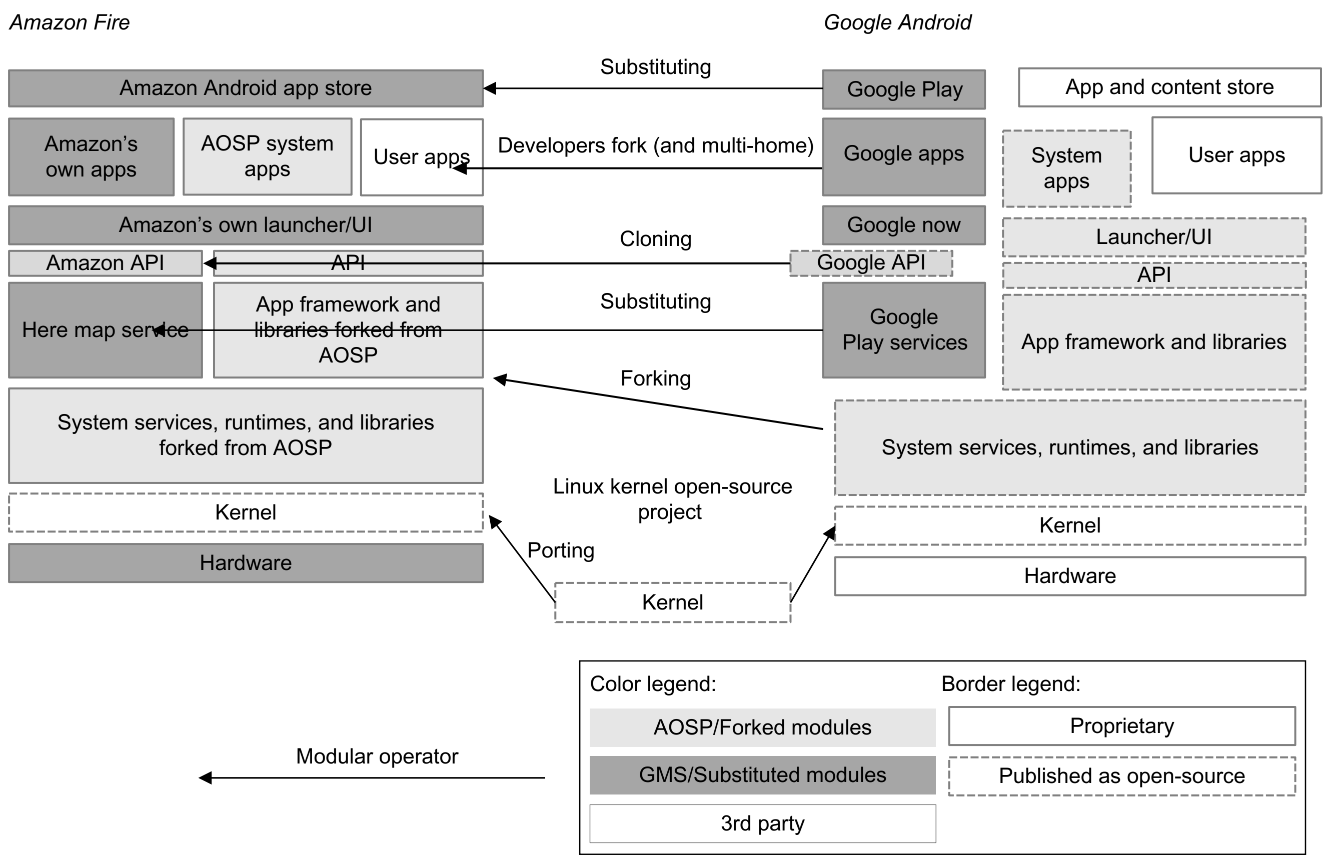
Platform-Forking Strategy

Our analysis illustrates varied approaches to achieve platform forking and reveals how platform forking is a platform-level strategic action that is distinct from mere software forking. Forkers orchestrated multiple combined activities to exploit Android's shared resources, at core and complement layers, to build a complete competing substitute platform.

As to the platform core, whose IPR Google has shared under AOSP, forkers have forked the code and directly exploited it to avoid most up-front, and, later, yearly, investments in platform technologies. Jolla has followed an alternative approach by developing its own core platform based on the open-source Mer platform. However, to enter the platform business, in addition to the core, forkers need complementing apps. Because app complements are shared for distribution while their IPR remains proprietary, their exploitation is not straightforward and needs to be carried out indirectly. The CyanogenMOD community solved this problem by hacking the Google Play app store and the Google Play Services on which apps relied. In so doing, the community simply enabled all Android apps on its platform. Amazon and Nokia, which operate as large legitimate businesses, cannot publicly promote hacking and have, therefore, had to invent another approach. They both replaced Google Play with an in-house-developed app store. To get the actual apps into their app store, they also meticulously cloned all of the APIs on which developers for the Android platform depend and substituted all proprietary implementations behind these APIs. These moves enabled developers, who developed their original apps for Google Play, to multi-home them on the competing forked platform. For example, Amazon and Nokia use Here's map service, which comes with literal copies of Google's service APIs and makes it cost almost nothing for developers to migrate and multi-home apps to the fork. To illustrate this point, Here's documentation (as of May 25, 2017) states: "If you have an application already utilizing Google Maps V1 or V2, porting is possible with almost no changes in the code by using the wrapper libraries provided with the SDK." Finally, Xiaomi adopted a twofold strategy. In the mainland-China market, with its local app economy, Xiaomi replaced the APIs and services with local Chinese versions, whereas in international markets, to secure access to existing Google services and third-party apps, Xiaomi struck a licensing deal with Google.

Platform forking can be conceptualized as a process in which the forker deploys a sequence of modular operators generating an alternative, complete modular platform design. However, platform fork combines modular operators with the exploitative activities of forking, cloning, and hacking that use the host platform's boundary resources to exploit its shared resources. Platform-forking conceptualization shifts the analytical perspective from code level forking to platform level. In software forking, the code is branched and modified, resulting in two separate codebases. However, in platform forking, instead of lines of code, a forker can be viewed as operating on the modules of the modular layered architecture to produce a forked platform. To achieve platform-level forking, a forker still forks individual modules, such as the platform core that has been open sourced. In addition,

Figure 2. How Amazon Built Its Fire Platform by Forking Android



it needs to substitute other modules that are proprietary, further forking the layered modular architecture at the platform level. It also invites ODP host complementors to fork their modules in the complement layers. As a complete example of the forking of a platform, Figure 2 illustrates how Amazon configured the Amazon Fire OS platform by substituting and exploiting Android's platform resources. In an orchestrated effort, Amazon forked the AOSP, substituted Google's proprietary modules, and ensured that developer-facing APIs are an exact clone so that developers can multi-home their apps on Amazon's Fire OS platform.

From a strategic perspective, platform forking results in a separate forked platform stack, competing with the ODP host's entire platform business. Platform forking can be viewed as an opportunistic strategy for rapid market entry that permits leapfrogging, avoids heavy up-front investments, and curtails technical and economic risks. Furthermore, platform forking creates a favorable situation in which complementors can multi-home their complements on the fork. By doing this, platform forking helps the forker solve the difficult chicken-and-egg dilemma faced during platform ignition. Orchestrating apps on your own from scratch is not easy, as illustrated by the failed Windows Phone platform backed up by Microsoft and its vast resources.

Platform forking can also be partial. Among the platform forks, Jolla is an unusual case: It did not exploit the platform core; instead, it exploited complements on app and device layers. Android relies on Java technology, which isolates apps from the underlying platform core. As a result, the apps can be run on any platform, such as Jolla Sailfish, provided that the platform shares a compatible runtime environment. By replacing the genuine Android Runtime Environment (ART) with a substitute obtained from the company Myriad, Jolla enabled the use of Android apps on its Sailfish platform. Similar to the idea of cloning APIs, Jolla has also cloned and ported Android's device interfaces (HAL) using available open-source libraries (libhybris and bionic), which provide HAL compatibility for generic Linux distributions (such as Mer). Using this approach, any Android manufacturer can use its existing device design to produce a Sailfish phone, as Sony has recently done. Jolla exemplifies an alternative strategy of not forking the platform core but, rather, of exploiting not just one but multiple complementor sides of the Android platform.

Defending Against Platform Forking Using Boundary Resources

The host can actively use boundary resources to defend against platform forks. We identify six ways in which Google strategically responded to the emerging exploitation of its Android platform. First, with regard to the exploitation of resources shared through resource openness, Google used the open-source license boundary resources by retracting from openness to close the further development of the selected parts of the core. Second, Google used the alliance contract as a boundary resource to bring unfriendly OHA members back in line. Google also defended itself against exploitation of the complements shared for distribution through access openness. The app-store boundary resource was defended in three ways: by threatening legal action, by claiming ownership of the service through rebranding it and strengthening its role in the ecosystem, and by modifying the developer distribution agreements. Finally, a client-library boundary resource was introduced to make API cloning difficult and to raise barriers against multi-homing.

In three of the responses, the boundary resources used had a direct connection to the exploiting events that prompted the response. For example, when Xiaomi drew upon the open-source license boundary resource, Google reacted by using the same boundary resource and moved its development into a closed source. Similarly, when Amazon threatened to steal Android's app-store business, Google increased its control over this boundary resource. Finally, when Amazon copied the API boundary resource, Google increased its control by wrapping it inside a client library (a new kind of boundary resource). The executive corroboration of the logic and rationale of these responses produced mixed results as to their expected effects and salience. Our interview with a platform forker's CTO (chief technology officer) supported the conclusion that API cloning is harmful, and that Google's creation of Google Play Services was a clever move to defend against it. Some respondents within the executive poll, however, classified Google's responses as a normal means of controlling the ecosystem. Only Google's sending of a cease-and-desist letter was classified as a strategic defense against a fork.

Our longitudinal analysis reveals that Google has, in many ways, been successful in protecting its competitive advantage by curtailing competitive duplication among most forks. CyanogenMOD was the first platform fork, but despite its commercialization effort, it did not succeed. This caused the hacker community to resurrect the related codebase into a new LineageOS moniker. Another largely failed attempt was Nokia X, which lasted less than a year. Although Xiaomi has been successful in China, Google has constrained its competitive duplication in Western markets by making it a licensor of its services. Amazon Fire is the only platform that has maintained success as a forked platform. Despite Amazon's less successful smartphone release, the company (as of November 2017), with its Fire platform and tablet, was the third-largest tablet manufacturer in the world with 10.9% market share.[5] As to the duplication of complements, Amazon Appstore (as of March 2017)[6] featured 600,000 apps, whereas Google Play had 2.8 million.

Discussion

Our study contributes to the ongoing research stream on open-platform strategies and their effects (Boudreau 2010, Eisenmann et al. 2008, Parker and Van Alstyne 2009). Previous research on the topic has followed a cooperative perspective, which assumes that collaboration between platform participants promotes platform growth and value generation. Exceptions include Parker and Van Alstyne (2009), who present six challenges that platform owners face, and Eaton et al. (2015) and Ghazawneh and Henfridsson (2013), who identify complementors' competitive actions that challenge the host's control of the platform's boundary resources. Pon et al. (2014) recognize proprietary platforms built using open platforms, but their focus is on shifts in control points and not the overall exploitative strategy of platform forking. Our study shifts attention on open platform strategizing from a cooperative to a competitive perspective and from within a platform ecosystem to between competing platforms. We theoretically frame this competition in light of the competitive advantage literature that has studied interfirm alliance networks (Lavie 2006), and we adapt it into the ODP context. We present a resource-based view of an ODP by differentiating between two types of shared resources and boundary resources, and we expose their distinct role in creating and sustaining competitive advantage. Competitive advantage is created not only through the leveraging of network effects for growth but also through opportunistic and exploitative strategies, such as platform forking. We also contribute to the platform-governance literature by revealing how openness that is too wide and too loosely governed shared resources render an ODP vulnerable to strategic exploitation. As to governance means, we show not only how boundary resources are versatile resources for cultivating a multitude of complementors within an ODP but also how they offer means with which to defend against exploitation to sustain an ODP's competitive advantage.

Competitive Advantage of an ODP and Platform Forking

In line with the relational view of competitive advantage (Dyer and Singh 1998) and with the extension of the resource-based view to interfirm networks (Lavie 2006), we note that in the ODP context, competitive advantage is primarily achieved by extracting relational rents. This is by contrast to sticking with valuable, rare, inimitable, and non-substitutable (VRIN) resources (Barney 1991) or information systems (IS) capabilities (Bharadwaj 2000, Wade and Hulland 2004) as sources of competitive advantage. To understand competitive advantage in the networked setting, Lavie (2006) distinguishes between shared resources and non-shared resources. The focal firm endows shared resources to the alliance, expecting that this will generate common benefits. We extend shared resource conceptualization in the ODP context by suggesting that platform complements, such as apps, should be viewed as a second type of shared resources. The above model remains consistent with the dominant cooperative perspective of platform strategy (e.g., Boudreau 2011, Eisenmann et al. 2006) and with recent theorization on how competitive advantage relies on complementarities and network effects (Koch and Windsperger 2017, Sun and Tse 2009). From this standpoint, our research presents boundary resources, such as an app store and an API, as means with which to extract *appropriated relational rents* (Lavie 2006) from the complementarities of the shared resources. Furthermore, as APIs are used to collect user data, boundary resources also play a central role in extracting *internal rents*. An example of this is Google's use of proprietary machine-learning algorithms in its advertising business, which relies on data collected through various platform APIs.

Furthermore, as Lavie (2006) points out, shared resources can generate unilateral accumulation of spillover rents, where one party gains private benefits. This is an even greater risk in the ODP context, where the audience for sharing is larger, the audience is at arm's length, and is therefore more difficult for the platform owner to control. Our study exposes, for the first time (to our knowledge), how platform forking by exploiting the host's shared resources forms a unilateral rent-extraction mechanism that generates negative *outbound-spillover rents* for the host and positive *inbound-spillover rents* for the forker. As with cooperative rent extraction, boundary resources constitute the primary target through which the platform fork exploits the shared resources that the host controls. Instead of its own boundary resources, a forker relies on use of the host platform's boundary resources. Correspondingly, to counter this move, the host can modify its boundary resources or create new ones to protect against platform forking.

To summarize, we advance a resource-based view of an ODP and contribute to the competitive advantage research in two ways. First, we explicate two types of shared resources for an ODP and expose their vulnerability to exploitation through an opportunistic platform forking strategy. Earlier research on platform competition has assumed that each platform creates its own resource base, and in line with this assumption, has suggested platform strategies, such as platform envelopment (Eisenmann et al. 2011), market entry with a higher quality platform (Zhu and Iansiti 2012), and distinctive positioning (Cennamo and Santalo 2013). By contrast, platform forking forms a competitive platform strategy that directly attacks and diminishes the host platform's competitive advantage by opportunistically exploiting the platform core and the complementary resources. Second, we contribute by exposing how boundary resources are critical in extracting (or inhibiting) all four types of rents, and are, therefore, a key determinant of an ODP's competitive advantage. As the parallel success of Google's Android and Apple's iOS in the same market illustrate, balancing between extracting positive and inhibiting negative rents can be accomplished in many ways. Apple has maximized internal rents from its proprietary resources. By avoiding openness, except while tapping into critical appropriated relational rents from app complements, it has protected itself against outbound spillover rents. Google, by contrast, following a more open platform strategy, has sought to maximize appropriated relational rents but at the same time has exposed itself to the threat of outbound spillover rents realized though platform forks.

Our analysis of platform forking as a competitive strategy provides additional insights for research on modular designs. Whereas Baldwin and Clark (2000) concentrate on economizing designs within vertical organizations by using modular operators, platform forking and the exploitative bundling of platform modules can be seen as an extension of the set of modular operators that operationalize the platform composition logics of ODPs and their ecosystems. Furthermore, our study extends past research on forking that has focused on processes of software forking (Robles and González-Barahona 2012) and their outcomes, such as fragmentation challenges (Parker and Van Alstyne 2009). Our study exposes, for the first time (to our knowledge), platform forking as an exploitative platform-level strategy, where a competing, hostile organization creates a forked alternative version of the entire platform.

Defending Against Platform Forking Using Platform Governance

ODP settings are hostile competitive environments that require hands-on active defending. The host must actively engage in platform governance to combat the exploitation of its shared resources. Previous platform-governance literature has predominantly focused on the positive outcomes of cultivating complementors (Wareham et al. 2014) through such means as control and the division of decision rights (Tiwana 2013). We expand this view by exposing the competitive threat posed by platform forking and by identifying its conditions resulting from ODP governance that is too loose. In terms of governance means, we extend the dominant cooperative perspective of boundary resources (e.g., Ghazawneh and Henfridsson 2013) to a fuller articulation of how they can be defensively used to respond to exploitation. Eaton et al. (2015) show how complementors often contest platform control, resulting in a distributed tuning of boundary resources between the host and its complementors. Our findings extend this kind of control mechanism for defending against *competing platforms*.

Platform forking can happen when the ODP host grants *both* access and resource openness and remains too flexible in governing the access, for example, by allowing complements to be distributed on *any* digital distribution service. In VRIN parlance (Barney 1991), it is in the host's best interest to keep its complements *rare* in relation to other platforms. Thus, the host must protect not only its own resources but also the resources that the complementors share. Although the complements' IPR is protected, due to loosened couplings through API and APK boundary resources, complements (as black-boxed modules) are easily "transferrable" and can be multi-homed to a competing compatible platform. In addition, although the platform owner can use isolating mechanisms (Rumelt 1984), such as withdrawing from openness or mounting a contractual enforcement, our study suggests that software designs, in the form of boundary resources, are novel means with which to protect the complements. For example, using a client-library boundary resource the platform host can speed up its development of the API boundary resources on which complements depend, thereby achieving a temporal advantage over the forker, who must constantly synchronize. Altogether, our research has illustrated six ways to use boundary resources to combat the competitive actions of forkers. In line with distributed tuning (Eaton et al. 2015), ensuing competitive interplay between the host and the forkers occurs over time. Our results illustrate that, over nine years, the platform's boundary resources became the constant center of strategic activity involving the platform host and its forkers. This resonates with Koch and Windsperger's (2017, p. 22) claim that "the higher the degree of digitization, the more the firm creates competitive advantage by actively shaping the digital environment." The strategic interplay between Google's Android and its forks highlights the strategic role of identifying and using specialized and dedicated digital assets to launch offensive or defensive actions.

We also contribute to the research on the cooperative mechanisms associated with boundary resources. Whereas previous research has examined the role of developer complementors (Eaton et al. 2015, Ghazawneh and Henfridsson 2013), our study posits boundary resources as an essential component in governing any complementor's operations. In the case of Android, HAL as well as CDD and CTS formed unique boundary resources for controlling device manufacturers. Ghazawneh and Henfridsson (2013) identify resourcing and securing functions for boundary resources. Our analysis expands resourcing to multiple separate functions. Boundary resources *define openness* (for access and resources) and *facilitate* complementors' work. Furthermore, boundary resources are not only used to jointly *capture value* with complementors but also act as critical intermediary resources through which valuable user data can flow to the platform host. Finally, our findings illustrate how the constant "digitizing" (Tilson et al. 2010, p. 749) of boundary resources, such as introducing the runtime environment and HAL in the case of Android, continually *loosens couplings* between the platform stack's layers, thus fostering complementors' activities but at the same time posing new threats to the platform host.

Managerial Implications

To prevent platform forking, ODP managers must pay closer attention to how they design boundary resources. Although firms often seek to maximize the generative effect of digital resources that are essential for platform growth, minimizing the effects of weak points that allow for continued platform forking is critical. Therefore, it is advisable to design boundary resources in ways that leave room for maneuvering at a later time. Preventing platform forking is extremely tricky because boundary resources are mostly designed in the initial stages when the entire platform ecosystem does not yet exist and when the desire and need for ignition are pressing. Hence, additional critical assumptions must be made about relevant actors' likely behaviors toward boundary resources when the platform resources eventually become valuable. For example, if Google had restricted app distribution on Android for third-party app stores, Amazon would have been unable to create a forked platform with its own app store. To counter exploitation, the host can also use boundary resources to combat exploiters. Furthermore, additional boundary resources can be introduced to control exploitation instead of engaging in resource- and time-consuming legal interventions. Digital strategizing offers additional benefits, such as rapid deployment, customizability, and precise targeting.

Finally, due to the strategic importance of boundary resources, their ownership may be contested. The ongoing legal case of Oracle versus Google illustrates this (see Menell 2016 for details). So far, the U.S. Supreme Court has confirmed that the sequence, structure, and organization (SSO) of a large enough API (such as Java API) can receive a copyright. Recently, the U.S. Court of Appeals for the Federal Circuit also ruled that Google's reuse of the APIs does not constitute fair-use and remanded the case to the lower district court to determine damages. As our findings illustrate, copying (i.e., cloning) the API is a central operation in platform forking. For this reason, the final rulings of ongoing court proceedings are of utmost importance for any ODP manager and for the future of platform forking.

Conclusions

Overall, our results expose the ODP ecosystem not only as a cooperative environment but also as a hostile competitive environment demanding proactive strategizing. The current study has been limited to platform forking as it pertains to the Android platform. We assume that similar exploitation approaches have been taken toward other ODPs and will also be taken in the future as access and resource openness are increasingly used together as part of platform strategies. We expect the emergence of the Internet of things, autonomous vehicles, and other embedded digital systems to form the next fertile battleground for such activities. In this regard, the growing phenomenon of sharing and exploiting digital resources in multiple ways should provide ample opportunities for replicating and refining our findings. Overall, our study indicates that a more granular model of platforms, including their various forms of openness, boundary resources, and shared resources, is required to understand strategic phenomena in today's digital age.

Although our study has focused on an ODP in which software is shared, our generic model should also apply to ODPs in which the resource being shared and complemented is data. Furthermore, the concept of sharing requires further research. In addition to forfeiting IPR or handing over possession of a resource, what are other ways of sharing, i.e., forfeiting the VRIN properties (Barney 1991) of a digital resource? Consequently, what are distinct ways in which to protect against the possible exploitation of varied forms of sharing? We invite further research to elaborate on and evaluate our model and to advance a more complete resource-based view of competitive advantage for an ODP. Finally, our study highlights that IS research rooted in a deeper understanding of digital technologies' roles and functions has much to contribute to research on platform strategies. We hope our initial exposure to theorizing on the competitive dynamics of ODPs can grow to a substantial topic in strategic-management research, and we invite researchers across these diverse fields to join us and explore this fascinating and dynamic topic.

Acknowledgments

This work has greatly benefited from constructive comments from three anonymous reviewers as well as the special issue's guest editors. All faults, however, remain the authors'.

Appendix A. Strategic Interplay Between Host and Forks Using Boundary Resources

No.	Date	Strategic action	Type	Boundary resource used
A1	Nov. 5, 2007	OHA announced and Android published (prerelease)		BR1, BR3, BR9
A2	Sept. 23, 2008	API, SDK, and Android Market published (official release)		BR3, BR8, BR9, BR10
A3	Oct. 20, 2008	HTC Dream/G1, first Android phone released in the U.S.		
A4	Nov. 4, 2008	Community published instructions to root HTC Dream/G1 device with Android and installed custom ROM on it	Hack (A3)	BR1
A5	Sept. 21, 2009	CyanogenMOD hacked Android Market into its custom ROM	Hack	BR8
A6	Sept. 24, 2009	Google sent cease-and-desist letter to CyanogenMOD community	Defend (A5)	
A7	May 20, 2010	Android 2.2 Froyo released		
A8	Aug. 16, 2010	Xiaomi MIUI commercial fork released	Fork	BR1
A9	Dec. 6, 2010	Android 2.3 Gingerbread: Google's new policy of closing source code became evident as selected AOSP apps were frozen to Froyo state (A6)	Defend (A8)	BR1
A10	July 13, 2011	Xiaomi M1 phone released in China (A8)	Fork	
A11	Sept. 28, 2011	Amazon Fire: serious challenge to Android in U.S. and European markets	Fork	BR1, BR8
A12	March 6, 2012	Books, music added to Android Market and rebranded as Google Play	Defend (A11)	BR8
A13	Sept. 9, 2012	Amazon announced a new Fire HD tablet with Microsoft Bing as the default search engine instead of Google Search	Exploit	
A14	Sept. 13, 2012	Acer (OHA member) and Alibaba announced forked Aliyun phone	Fork	BR1
A15	Sept. 13, 2012	Google successfully threatened Acer to back off from releasing Aliyun phone	Defend (A14) → Accommodate	BR16
A16	Sept. 16, 2012	Amazon announced Map API (a clone of Google Maps)	Clone	BR3
A17	Sept. 26, 2012	Google released proprietary Google Play Services wrapping APIs	Defend (A16)	BR3, BR7
A18	March 31, 2014	Xiaomi Redmi (int.) announced, with Google services preinstalled	Accommodate	BR7, BR8
A19	Sept. 9, 2014	Amazon released Amazon app (capable of selling Amazon's digital content, including forked apps) in Google Play	Exploit	BR8, BR11
A20	Sept. 25, 2014	Google prohibited apps in Google Play that sell apps or games	Defend	BR11
A21	Sept. 25-Dec. 11, 2014	Amazon removed Amazon app and replaced it with a new Amazon Shopping app that does not include apps or games	Accommodate (A20)	BR8, BR11
A22	Dec. 16, 2014	Amazon published updated Maps API v2	Clone	BR3

Appendix B. Boundary Resources and Their Strategic Functions for ODP

Function	Boundary resource[a]	Description	Control aspect	Examples in our cases
Define openness	BR1: Open-source license	Defines resource openness for software, for example, allows third parties to study the software, contribute improvements, or make their own derivatives	Choose between licenses/families to reach desired effect	ASL 2.0 for AOSP
Define openness	BR2: End-user license agreement (EULA)	For proprietary software, defines contract between licensor and purchaser	Fully modifiable	Sailfish EULA
Loosen couplings; Define openness	BR3: Application programming interface (API)	Grants access openness for app complementors. Provides generic libraries and services for developers so that they can focus on the specific functionality of the app	Extend/remove functionality from the API. Use API key, signing, and authentication mechanisms to restrict access	Location API Java API
Loosen couplings; Define openness	BR4: Hardware abstraction layer (HAL)	Grants access for device complementors. It is the standard interface that device manufacturers need to implement for the OS	Limit API to ease adaptation	Android HAL
Loosen couplings; Define openness	BR5: Application packaging definition	Indirectly grants access for the alternative app store by providing the specification for distribution of apps through any channel	Use digital signing for identification and to integrate with API key mechanisms	APK and application manifest
Loosen couplings	BR6: Runtime environment	Loosens the coupling between platform and apps by running apps in a sandboxed runtime environment on top of the platform	Build proprietary implementation/ be compatible with standard	Android Runtime (ART)
Loosen couplings; Facilitate; Capture value	BR7: Client library	Bypasses manufacturer-controlled OS updates to distribute APIs and service updates more efficiently. A value-capture mechanism for user data flowing to the platform	Can be made proprietary and available only for official platform users	Google Play Services
Facilitate; Capture value	BR8: App store	Provide a distribution and monetization channel for complementors and for the platform owner	Keep exclusive, or allow alternative app stores	Google Play
Facilitate	BR9: Software development kit (SDK)	SDK lowers threshold by providing APIs and developer tools necessary for building, testing, and debugging apps for the platform	Restrict access to registered developers or to specific development platforms	Android SDK
Control	BR10: SDK license	Defines terms and conditions for using SDK	Fully modifiable	SDK license
Control	BR11: Distribution agreement	Govern and control collaboration for distribution of complements	Fully modifiable	Developer Distribution Agreement
Control	BR12: Other service agreements	Govern and control collaboration regarding various services (including ads, mapping)		Here Terms and Conditions
Control	BR13: Policy guidelines	Various policies and guidelines for complementors		AdSense program policies
Control	BR14: Compatibility definition document (CDD)	Ensure compatibility among devices, the software platform, and apps built on top of it	Limit scope to ease up integration vs. extend to secure platform unity	Android CDD
Control	BR15: Compatibility test suite (CTS)		Require passing the test suite to join the platform	Android CTS
Control	BR16: Alliance contract	Ensure and agree on own and other parties' interests in the alliance	Fully modifiable	Android MADA

[a]We present boundary resources as generalized complete artifacts. It is also possible to identify each subcomponent, for example, each clause in the terms and conditions, as separate boundary resources (see Eaton et al. 2015).

Endnotes

[1] This refers to gaining superuser rights and access within Unix platforms.

[2] Custom ROM building commonly refers to the practice of replacing the original system and its firmware (residing in the ROM with a new, customized version.

[3] Interface parity refers to equivalent APIs.

[4] Essentially, Google Play Services contains the APIs, but the term "wrap" is used here to highlight the shielding purpose and packing aspect that make APIs controllable by Google and upgradable as an APK file. For further details, see https://developers.google.com/android/guides/overview.

[5] https://www.idc.com/getdoc.jsp?containerId=prUS43193717.

[6] https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

References

Armstrong M (2006) Competition in two-sided markets. *RAND J. Econom.* 37(3):668–691.

Baldwin CY, Clark KB (2000) *Design Rules: The Power of Modularity* (MIT Press, Cambridge, MA).

Baldwin CY, Woodard CJ (2008) The architecture of platforms: A unified view. Working paper, Harvard Business School, Boston.

Baldwin CY, Woodard CJ (2009) The architecture of platforms: A unified view. Gawer A, ed. *Platforms, Markets and Innovation* (Edward Elgar Publishing, Cheltenham, UK) 19–44.

Barnett JM (2011) The host's dilemma: Strategic forfeiture in platform markets for informational goods. *Harvard Law Rev.* 124(8): 1863–1938.

Barney J (1991) Firm resources and sustained competitive advantage. *J. Management* 17(1):99–120.

Bharadwaj AS (2000) A resource-based perspective on information technology capability and firm performance: An empirical investigation. *MIS Quart.* 24(1):169–196.

Boudreau K (2010) Open platform strategies and innovation: Granting access vs. devolving control. *Management Sci.* 56(10): 1849–1872.

Boudreau KJ (2011) Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation. *Organ. Sci.* 23(5):1409–1427.

Boudreau KJ, Jeppesen LB (2015) Unpaid crowd complementors: The platform network effect mirage. *Strategic Management J.* 36(12):1761–1777.

Bryant A, Charmaz K (2007) *The SAGE Handbook of Grounded Theory* (SAGE, London).

Cennamo C, Santalo J (2013) Platform competition: Strategic trade-offs in platform markets. *Strategic Management J.* 34(11): 1331–1350.

Dahlander L, Gann DM (2010) How open is innovation? *Res. Policy* 39(6):699–709.

de Reuver M, Sørensen C, Basole RC (2017) The digital platform: A research agenda. *J. Inform. Tech.*, https://doi.org/10.1057/s41265-016-0033-3.

Dyer JH, Singh H (1998) The relational view: Cooperative strategy and sources of interorganizational competitive advantage. *Acad. Management Rev.* 23(4):660–679.

Eaton B, Elauf-Calderwood S, Sørenson C, Yoo Y (2015) Distributed tuning of boundary resources: The case of Apple's iOS service system. *MIS Quart.* 39(1):217–243.

Eisenmann T, Parker G, Van Alstyne M (2011) Platform envelopment. *Strategic Management J.* 32(12):1270–1285.

Eisenmann T, Parker G, Van Alstyne MW (2006) Strategies for two-sided markets. *Harvard Bus. Rev.* 84(10):92–101.

Eisenmann TR, Parker G, Van Alstyne M (2008) Opening platforms: How, when and why? Working paper, Harvard Business School, Cambridge, MA.

Farrell J, Saloner G (1992) Converters, compatibility, and the control of interfaces. *J. Indust. Econom.* 40(1):9–35.

Fleming L, Waguespack DM (2007) Brokerage, boundary spanning, and leadership in open innovation communities. *Organ. Sci.* 18(2):165–180.

Gamalielsson J, Lundell B (2014) Sustainability of open source software communities beyond a fork: How and why has the LibreOffice project evolved? *J. Systems Software* 89:128–145.

Gawer A, Henderson R (2007) Platform owner entry and innovation in complementary markets: Evidence from Intel. *J. Econom. Management Strategy* 16(1):1–34.

Ghazawneh A, Henfridsson O (2013) Balancing platform control and external contribution in third-party development: The boundary resources model. *Inform. Systems J.* 23(2):173–192.

Karhu K, Tang T, Hämäläinen M (2014) Analyzing competitive and collaborative differences among mobile ecosystems using abstracted strategy networks. *Telematics Informatics* 31(2): 319–333.

Koch T, Windsperger J (2017) Seeing through the network: Competitive advantage in the digital economy. *J. Organ. Design* 6(1):6.

Lavie D (2006) The competitive advantage of interconnected firms: An extension of the resource-based view. *Acad. Management Rev.* 31(3):638–658.

Lemley MA, Reese RA (2003) Reducing digital copyright infringement without restricting innovation. *Stan L. Rev.* 56:1345.

Lerner J, Tirole J (2002) Some simple economics of open source. *J. Indust. Econom.* 50(2):197–234.

March JG (1991) Exploration and exploitation in organizational learning. *Organ. Sci.* 2(1):71–87.

Menell PS (2016) API copyrightability bleak house: Unraveling and repairing the Oracle v. Google jurisdictional mess. *Berkeley Tech. Law J.* 31(3):1515–1595.

Nalebuff BJ, Brandenburger AM (1996) *Co-Opetition* (HarperCollins, London).

Nyman L, Lindman J (2013) Code forking, governance, and sustainability in open source software. *Tech. Innovation Management Rev.* 3(1):7–12.

Nyman L, Mikkonen T, Lindman J, Fougère M (2012) Perspectives on code forking and sustainability in open source software. Hammouda I, Lundell B, Mikkonen T, Scacchi W, eds. *Open Source Systems: Long-Term Sustainability*, IFIP Adv. Inform. Comm. Tech., Vol. 378 (Springer, Berlin Heidelberg), 274–279.

Parker G, Van Alstyne M (2009) Six challenges in platform licensing and open innovation. Working paper, Tulane University, New Orleans.

Parker GG, Van Alstyne MW (2005) Two-sided network effects: A theory of information product design. *Management Sci.* 51(10): 1494–1504.

Pon B, Seppälä T, Kenney M (2014) Android and the demise of operating system-based power: Firm strategy and platform control in the post-PC world. *Telecomm. Policy* 38(11):979–991.

Robles G, González-Barahona JM (2012) A comprehensive study of software forks: Dates, reasons and outcomes. Hammouda I, Lundell B, Mikkonen T, Scacchi W, eds. *Open Source Systems: Long-Term Sustainability*, IFIP Adv. Inform. Comm. Tech., Vol. 378 (Springer, Berlin Heidelberg), 1–14.

Rochet JC, Tirole J (2003) Platform competition in two-sided markets. *J. Eur. Econom. Assoc.* 1(4):990–1029.

Rumelt RP (1984) Toward a strategic theory of the firm. Lamb R, ed. *Competitive Strategic Management* (Prentice Hall, Englewood Cliffs, NJ), 556–570.

Schilling MA (2000) Toward a general modular systems theory and its application to interfirm product modularity. *Acad. Management Rev.* 25(2):312–334.

Schreieck M, Wiesche M, Krcmar H (2016) Design and governance of platform ecosystems-key concepts and issues for future research. Working paper, Technical University of Munich, Munich.

Simcoe T, Watson J (2016) Forking, fragmentation and splintering. Working paper, Boston University, Boston.

Sun M, Tse E (2009) The resource-based view of competitive advantage in two-sided markets. *J. Management. Stud.* 46(1):45–64.

Tilson D, Lyytinen K, Sørensen C (2010) Research commentary–Digital infrastructures: The missing IS research agenda. *Inform. Systems Res.* 21(4):748–759.

Tilson D, Sørensen C, Lyytinen K (2012) Change and control paradoxes in mobile infrastructure innovation: The Android and iOS mobile operating systems cases. *Proc. 45th Hawaii Internat. Conf. Systems Sci.* (IEEE Computer Society, Los Alamitos, CA), 1324–1333.

Tiwana A (2013) *Platform Ecosystems: Aligning Architecture, Governance, and Strategy* (Morgan Kaufmann, Waltham, MA).

Tiwana A, Konsynski B, Bush AA (2010) Research commentary–Platform evolution: Coevolution of platform architecture, governance, and environmental dynamics. *Inform. Systems Res.* 21(4):675–687.

Vesa M, Vaara E (2014) Strategic ethnography 2.0: Four methods for advancing strategy process and practice research. *Strategic Organ.* 12(4):288–298.

Vetter GR (2016) Opportunistic free and open source software development pathways. *Harvard J. Law Tech.* 30(1):S167.

Viseur R (2012) Forks impacts and motivations in free and open source projects. *Internat. J. Adv. Comput. Sci. Appl.* 3(2): 117–122.

von Krogh G, von Hippel E (2006) The promise of research on open source software. *Management Sci.* 52(7):975–983.

Wade M, Hulland J (2004) Review: The resource-based view and information systems research: Review, extension, and suggestions for future research. *MIS Quart.* 28(1):107–142.

Wareham J, Fox PB, Cano Giner JL (2014) Technology ecosystem governance. *Organ. Sci.* 25(4):1195–1215.

West J (2003) How open is open enough?: Melding proprietary and open source platform strategies. *Res. Policy* 32(7):1259–1285.

West J, Bogers M (2014) Leveraging external sources of innovation: A review of research on open innovation. *J. Prod. Innovation Management* 31(4):814–831.

West J, Gallagher S (2006) Challenges of open innovation: The paradox of firm investment in open-source software. *RD Management* 36(3):319–331.

West J, O'Mahony S (2008) The role of participation architecture in growing sponsored open source communities. *Indust. Innovation* 15(2):145–168.

Wheeler D (2015) Why open source software/free software (OSS/FS, FOSS, or FLOSS)? Look at the Numbers!, https://www.dwheeler.com/oss_fs_why.html#forking.

Williamson PJ, Meyer AD (2012) Ecosystem advantage: How to successfully harness the power of partners. *California Management Rev.* 55(1):24–46.

Yin RK (2009) *Case Study Research: Design and Methods*, 4th ed. (Sage, Thousand Oaks, CA).

Yoo CS (2016) Open source, modular platforms, and the challenge of fragmentation. Working paper, University of Pennsylvania, Philadelphia.

Yoo Y, Henfridsson O, Lyytinen K (2010) Research commentary–The new organizing logic of digital innovation: An agenda for information systems research. *Inform. Systems Res.* 21(4):724–735.

Yoo Y, Boland RJ, Lyytinen K, Majchrzak A (2012) Organizing for innovation in the digitized world. *Organ. Sci.* 23(5):1398–1408.

Zhu F, Iansiti M (2012) Entry into platform-based markets. *Strategic Management J.* 33(1):88–106.