1-1-2010

GOVERNING THIRD-PARTY DEVELOPMENT THROUGH PLATFORM BOUNDARY RESOURCES

Ahmad Ghazawneh
Jönköping International Business School, ahmad.ghazawneh@ihh.hj.se

Ola Henfridsson
Viktoria Institute & University of Oslo, ola.henfridsson@viktoria.se

GOVERNING THIRD-PARTY DEVELOPMENT THROUGH PLATFORM BOUNDARY RESOURCES

Completed Research Paper

Ahmad Ghazawneh
Jönköping International Business School
Jönköping, Sweden
ahmad.ghazawneh@ihh.hj.se

Ola Henfridsson
Viktoria Institute & University of Oslo
Gothenburg, Sweden
ola.henfridsson@viktoria.se

Abstract

Third-party development is increasingly relevant for software platform owners seeking to generate complementary assets in the form of applications. The governance of such development involves two seemingly conflicting goals: the maintenance of platform control and the transfer of design capability to users. A key element in simultaneously accommodating these goals is platform boundary resources. However, so far, there is a dearth of theoretical accounts of the role of boundary resources and the process by which such resources can be used to govern third-party development. Drawing on case study research of Apple's iPhone developer program, we synthesize boundary objects theory and innovation networks literature to develop a process perspective of third-party development governance through boundary resources. In doing this, our research extends and complements existing governance literature and contributes new knowledge about an alternative form of system development.

Keywords*:* Third-party development, governance, boundary resources, platform, innovation, innovation networks, boundary object.

Introduction

Third-party software development is increasingly relevant for proprietary platform owners. The magnificent success of Apple's third-party developer program, generating an enormous range of different applications for iPhone and iPad users, has stimulated significant interest in such development. Yet, the decision to develop, or tap into, third-party developer programs is no guarantee for creating successful innovation networks. In fact, consumer device firms such as Nokia and SonyEricsson have governed third-party development much longer than Apple without necessarily achieving the same level of success. In a proprietary context, the governance of such development involves two seemingly conflicting goals: the maintenance of platform control and the transfer of design capability to users.

Reviewing the literature, there exists an extensive stream of research on organizing open source software development for generating external software contributions (von Krogh and von Hippel 2006). However, the difference between platform ownership and open source sponsorship is profound. Rather than making the platform a commons for communal participation (Benkler 2006; Ostrom 2005), platform owners maintain control over the platform and seek to establish boundary resources that ”transfer design capability to users” (von Hippel and Katz 2002, p. 824) for generating complementary assets in the form of applications (cf. Teece 1986). Such resources typically consist of software development kits (SDK), application programming interfaces (API), and/or other tools that enable application development. Over time, such resources become an important element in governing emerging innovation networks.

In this paper, we investigate platform governance with a specific emphasis on the boundary resources offered to third-party developers. Surprisingly little is known about the actual process of governing third-party development through boundary resources. Our literature review shows that most related research can be found in the area of open source. While important as a backdrop, this literature has little to offer when it comes to analyzing the tension between maintaining platform control and, at the same time, stimulate third-party developers to join forces with the platform owner by developing applications. The research question addressed in this paper is therefore: *What is the governance process by which proprietary platform owners can simultaneously maintain platform control and stimulate third-party development through platform boundary resources?* In order to address this research question, we engaged in detailed empirical analysis of Apple's use of boundary resources in governing the iPhone innovation network. We collected extensive data on changes in boundary resources made in response to Apple's perceived needs of simultaneous platform control and a wide scope of motivated iPhone developers. Synthesizing boundary object theory (Bergman et al. 2007; Star and Griesemer 1989) and innovation networks literature (Boland et al. 2007; Chesbrough et al. 2006; Van de Ven et al. 2008; Yoo et al. 2009), we develop a process perspective (Langley 1999) with which to understand the governance of third-party development for software platforms. This perspective has significant implications for both research and practice.

The remainder of this paper is structured as follows. Section two reviews the related literature on platform governance to identify the need of research into boundary resources for third-party development. The next section presents a synthesis of the boundary object perspective and innovation network literature to build a theoretical basis for our analysis of Apple's developer program. After presenting the research method, we then present the case story, portraying four distinct phases of Apple's use of boundary resources for governing third-party development. Section six presents our analysis of the case yields a process perspective with implications for research and practice, while the last section concludes the paper.

Related Literature

Platform Governance and Control

Platforms involve the development of common resources from which to generate derivative products and services (Robertson and Ulrich 1998). Over time, it has become evident that the use of platforms coincides with a number of critical organizational issues that affect their governance and control. For instance, the open source literature reports important insights on knowledge-creation (Lee and Cole 2003), new forms of commitment and motivation (von Hippel and von Krogh 2003), pooling of heterogeneous user interests (Franke and von Hippel 2003), and new competitive dynamics (Fitzgerald 2006; Garud et al. 2002; West and Gallagher 2006). These insights have accordingly formed the basis for exploring governance structures for open source projects (Markus 2007; West 2003) including legal structures (O'Mahony 2005), relationships between vendors and open source communities (Dahlander and Magnusson 2005), and role structures (Mockus et al. 2002).

While traditional studies typically conceive of governance as the establishment of control structures to meet changing IS role demands (Allen and Boynton 1991; Brown 1997; Dixon and John 1989; Zmud et al. 1986) or the enabler of fusing business and IT by controlling the formulation, implementation, and adoption of IT (Van Grembergen et al. 2003; Weill and Ross 2005), most platform governance research is indeed found in the open source literature. Addressing collective action dilemmas, development coordination problems, and creating a better climate for contributors, the open source governance literature tends to focus on structures, rules, practices, and norms for nurturing open source communities (Markus 2007). Markus' extensive review distinguishes two strands of literature. First, the 'monolithic' strand that depicts open source governance as a unitary phenomenon by contrasting it with the traditional way of developing and governing proprietary software (Raymond 1999), seeing it as a different approach to innovation (von Hippel and von Krogh 2003), and even describing it as a distinctive method of governing productive activity (Demil and Lecocq 2006). Second, the 'multidimensional' strand, emphasizing multiple types and variations of governance mechanisms, examines the diversity in OSS licenses (de Laat 2005), legal organization structures of OSS (O'Mahony 2005), relationships between vendors and OSS communities (Dahlander and Magnusson 2005), role structures of OSS communities (Mockus et al. 2002), and the operation of technical business processes (Scacchi 2002).

The Challenge of Third-Party Development in the Proprietary Context

The possibility of becoming the obligatory passage point of application developers has triggered significant interest in platform governance in the proprietary context. Software platform owners increasingly recognize the importance of supporting application developers to build and sustain platform innovation (Evans et al. 2006; Messerschmitt and Szyperski 2003). Using examples such as Microsoft and Palm (Gawer and Cusumano 2002), as well as Google, Amazon, Apple, and eBay (Tapscott and Williams 2006), the literature suggests platforms as the basis for software leadership (Gawer and Cusumano 2002).

Such leadership comes typically from the wealth of the platform's complementary assets in the form third-party applications and services. They are essential for the platform's user value and the potential success of the emerging innovation network (Taudes et al. 2000; Teece 1986). It is therefore considered important to design platform resources that facilitate a shift of design capability to external actors (von Hippel and Katz 2002), who typically make a business on developing applications for the platform. The platform owner may then reap the benefits of distributing, brokering, and operating the developed applications, transforming the owner from a software producer into a distribution channel (Meyer and Seliger, 1998).

However, the path towards such benefits may be filled with long-term battles with competitors (Ferguson 2005; Gawer and Cusumano 2008). This is not surprising since the innovation network around the platform typically involves multiple and heterogeneous actors (Boland et al. 2007; Yoo et al. 2009). More importantly, the proprietary platform owner deals with a delicate problem, seeking to maintain its control of the platform and its value proposition, at the same time as it must maintain acceptance of this control across the innovation network. This problem is much more profound in the proprietary context than in the open source context.

Given the challenge of third-party development in the proprietary context, it is vital to develop new knowledge about its nature and how it plays out in the practice of platform governance. Despite the governance contributions in the open source literature, however, little is known about the means by which platform governance is accomplished through direction, control, and coordination of third-party developers through the common resources of a platform. Indeed, there is a dearth of theoretical accounts that examine the boundary resources that enable third-party development.

Theoretical Basis

Seeking to understand the governance of third-party development through boundary resources, we synthesize the innovation networks literature (Boland et al. 2007; Chesbrough et al. 2006; Van de Ven et al. 2008; Yoo et al. 2008; 2009) and the boundary objects perspective (Bergman et al. 2007; Star and Griesemer 1989). First, the innovation networks literature provides a basis for exploring platform governance from the perspective of (a) control and coordination and (b) knowledge resources (Yoo et al. 2009). Second, boundary objects theory provides a basis for understanding how and why platform boundary resources such as SDKs and APIs enable coordination of activities across multiple knowledge resources where heterogeneous but co-operating actors are tied together (Bergman et al. 2007; Briers and Chua 2001; Star and Griesemer 1989).

We view third-party development as an innovation network distributed across multiple actors and technologies (Boland et al. 2007; Chesbrough et al. 2006; Van de Ven 2005; Van de Ven et al. 2008; Yoo et al. 2008; 2009). The formation of a network of actors secures the supply of required resources, knowledge, and capabilities (Van de Ven et al., 2008). In this regard, the knowledge resources are distributed across many actors, where the degree of innovation complexity determines the extent of knowledge heterogeneity (Yoo et al. 2008). With regard to control and coordination, innovation networks can operate along a continuum ranging from the quite centralized networks you would find in the traditional industry sectors such as the automotive industry (Clark 1985; Henfridsson et al. 2009), to the relatively decentralized networks that are formed around open source software (Ljungberg 2000).

While open source innovation networks are characterized by decentralized control of fairly homogeneous knowledge resources (Yoo et al. 2008), third-party development is characterized by centralized control by the platform owner and relatively heterogeneous knowledge resources. The heterogeneity of knowledge resources comes from the diversity of market segments and niches that application-centered software platforms tend to offer. Innovation in third-party development is mostly driven by knowledge originating from diverse disciplines and social worlds, which become integrated through the platform of which its owner exercises control (Yoo et al. 2008; 2009). One of the main key challenges that face platform owners is to maintain tight integration at the platform level, while allowing loose couplings horizontally to cater for the needs across different knowledge communities (Baldwin and Clark 2000; Yoo et al. 2009). This can be understood by conceptualizing boundary resources as boundary objects.

We use boundary objects theory (Bergman et al. 2007; Carlile 2002; Star and Griesemer 1989) to understand how platform owners can combine centralized platform control with decentralized pooling of heterogeneous knowledge resources. Essentially, platform boundary resources can be viewed as a boundary object, which is an artifact plastic enough to cut across multiple social worlds by providing enough structure to support several parties and their employed activities within separate social worlds (Bergman et al. 2007; Star and Griesemer 1989). As Star and Griesemer (1989) argue, boundary objects can maintain a common identity across heterogeneous sites while still strongly structured in individual use.

Examples of boundary objects types to mediate different actors in third-party development include: repositories, coincident boundaries, and ideal types (Star and Griesemer 1989). First, repositories supply a common reference point of data designed to deal with heterogeneity caused by differences in unit of analysis. They provide shared definitions and values for solving problems. Examples of a repository are a database, library, or museum. Second, coincident boundaries represent common objects, with different contents, resembling the boundaries and dependences of actors involved. These common objects enable actors to use their different perspectives relatively autonomously and still share a common referent. An example of a coincident boundary is a Gantt chart, roadmap, or a workflow. Finally, ideal types are objects that do not accurately describe the details of any single subject but can be observed and then used across different functional settings. An example of an ideal type object is a diagram, prototype, mockup or computer simulation.

Drawing on this theoretical basis, we investigated the process by which platform owners can govern third-party development through boundary resources.

Methodology

We conducted detailed case study research of Apple’s iPhone developer program since its inception. Using multiple data sources, we adopted Romano et al.’s (2003) methodology for analyzing Internet-based qualitative data to build a process perspective (Langley 1999) of governing third-party development through boundary resources.

Research Context and Case Selection

Announced in January 2007, Apple’s iPhone was an unconventional mobile device equipped with numerous high-quality and patented features. Early on, Apple introduced a developer program and an associated delivery channel that would allow third-party developers to design and distribute native applications on top of iPhone’s operating system. In addition, Apple opened a digital distribution and delivery channel called the “AppStore”, which allowed users to browse, search, and download third-party applications. Although the device itself received positive reviews, analysts agree that Apple’s developer program is a contributing factor to the tremendous growth of applications, application downloads, and registered application developers (see Figure 1).

There are a number of reasons why we chose Apple’s iPhone as our case for studying boundary resources in third-party development. First of all, the iPhone developer program is a unique or extreme case (Yin 2009), which is especially suited for analyzing governance in third-party development. Its short and so far successful history involves a number of governance changes where boundary resources played a vital role. This makes the case suitable for our research purposes, which do not necessarily require a successful case but data manifesting enough variety in governance to enable grounded theorizing. Indeed, Apple transformed its governance strategy a number of times over the study period. This allowed us to isolate specific attempts to govern third-party development for tracing underlying mechanisms of each episode. This enabled so-called temporal decomposition, which is important to establish process variance in process studies (Langley 1999). In addition, Apple’s governance strategy changes were reflected in boundary resources including the SDK and APIs. Finally, there exist substantial amounts of data on Apple’s iPhone, making detailed study on the governance possible on the basis of publicly available data.

Data Collection and Analysis

Data from six primary sources informed this research (Table 1). Multiple data sources are valuable for creating valid generalization forms and constructs, as well as for improving data quality (Creswell 2007; Seale 1999; Soy 1996).

Table 1. Data Sources

Data Sources	Description
Interviews	Two interviews with Apple’s CEO, Steve Jobs • 6-minute video recorded interview by ABC News. • Transcribed interview, by Time magazine. Two interviews with Apple’s Senior VP Worldwide Product Marketing, Phil Schiller. • Transcribed interview by The Engadget (*highly profiled Group-edited blog about technology*). • Transcribed interview by Business Week. One interview with a developer. • 26-minute recorded and transcribed interview with notable iPhone developer. The interview was conducted by two well-known high-tech analysts.
Press releases and announcements.	• All press releases collected from Apple’s online press release library (January 2007 – April 2010). 40 press releases were selected for further analysis. • Developer news and announcements published by Apple at the iPhone Dev Center.
Conferences, workshops and special events.	Data collected from video archives and transcriptions of keynotes, speeches, and presentations took place at: • MacWorld Conference & Expo: MacWorld 2007 and MacWorld 2008. • Apple Worldwide Developers Conference (WWDC): WWDC2007, WWDC 2008 and WWDC 2009. • Apple’s SDK events: iPhone SDK 2008, iPhone SDK 2009 and iPhone SDK 2010. • Apple’s special events: Rock and Roll event 2009 and Tablet event 2010.

Documents	All publically available case documents such as the Registered iPhone developer agreement, iPhone human interface guidelines, and SDK agreements.
E-Mails	More than 15 messages between Apple and developers, as well as developers and the public. These messages were publically revealed by developers in their websites, or found on the collected online articles.
Online articles	More than 480 articles from multiple online sources: • General magazines, newspapers and journals such as BusinessWeek.com, NYTimes.com and WSJ.com. • Technology-focused magazines and journals such as ComputerWorld.com, MacWorld.com, and TheRegister.co.uk. • Highly profiled Group-edited blogs about technology such as TechCrunch.com, GigaOM.com and Engadget.com • Highly profiled tech news and analysis websites focusing on Apple news, its products and marketing strategies such as AppleInsider.com, iLounge.com and Roughlydrafted.com

We adopted Romano et al. (2003)'s data analysis method for making sense of the collected data material. It provides a structured approach to analyze the rich, interesting, dynamic data existing on Apple's developer program. Our data analysis proceeded as a three-step process: elicitation, reduction, and visualization. First, we used six data sources to elicit relevant data to be included in our case database. Our initial search queries included keywords such as iPhone, Apple, AppStore, iPhone platform, and combinations of these keywords. We concentrated our searches to the time period of January 2007 to April 2010. Then, in view of the massive material, we started by selecting relevant data material for our study. The selection was based on an intensive and carefully review of all collected data types, where we were mainly looking for content that contained our initial coding (Charmaz 2006) categories such as 3rd party developers, SDK, API, platform governance, developers' community, and iPhone ecosystem.

The selected data was stored in a Qualitative Data Analysis (QDA) software system based on the day/month data appeared online, which helped us tracing the historical process and securing a correct timeline of events. Using the QDA software, we then started coding the selected data corresponding to the initial categories. Using our theoretical basis including concepts such as innovation network, knowledge heterogeneity, and boundary resources, we identified major key events and activities related to Apple's developer program. Finally, we visualized our findings as a process decomposed into four phases, which served as the backbone for generalizing our findings into a process perspective of third-party development through boundary resources.

Results

Our data analysis yielded four major phases by which Apple governed third-party development through boundary resources. In what follows, we describe the four phases. Figure 1 provides a timeline of each identified phase and traces releases of device/platform and boundary resources, growth statistics (number of developers, applications, and download rate), and key events.

Phase I: Apple's Safari browser strategy (July 2007 - March 2008)

In July 2007, Apple announced its initial third-party development strategy at the Apple Worldwide Developers Conference (WWDC 2007). The boundary resource used was the Safari web browser. In his keynote, Apple's CEO, Steve Jobs, said:

> *So we've [Apple] got an innovative new way to create applications for mobile devices, really innovative and its all based on the fact that iPhone has the full Safari inside it, the full safari engine is inside an iPhone. And it gives us tremendous capabilities more than is ever been in a mobile device to this date. And so you [third-party developers] can write amazing web 2.0 and AJAX apps that look exactly and behave exactly like apps on the iPhone. And these apps can integrate perfectly with iPhone services; they can make a call, they can send an email, they can look-up a location on Google maps.*

As emphasized by Jobs, the main advantages of using Safari as boundary resource were the distribution, update process, and security of applications:

> *After you [third-party developers] write them you have instant distribution, you don't have to worry about distribution just put them on your Internet server. And they are really easy to update, just change the code on your own server, rather than having to go through this really complex update process. And they are secure, with the same kind of security you have used for transactions with Amazon or a Bank, and they run securely on the iPhone, so they don't compromise its reliability or security. And guess what, there is no SDK that you need, you got everything you need if you know how to write apps using the most modern web standards to write amazing apps for the iPhone today.*

After the release of the iPhone [June 27, 2007], a growing number of companies ported their applications to the Safari web browser. Google, Flicker, New York Times, YouTube, Twitter, and Facebook were among the first movers to bring their applications to the iPhone. Supporting its strategy, Apple held a 'confidential' and 'invite-only' iPhone Tech Day Workshop for developers on July 23, 2007. Open to Apple Developer Connection members, the objective was to educate developers how to create and optimize web applications for the iPhone. In the Apple Developer Connection only e-mail, Apple clarified the structure and content of the workshop:

> *The iPhone Tech Day Workshop is an intensive, hands-on 1-day session with in-depth technical presentations followed by coding & debugging with access to the experts to provide 1-on-1 assistance. The day will include: – Ensuring Safari on iPhone compatibility – Optimizing Web Content for Safari on iPhone – Best practices for media and synchronized data with iPhone – Coding and Testing.*

Early on after this workshop, information about iPhone's development was spread on mailing lists and developer forums. It did not take long until new frameworks and sample codes were released, which multitudes of other developers integrated into their own applications. Services like mobile browsing via Widgets appeared. All efforts were trying to leverage the iPhone's development capabilities, and to make use of any external and open API to by-pass the Safari web browser and to minimize page load times.

The web browser strategy of Apple generated considerable controversy among developers, telecom executives, consulting firms, venture capitalists, and tech-journalists. The majority viewed this strategy as an immense failure in reinventing the cell phone business. Some held extreme views, predicting this strategy to make iPhone fail. As an example, venture capitalist Paul Kedrosky noted:

> *Is Apple serious that it won't let third-party developers build software for the thing? If so, and put simply, the device will fail. A closed-box consumer electronics mentality will work in music players, but the future of mobile devices is as a platform, and that requires developer*

Apple's strategy was criticized by huge number of developers. At a special iPhone event organized by William Hurley, an executive at the software maker BMC, hundreds of developers expressed their desires of writing native applications for the iPhone. Later on, the iPhone was 'Jailbroken', opening up the possibility for users to install any third-party native applications via unofficial installers. In response, Apple changed their mind completely. In a message posted at the '*Hot News*' section of Apple's Web site on October 17, 2007, Steve Jobs, confirmed what rumors had been saying for some time already. Apple would release a SDK for the iPhone:

> *Let me just say it: We want native third party applications on the iPhone, and we plan to have an SDK in developers' hands in February. We are excited about creating a vibrant third party developer community around the iPhone and enabling hundreds of new applications for our users. [...] we believe we have created the best mobile platform ever for developers.*

The four-month delay of releasing the SDK can be traced to the fact that Apple was doing two seemingly opposing things at once. While attempting to provide an advanced and open platform to developers, they simultaneously tried to secure the iPhone from potential viruses, malware, and privacy attacks. One month delayed, the SDK was presented to invited media, analysts, and developers.

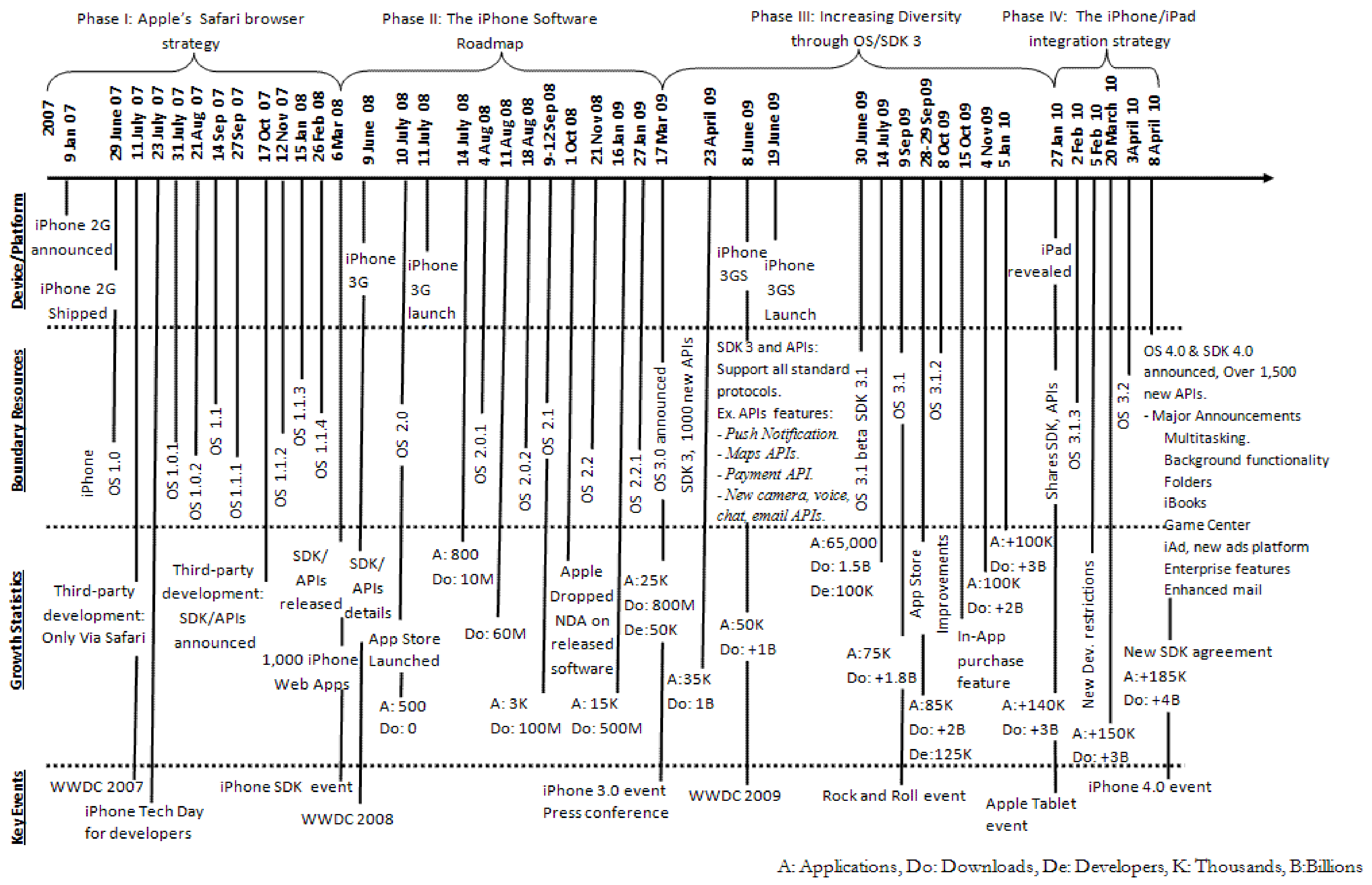

Figure 1. The iPhone platform

Phase II: The iPhone Software Roadmap (March 2008– March 2009)

This phase was initiated by the introduction of the '*iPhone Software Roadmap'* including a set of entirely new boundary resources:

SDK: The new SDK came with various components and features. First, the *Xcode development environment* was a source editor that enabled code completely to the APIs, project and source control management, and debugging. Second, the *interface builder* handled the graphical user interface, allowed rapid development and drag-n-drop, connected to library of iPhone controls and codes, as well as allowed for localization. Third, the *instruments* tool was a comprehensive analysis tool that allowed developers to check different performance aspects of their applications. Finally, the *iPhone simulator* tool simulated the entire range of APIs and facilitated testing applications on Mac computers.

APIs: In addition, Apple provided all their internally used iPhone's APIs (see Table 2 for an overview), covering different architectural layers of the iPhone operating system.

Table 2. iPhone OS layers and associated APIs

Layers	**APIs and layer components**
Core OS	Power management, security, TCP/IP sockets, file system, and so on.
Core services	Collections, address book, networking, file access, core location, net services, threading, URL utilities, and so on.
Media	Core audio, OpenAL, audio mixing, audio recording, video playback, PDF, Core animation, and so on.
Coco Touch	Multi-touch systems, localizations, web view, people and image picker, camera

Distribution channel: On July 10, 2008**,** the AppStore was introduced as and the exclusive distribution channel for users to search, browse, buy and download iPhone applications. There would be two types of applications: (1) Free applications, can be provided and downloaded for free, and (2) Paid applications, developers will get 70% of the sales revenues and the rest goes to Apple. Steve Jobs explained the idea of the App Store:

> *This is an application we've written to deliver apps to the iPhone. And we are gonna put it in every single iPhone with the next release of the software. And so our developers are gonna be able to reach every iPhone user through the App Store.*

Application review: As to ensure platform integrity, Apple set up an application review process. Each application had to be submitted to the App Store *Review Team,* assuring its compatibility with Apple's guidelines and rules. For instance, restricted application types included porn, bandwidth hogs, illegal content, malicious and applications that could cause 'unforeseen' problems. Moreover, voice over Internet protocol (VoIP) applications were only allowed over wireless LAN connections. Demonstrating that Apple was serious about these restrictions, the App Store *Review Team* rejected several applications. In fact, they even pulled out applications that were originally accepted for distribution via the App Store.

Technical Support: Apple also offered a complete set of technical resources, support, and access to pre-released software. It provided developers with a complete and integrated process for developing and distributing applications for iPad, iPhone, and iPod touch. There were two types of membership at 99$/year: individual and company.

Incentives: A 100 million USD fund was announced to invest in startups that created applications for the iPhone. This so-called iFund opportunity was announced by KPCB, a venture capital firm. KPCB described the motives:

> *KPCB's iFund is a $100M investment initiative that will fund market-changing ideas and products that extend the revolutionary new iPhone and iPod touch platform [...] Focus areas include location based services, social networking, mCommerce (including advertising and payments), communication, and entertainment.*

Having these boundary resources in place, thousands of third-party developers started to design applications for distribution through the App Store. In the beginning of 2009, there were 25,000 developed applications, 50,000 registered developers, and 800 million application downloads. During this phase, Apple released the second major version of their OS/SDK, at the same time as they released the second generation of the iPhone (3G). This release was then followed by five main OS/SDK updates coming with new features and enhancements to the platform, and support to the development community.

One important action taken by Apple was to drop the controversial iPhone developer non-disclosure agreement (NDA) on released software. According to an Apple press release, the NDA was primarily put in place since:

> *The iPhone OS includes many Apple inventions and innovations that we would like to protect, so that others don't steal our work. It has happened before. While we have filed for hundreds of patents on iPhone technology, the NDA added yet another level of protection. We put it in place as one more way to help protect the iPhone from being ripped off by others.*

However, the NDA created too much of a burden on third-party developers contributing to the iPhone's success. As a result, the agreement was replaced with a new one that did not cover released software. Although the new boundary resources, the mainstream developer was unsatisfied with Apple leashing various functionalities and eliminating their use. Such functionalities included (but were not limited to) *push notifications, background functionality, payment APIs, multitasking and Flash support*. The developer and tech-writer Hank Williams considered Apple prohibiting real innovation by setting up these restrictions:

> *Apple opened up the iPhone and introduced a software developer's kit (SDK)……… Unfortunately, some of the details we have discovered about the SDK will have a real impact on the ability of developers to innovate on the iPhone. The issue is that Apple has set a policy that third party application developers can't create applications that run in the background. ………….. Apple's tools are great, but they just make apps that were already possible, easier to build and easier to use. But as a developer, I want to create things that have been bottled up in my head but without the right platform, were fundamentally impossible to do.*

In order to satisfy the needs of the developers and maintain an externally flexible and internally coherent platform, Apple entered a new phase that provided major SDK/APIs releases and a set of new critical functionalities, tools, and policies.

Phase III: Increasing Diversity through OS/SDK 3 (March 2009- January 2010)

In view of significant criticism for their control strategy, Apple responded with a major and critical release of the third version of the OS/SDK. Available on the iPhone 3GS, the new release brought over 100 new features to the framework and 1,000 new APIs. Some of these releases were critical in that they corresponded to previous concerns of third-party developers including *push notifications, map APIs, support for serial I/O, and payment APIs.* Scott Forstall, the SVP of iPhone software at Apple, promoted the new APIs at his WWDC2009 keynote:

> *The SDK has more than a 1000 new APIs... let me talk you through a few…… For 3.0, we're opening up the possibility for developers to build apps that talk directly to third-party hardware……… Next is Maps. You can embed Google Maps into your applications. This is the heart of our app that we ship with the phone. You get all of the same functionality. And we're allowing developers to build turn-by-turn apps…….. Next, push, which is in iPhone OS 3.0… there are 3 types of notifications you can push, text alerts, numerical badges, and sounds alerts…. Ever since we announced 3.0, we've been seeding developers with betas, and here's a few to show you what they've accomplished [several App demos built on OS/SDK 3.0 and new APIs was promoted lively]… We think you'll love 3.0… it will be available June 17th… For developers, we're giving you the GM [Gold Master release] seed today.*

While Apple entered this phase with more than 25,000 developed applications [March 17, 2009], the number had doubled to 50,000 on the official release date of the new SDK/OS version [June 8, 2009]. Five weeks later, the number of applications had increased to 65,000. The number of registered developers doubled from 50,000 [March 17, 2009] to 100,000 [July 14, 2009]. This followed by releasing new major OS/SDK (3.1) and a set of sub-releases that enhanced the overall framework.

Due to the high number of applications (85,000), developers (125,000) and download rates (2 billion) [September 2009], Apple improved the App Store services in order to facilitate the promotion of applications submitted by all third-party developers, especially those who lacked marketing resources. Apple rolled out its "Genius" feature for recommending apps based on users' download history. In so doing, Apple assured its ecosystem was enticing for minor developers. The "Genius" feature was one step. The "Categories" features was clearly another step, which created an easy way for third-party developers based their applications around certain categories. The number of developed applications jumped to 100,000 and download rate to 3 billion [5 January 2010].

Apple regularly revisited their *application review and restrictions policies;* during this phase they introduced new restrictions and warned developers against adding restricted content and features to their submitted applications. For instance, one restriction concerned location-aware ads. As Apple's Dev Center declared:

> *If you build your application with features based on a user's location, make sure these features provide beneficial information. If your app uses location-based information primarily to enable mobile advertisers to deliver targeted ads based on a user's location, your app will be returned to you by the App Store Review Team for modification before it can be posted to the App Store.*

Phase IV: The iPhone/iPad integration strategy (Jan 2010 -)

On January 27, 2010, Apple announced the iPad, a tablet computer based on a modified version of the iPhone OS. Seeking to make this tablet compatible with all previously built iPhone applications, Apple introduced an enhanced SDK version that supported both the iPhone and the iPad. Scott Forstall, the SVP of iPhone software at Apple, highlighted this at Apple's special tablet event:

> *We've enhanced the iPhone SDK to now support development for the iPad as well and we are releasing this SDK today…… and this SDK even includes an iPad simulator….. You know, we think it's gonna be a whole of the gold rush for developers as they build apps for the iPad.*

The App Store was also enhanced to match this new strategy. As Forstall noted:

> *And of course every iPad comes with the App Store loaded upon it, so you have a great distribution channel to geek your apps out to all of our customers. Now, we are gonna take and highlight and feature all of the apps that built specially for the iPad in the store, you still be able to get to all the iPhone apps, but if you create an application specifically for the iPad we are gonna put it front and send it. We are really excited about the possibilities for developers on the iPad.*

The number of developed applications increased rapidly from 140,000 to 185,000 and download rate from 3 billion to 4 billion in less than three months. During this time period, Apple released two OS/SDK updates (3.1.3 and 3.2) that brought new enhancements of the platform and supported the iPad. This was immediately followed by the release of OS/SDK 4.0 [April 8, 2010] that entailed more than 100 new OS features and more than 1,500 new APIs. Steve Jobs commented:

> *We are giving a developer preview of OS 4, the next major release of the iPhone OS, we've been working on this for a while........ iPhone OS 4 delivers over 1,500 new APIs to developers, a lot of stuff that developers have been asking for*

The most significant features of this release were: multitasking, folders, enhanced mail, iBooks, enterprise features, game center and iAd. These features were responses third-party developer requests. Multitasking for instance, was supported by seven services that could give developers the ability to add various multitasking types to their applications.

Once these new features for increasing diversity were in place, Apple's attention was shifted to meta/cross platforms. Apple did not want any meta/cross platform to exist between the iPhone and third-party developers. According to analysts and hinted by Steve Jobs, the main reason behind this was that it would facilitate simultaneous development for competitors' platforms and so Apple will has less control over the iPhone ecosystem. This was achieved by banning SDKs and APIs to talk to such meta-platforms. This was admitted by Jobs in an email to a developer. He wrote:

> *We've been there before, and intermediate layers between the platform and the developer ultimately produces sub-standard apps and hinders the progress of the platform.*

Adobe Flash was one of the main and critical meta-platforms that Apple interdicted. Apple did not support Flash for neither web nor mobile-based applications. However, in October 2009, Adobe announced that a 'Packager for iPhone' applications would be included in the next version of its Flash developer tool, the Creative Suite 5 (CS5). This would automatically convert Flash applications into iPhone applications. While Flash applications still would not run on the iPhone, the CS5 simply turned Flash applications into iPhone applications automatically.

Despite Adobe's efforts to support their developers, and prior to the anticipated launch of CS5 on April 12, 2010, Apple released a new developer license agreement. In this new agreement, there was a nestled passage that had major implications for third-party developers, and disastrous consequences for Adobe's anticipated release of Flash CS5. Apple banned "*applications that link to Documented APIs through an intermediary translation or compatibility layer*". Section 3.3.1 in the iPhone SDK agreement reads:

> *Applications may only use Documented APIs in the manner prescribed by Apple and must not use or call any private APIs. Applications must be originally written in Objective-C, C, C++, or JavaScript as executed by the iPhone OS WebKit engine, and only code written in C, C++, and Objective-C may compile and directly link against the Documented APIs (e.g., Applications that link to Documented APIs through an intermediary translation or compatibility layer or tool are prohibited).*

It appeared as the ban directly applied to Adobe's new features and any other meta-platforms that would attempt to follow the same strategy for by-passing Apple's SDKs and APIs. This also gave Apple the possibility to decide which intermediary tools they would accept or reject. Another meta-platform that was not supported by the iPhone was Java, since Java applets running on the iPhone would also be outside the bounds of the SDK agreement. Section 3.3.2 reads:

> *An Application may not itself install or launch other executable code by any means, including without limitation through the use of a plug-in architecture, calling other frameworks, other APIs or otherwise. No interpreted code may be downloaded and used in an Application except for code that is interpreted and run by Apple's Published APIs and built-in interpreter(s).*

Discussion

In this paper, our objective is to develop an empirically grounded perspective of the process by which firms can govern third-party development through the boundary resources of software platforms. In what follows, we describe this process perspective by detailing its core concepts and underlying dynamics, supported by evidence from our case study. We then discuss the implications of the process perspective for the research and practice of third-party development, and conclude by presenting possible future research opportunities.

A Process Perspective of Governing Third-Party Development

Drawing on boundary objects theory (Bergman et al. 2007; Carlile 2002; Star and Griesemer 1989), there were three main boundary resources that Apple used to govern the third-party development in the iPhone community: SDK, APIs, and the developers' agreement. These resources served as important means with which Apple remained in control, yet stimulated the mobilization of new knowledge resources in the form of developers targeting new market segments. This dual activity is at the core of the challenge for platform owners engaging in third-party development. First, APIs served as *repository* boundary objects by working as a developers' reference point that supplied specifications, data structures, object classes, and protocols, and so on. They helped developers solve immediate problems by providing them with shared definitions and values. APIs also served as *coincident* boundary objects as they offered third-party developers a common referent for using their different perspectives independently. Second, the SDK served as an *ideal* boundary object as it provided support across different functional settings without accurately describing the details of specific applications. And, finally, developers' agreements shaped as *standardized form* boundary objects by being devised as methods of common communication across dispersed actors: third-party developers, Apple and their SDK and APIs.

We posit that governing third-party development can be seen as cyclical pattern of actions across platform boundary resources. This cyclical pattern is driven by the platform owner and its responses to changes in the emerging innovation network. The platform owner constructs new platform designs, secures platform control through agreements compatibility, increases knowledge heterogeneity through distribution channels, and counteracts foreign boundary resources designed to infringe on the platform. Figure 2 summarizes our process perspective and illustrates key items and relationships (cf. Langley 1999).

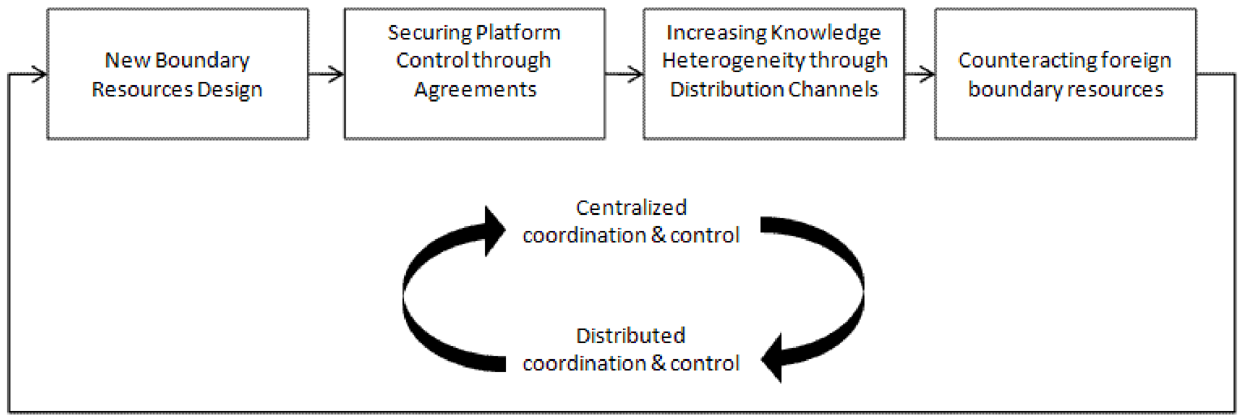
New Boundary Resources Design
Securing Platform Control through Agreements
Increasing Knowledge Heterogeneity through Distribution Channels
Counteracting foreign boundary resources
Centralized coordination & control
Distributed coordination & control

Figure 2. A process perspective of governing third-party development

New boundary resources design. A new cycle of third-party development begins when the platform owner recognizes insufficiency in current boundary resources. Such recognition is usually triggered by feedback from the community of third-party developers. It may also result from attempts by competitors to introduce foreign boundary resources. Responding to the current boundary resources' perceived insufficiency, the platform owner attempts to improve the boundary resources and/or implement new ones to increase the proficiency of the innovation network. In the first phase, Apple, for example, recognized the insufficiency of Safari as the only approach to third-party development. Triggered by community feedback, Apple decided to release an SDK and APIs that allowed third-party developers to develop and port applications directly to the platform. Similarly, in the second phase, after releasing the initial SDK (OS/SDK 2.0), Apple perceived the insufficiency of these the released boundary resources, and shifted to a new improved and enhanced design. In the third phase, after releasing a new SDK (OS/SDK 3.0) and more than 1,000 new APIs, Apple recognized the insufficiency of such boundary resources. Lastly, in the fourth phase, Apple released a new SDK (OS/SDK 4.0) with more than 100 new features and over 1,500 new critical APIs to meet the needs of developers.

Securing platform control through agreements. The design of new boundary resources is followed by revisiting platform-associated agreements intended to secure platform control. Existing agreements and rules can be canceled, modified, and/or new ones can be issued. Our study shows that agreements, and their compatibility with new introduced design, are critical for the platform owner. They ensure that third-party developers do not exploit weaknesses in boundary resources. In the first phase, Apple did not require any SDK agreement, since the platform was already protected by only allowing web-based development. In the second phase, Apple released a non-disclosure agreement (NDA) for released software, which maintained their new platform design (OS/SDK 2.0). In view of severe criticism, Apple released a new SDK agreement without NDA on released software. In the third phase, the SDK agreement was slightly updated to maintain platform control in view of all new APIs released. In the fourth phase, a major update of the SDK agreement was done, interdicting any alien boundary resources attempting to create a new layer between the platform and third-party developers.

Increasing knowledge heterogeneity through distribution channels. Innovation networks rely on heterogeneity in knowledge resources (Yoo et al. 2008, 2009). A platform's distribution channel may be used leverage such resources by expanding its reach to a wider user community. In the case of Apple, the distribution channel has expanded over time. During the first phase, the distribution channel was the Safari web browser. In the second phase, Apple introduced the App Store as the exclusive distribution channel of iPhone applications. It provided the possibility for end-users to browse, search, view, download, and rate applications. In the third phase, the App Store was enhanced with multiple features to better cater for the new version of OS/SDK 3.0 and its dramatic increase of applications. For instance, the 'Genius' and 'Categories' features supported third-party developers and their application distribution to the huge number of platform users. In the fourth phase, a new OS/SDK was introduced that supported both the iPhone and the iPad platforms. In response, Apple enhanced the App Store, so it could support both platforms with identical boundary resources.

Counteracting foreign boundary resources. A platform owner needs to counteract foreign boundary resources over time to avoid that the platform avoids getting infringed by competitors. This is a delicate governance issue for the platform owner, since meta/cross platforms typically exist for making some third-party software run well on the core platform. Ultimately, it is the core platform's proprietor to decide whether to enable the intermediation of such external platforms or not. Allowing such intermediation increases the impact of the platform, but this impact is gained at the expense of more distributed platform control. Apple decided not to allow meta/cross platforms, although third-party developers regularly expressed their concerns. In the first phase, Apple allowed third-party developers to use Web 2.0 and Ajax for the development of their web-based applications. Flash was totally blocked, meaning that users could not view flash in their Safari web browser. In the second phase, they decided to keep Flash and other meta/cross platforms interdicted; they decided not to allow third-party developers to use such functionality on Apple's platform. In the third phase, Apple maintained this position. Some meta/cross platforms invented ways to overcome Apple's interdiction and rules. For instance, Adobe announced that the firm would bring Flash to the iPhone. They would have allowed third-party developers to use Adobe's tools and simply turn Flash applications into iPhone applications automatically. In response, Apple updated the SDK agreement and rules to prevent Adobe's competitive move.

The dynamics of third-party development. Looking back at the case, it is clear that controlling a platform over time requires successful balancing of the control efforts and the need to incorporate new knowledge resources. Apple intended to exercise centralized control over the iPhone innovation network (cf. Yoo et al. 2008). However, already early on, Apple had to reconsider its initial web browser strategy and release a SDK and a set of APIs as boundary resources for third-party developers. Yet, the firm actively governed the iPhone network for staying in control over the rapidly growing innovation network. Perhaps the most extreme example of this control ambition was interdicting Adobe's solution for making Flash work on the iPhone.

Our process perspective suggests that governing third-party development involves a series of actions that is repeated over time. Using boundary resources as the main mediator, each repeated series of actions involve both acts for keeping control and for stimulating external contributions.

Implications

There are a number of implications of our process perspective. First, it complements and extends the literature on open source governance (Dahlander and Magnusson 2005; de Laat 2005; Demil and Lecocq 2006; Markus 2007; O'Mahony 2005; West 2003) by detailing governance in the context of platform ownership. Our process model shows that governance involves a delicate balance act of the platform owner, trying to keep control of the platform while simultaneously seeking to expand the diversity of potential developers.

Second, our process model presents new insights about the process by which boundary resources are used to transfer design capability to application developers. We depict a process including four inter-related steps: (1) the development of new boundary resources, (2) securing platform control through agreements, (3) increasing knowledge heterogeneity through distribution channels, and (4) counteracting foreign boundary resources. Platform owners need to pay heedful attention to the control and coordination mode of the innovation network, since each change in boundary resources has potential impact for the possibility to attract new type of knowledge resources and to control the platform. In this regard, our process perspective contributes to the body of knowledge developed around von Hippel's seminal contributions around using tools for user innovation (Franke and von Hippel 2003; von Hippel 2001).

Finally, we contribute to the continued investigation of digital innovation (Henfridsson et al. 2009; Svahn et al. 2009; Yoffie 1997; Yoo et al. 2008) by illustrating how business strategy and software platforms are interweaved. The received view in the IS discipline makes a clear distinction between business strategy and IT strategy. Clearly, boundary resources are means with which to exercise strategy in both domains.

Conclusion

In this paper, we synthesized the boundary object perspective (Bergman et al. 2007; Star and Griesemer 1989) and innovation networks literature (Yoo et al. 2009) to study the governance of third-party development as an alternative form of systems development. Based on an extensive and detailed case study research of Apple's iPhone innovation ecosystem, we developed an empirically grounded process perspective of governing third-party development through platforms boundary resources. In particular, we depict how this involves a delicate balance between platform control and stimulation of application development by third-party developers.

Future studies could address several limitations in our work. First of all, this paper draws on a single case study, where Apple has been tremendously successful in stimulating external contributions. It would be useful to compare our results with investigations of other platform owners' attempts to attract third-party developers to build strong innovation networks. Second, while process theorizing provides a detailed and situated view of the research phenomenon, future studies could adopt a variance perspective that would develop a causal model for testing different boundary resource strategies in terms of growth of applications, applications developers, and knowledge heterogeneity. Finally, another direction for future work would be to investigate governance through boundary resources from the perspective of third-party developers.

References

Allen, B.R. and Boynton, A.C. "Information Architecture: In Search of Efficient Flexibility," *MIS Quarterly*, (15:4), December 1991, pp.435-445.

Baldwin, C.Y. and Clark, K.B. *Design Rules - The Power of Modularity*. MIT Press, Cambridge, MA, 2000.

Benkler, Y. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press. 2006.

Bergman, M., Lyytinen, K. and Mark, G. "Boundary Object in Design: An Ecological View of Design, " Journal of the AIS, (8:11), 2007, PP.546-568.

Boland, R. J., Lyytinen, K., & Yoo, Y. "Wakes of innovation in project networks: The case of digital 3-D representations in architecture, engineering, and construction," *Organization Science*, (18:4), 2007, pp.631–647.

Briers, M. and Chua, W. F. "The role of actor-networks and boundary objects in management accounting change: a field study of an implementation of activity-based costing," *Accounting, Organizations, and Society,* (26:3), April 2001, pp. 237- 269.

Brown, C. "Examining the emergence of hybrid IS governance solutions: evidence from a single case site," *Information Systems Research*, (8:1), March 1997, pp.69–94.

Carlile, P. "A pragmatic view of knowledge and boundaries: Boundary objects in new product development," *Organization Science*, (13:4), July/August 2002, pp. 442-455.

Charmaz, K. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. Thousands Oaks, CA: SAGE Publications, 2006.

Chesbrough, H., Vanhaverbeke, W., West, J. *Open Innovation: Researching a New Paradigm*, Oxford University Press, Oxford, 2006.

Clark, K.B. "The Interaction of Design Hierarchies and Market Concepts in Technological Evolution," *Research Policy,* (14), 1985, pp 235-251.

Creswell, J. W. *Qualitative inquiry and research design: choosing among five traditions*, 2nd ed., Sage Publications, Thousand Oaks, Calif., IS, 2007.

Dahlander, L., & Magnusson, M. G. "Relationships between open source software companies and communities: Observations from Nordic firms". *Research Policy*, (34:4), 2005, pp.481–493.

Dixon, P.J. and John D.A. "Technology Issues Facing Corporate Management in the 1990s," *MIS Quarterly*, (13:3), September 1989, pp.247-255.

de Laat, P. B. "Copyright or copyleft?: An analysis of property regimes for software development", *Research Policy*, (34:10), 2005, pp.1511–1532.

Demil, B., & Lecocq, X. "Neither market nor hierarchy or network: The emergence of bazaar governance", *Organization Studies*, (27:10), 2006, pp.1447–1466.

Evans, D.S., Hagiu, A. and Schmalensee, R. *Invisible Engines: How Software Platforms Drive Innovation and Transform Industries*, Cambridge MA: MIT Press, 2006.

Ferguson, C. H. "What's Next for Google?," *MIT Technology Review*, (108:1), January 2005, pp. 38-46.

Fitzgerald, B. "The Transformation of Open Source Software," *MIS Quarterly* (30:3), 2006. pp. 587-598.

Franke, N., and von Hippel, E. "Satisfying Heterogeneous User Needs Via Innovation Toolkits: The Case of Apache Security Software," *Research Policy* (32:7), 2003, pp. 1199-1215.

Garud, R. Jain, S. and Kumaraswamy, A. "Institutional entrepreneurship in the sponsorship of common technological standards: The case of Sun Microsystems and Java," *Academy of Management Journal*, (45), 2002, pp. 196-214.

Gawer, A., and Cusumano, M. *Platform leadership: How Intel, Microsoft and Cisco drive industry innovation*, Boston: Harvard Business School Press, 2002.

Gawer, A., and Cusumano, M. "How Companies Become Platform Leaders," *MIT Sloan Management Review*, (49:2), Winter 2008, pp.28-35.

Henfridsson, O., Yoo, Y., and Svahn, F. "Path Creation in Digital Innovation: A Multi-Layered Dialectics Perspective," *Sprouts: Working Papers on Information Systems* http://sprouts.aisnet.org/9-20 (9:20), 2009.

Langley, A. "Strategies for Theorizing from Process Data," *Academy of Management Review,* (24:4), 1999, pp.691-710.

Lee, G.K., Cole, R.E., "From a firm-based to a community based model of knowledge creation: the case of the Linux kernel development," *Organization Science,* (14:6), 2003, pp. 633–649.

Ljungberg, J. 2000. "Open Source Movements as a Model for Organizing," *European Journal of Information Systems,* (9:3), pp 208-216.

Markus, L.M. "The Governance of Free/Open Source Software Projects: Monolithic, Multidimensional, or Configurational?," *Journal of Management and Governance,* (11:2), 2007, pp. 151-163.

Messerschmitt, D.G., Szyperski, C. *Software Ecosystem: Understanding an Indispensable Technology and Industry*, MIT press, 2003.

Meyer, M. H. and Seliger, R. "Product platforms in software development," *MIT Sloan Management Review*, (40:1), Fall 1998, pp.61-74.

Mockus, A., Fielding, R. T., & Herbsleb, J. D. "Two case studies of open source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology*, (11:3), 2002, pp.309–346.

O'Mahony, S. *Nonprofit foundations and their role in community-firm software collaboration*. In J. Feller, B. Fitzgerald, S. A. Hissan & K. R. Lakhani (Eds.), Perspectives on free and open source software (pp. 393–413). Cambridge, MA: The MIT Press. 2005.

Ostrom, E. *Understanding Institutional Diversity*. Princeton, NJ: Princeton University Press. 2005.

Raymond, E. S. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly and Associates, Inc., Sebastopol, CA, 1999.

Robertson, D. and Ulrich, K., "Planning for product platforms." *Sloan Management Review*, Summer 1998, pp.19–31.

Romano, N. C., Donovan, C., Chen, H., & Nunamaker, J. F. "A methodology for analyzing web-based qualitative data," *Journal of Management Information Systems*, (19:4), 2003, pp. 213-246.

Scacchi, W. "Understanding the requirements for developing open source software systems," *IEE Proceedings–Software*, (149:1), 2002, pp.24–39.

Seale, C. *The quality of qualitative research*, Sage Publications, London; Thousand Oaks, Calif., 1999.

Soy, S. K. "The case study as a research method," from University of Texas, Graduate School of Library and Information Science Website, 1996. Available at: http://www.gslis.utexas.edu/~ssoy/usesusers/l391d1b.htm. [retrieved 14 Jan 2010].

Star, S. L. and Griesemer, J. R. "Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39," *Social Studies of Science*, (19:3), August 1989, pp. 387-420.

Svahn, F., Henfridsson, O., and Yoo, Y. "A Threesome Dance of Agency: Mangling the Sociomateriality of Technological Regimes in Digital Innovation," in: *Proceedings of ICIS 2009*. Phoenix, AZ, 2009.

Tapscott D. and Williams A.D. "Wikinomics: How Mass Collaboration Changes Everything," New York (N.Y.): Portfolio, 2006.

Taudes A., Feurstein M., and Mild A., "Options Analysis of Software Platform Decisions: A Case Study," *MIS Quarterly*, (24:2), June 2000, pp. 227-243.

Teece, D.J. "Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing and Public Policy," *Research Policy,* (15), 1986, pp 285-305.

Van de Ven, A.H. 2005. "Running in Packs to Develop Knowledge-Intensive Technologies," *MIS Quarterly,* (29:2), pp 365-378.

Van de Ven, A. H., Polley, D., Garud, R., and Venkatraman, S. "The Innovation Journey," New York: Oxford University Press, 2008.

Van Grembergen, W., De Haes, S., and Guldentops, E. "Structures, Processes and Relational Mechanisms for IT Governance," *in Strategies for Information Technology Governance*, Idea Group, London, 2004.

Von Hippel, E. "Perspective: User Toolkits for Innovation," *The Journal of Product Innovation Management*, 18, 2001, pp. 247-257.

Von Hippel, E. and R. Katz, “Shifting Innovation to Users via Toolkits,” *Management Science*, (48:7) ,June 2002, pp. 821-834.

Von Hippel, E., and Von Krogh, G. "Open Source Software and The "Private-Collective" Innovation Model: Issues for Organization Science," *Organization Science,* (14:2), 2003, pp. 209-223.

von Krogh, G., and Von Hippel, E. "The Promise of Research on Open Source Software," *Management Science* (52:7), 2006, pp 975-983.

Weill, P. and Ross, J.W. “A Matrixed Approach to Designing IT Governance,” *MIT Sloan Management Review,* (46:2), 2005, pp. 26-34.

West, J. "How Open Is Open Enough? Melding Proprietary and Open Source Platform Strategies," *Research Policy,* (32), 2003, pp. 1259-1285.

West, J., and Gallagher, S. 2006. "Challenges of Open Innovation: The Paradox of Firm Investment in Open-Source Software," *R&D Management,* (36:3), pp 319-331.

Yoffie, D.B. "Introduction: Chess and Competing in the Age of Digital Convergence," in: *Competing in the Age of Digital Convergence*, D.B. Yoffie (ed.). Boston, MA: Harvard Business School Press, 1997, pp. 11-35.

Yin, R.K. *Case Study Research: Design and Methods*, (4 ed.). Thousand Oaks: SAGE Publications, 2009.

Yoo, Y., Lyytinen, K., and Boland, R.J. “Distributed Innovation in Classes of Networks,” Proceedings *of the Fourty-First Annual Hawaii International Conference on System Sciences* (CD-ROM), January 7-9, Computer Society Press, 2008.

Yoo, Y., Lyytinen, K., and Boland, R. J. “Innovation in the Digital Era: Digitization and Four Classes of Innovation Networks,” Working Paper, Temple University, 2009.

Zmud, R.W., Boynton, A.C., and Jacobs, G.C. “The Information Economy: A New Perspective for Effective Information Systems Management,” *Data Base*, (18:1), 1986, pp.17-23.