doi:10.1111/j.1365-2575.2012.00406.x

Balancing platform control and external contribution in third-party development: the boundary resources model

Ahmad Ghazawneh* & Ola Henfridsson†

*Informatics Department, Jönköping International Business School, SE-551 11, Jönköping, Sweden, email: ghah@ihh.hj.se, and †Department of Applied Information Technology, Chalmers University of Technology, SE-412 96, Gothenburg, Sweden, email: ola.henfridsson@chalmers.se

Abstract. *Prior research documents the significance of using platform boundary resources (e.g. application programming interfaces) for cultivating platform ecosystems through third-party development. However, there are few, if any, theoretical accounts of this relationship. To this end, this paper proposes a theoretical model that centres on two drivers behind boundary resources design and use – resourcing and securing – and how these drivers interact in third-party development. We apply the model to a detailed case study of Apple's iPhone platform. Our application of the model not only serves as an illustration of its plausibility but also generates insights about the conflicting goals of third-party development: the maintenance of platform control and the transfer of design capability to third-party developers. We generate four specialised constructs for understanding the actions taken by stakeholders in third-party development: self-resourcing, regulation-based securing, diversity resourcing and sovereignty securing. Our research extends and complements existing platform literature and contributes new knowledge about an alternative form of system development.*

Keywords: platform, third-party development, boundary resources model, application programming interfaces (APIs), resourcing, securing

INTRODUCTION

The last few years have witnessed a significant increase in the frequency and magnitude of third-party development (Yoo *et al*., 2010). The difficulty to make informed decisions about which applications, systems or services to develop (Henfridsson & Lindgren, 2010), and how they should be designed to work as effective responses to the information processing requirements of an accelerated and turbulent market (Mathiassen & Sørensen, 2008), make the involvement of third-party application developers increasingly attractive for software platform

owners (Bosch, 2009; Boudreau & Lakhani, 2009). The prime example, serving as a role model for many followers, is Apple's iPhone platform (Kenney & Pon, 2011).

We define third-party development as a type of systems development where one actor, the third-party developer, on behalf of someone else, the platform owner, develops applications, services or systems (hereafter referred to as applications) for satisfying end-users of the platform. Third-party development tends to be governed by arm's-length, contractually oriented relationships (Boudreau & Lakhani, 2009). The third-party application developer is seldom directly compensated for the development work put in. Instead, the developer is offered a marketplace for its applications with greater reach than otherwise would be available to the single actor (West & Mace, 2010). The incentive for the platform owner is the possibility to diversify its offer to customers through voluminous development across a range of application areas (Boudreau, forthcoming). The platform owner taps into multiple networks of developers, characterised by heterogeneous innovation capability and knowledge resources (Boland *et al*., 2007; Yoo *et al*., 2010). In addition, if successful, the platform owner can set up a revenue sharing business model where a specific portion of the revenue is withheld as a compensation for the distribution and support of the applications (West & Mace, 2010).

To successfully build platform ecosystems (El Sawy *et al*., 2010; Selander *et al*., 2010), the focus of the platform owner must shift from developing applications to providing resources that support third-party developers in their development work (Prügl and Schreier, forthcoming). We refer to such resources as platform boundary resources, i.e. the software tools and regulations that serve as the interface for the arm's-length relationship between the platform owner and the application developer. Platform boundary resources are imperative to 'transfer design capability to users' (von Hippel & Katz, 2002, p. 824), which, in turn, is intended to generate complementary assets in the form of applications (cf. Teece, 1986). Still, we know that boundary resources also are designed for controlling the platform and the ecosystem that evolves from its use in application development and deployment (Ghazawneh & Henfridsson, 2010). In this regard, there is a delicate tension in boundary resource design between maintaining platform control and, at the same time, stimulating third-party developers to join forces with the platform owner by developing applications.

The research question addressed in this paper is: *How can we understand the role of boundary resources in platform owners' efforts to cultivate third-party development?* Prior research documents the significance of using platform boundary resources such as application programming interfaces (APIs) for cultivating platform ecosystems through third-party development. However, there are few, if any, theoretical accounts of this relationship. For instance, even though Ghazawneh & Henfridsson (2010) provide a process perspective of boundary resources and third-party development, it lacks the coherence and analytical distinctiveness expected in theory development based on case study research (cf. George & Bennett, 2005). To this end, this paper proposes a theoretical model including clearly defined constructs with significant implications for future research on third-party development and platform ecosystems. We apply the model to a detailed case study (George & Bennett, 2005; Gerring, 2007) of Apple's iPhone platform to illustrate the model. We also generate four specific modes of simultaneously addressing the needs of platform control and external contribution.

BOUNDARY RESOURCES IN THIRD-PARTY DEVELOPMENT

There is little doubt that software platforms lay at the heart of third-party development. Following Tiwana *et al*. (2010, p. 676), we refer to *software platforms* as 'the extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they operate'. In this regard, platforms involve the development of common resources from which to generate derivative applications and services (Robertson & Ulrich, 1998).

Previous platforms research pays little attention to external contributions in the development of derivative applications (see e.g. Robertson & Ulrich, 1998). Originating in product development research, platform design has been viewed as a strategy for combining scale economics and product differentiation at the same time. For instance, consider the title of Meyer & Lehnerd's (1997) seminal book on the topic, which underlines corporate value building and cost leadership as the main elements in explaining the power of platforms. In fact, the received literature views platforms as the basis for product portfolios, serving different market needs but still building on a common base of standardised components. Over the years, software product line engineering has emerged along the same line of thinking (Bosch, 2009; Pohl *et al*., 2005).

More recent platforms research paints a somewhat different picture. It is increasingly recognised that external application developers may play a significant role in platform innovation (Messerschmitt & Szyperski, 2003; Evans *et al*., 2006; Bosch, 2009; Boudreau, forthcoming) and serve as the basis for software leadership (Gawer & Cusumano, 2002). In this stream of research, it is noted that the wealth of derivative applications needed to stay competitive is difficult to accomplish in-house. In fact, dealing with the volatile information processing requirements of an accelerated and turbulent market (Mathiassen & Sørensen, 2008) requires an approach encompassing lean interfaces between the platform and the applications generated from its extensible codebase (Yoo *et al*., 2010). Given its arm's-length, contractually oriented relationships (Boudreau & Lakhani, 2009), third-party development offers a mode of systems development that enables the leanness required for incorporating the ideas of developer communities in cultivating digital ecosystems.

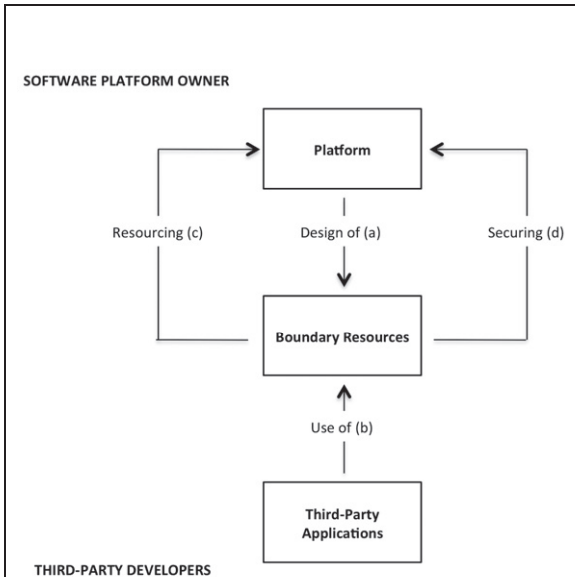
To enable third-party development, design capability needs to be shifted to external actors (von Hippel & Katz, 2002; Prügl & Schreier, forthcoming) who are capable to serve end-users through application development. We refer to *applications* as executable pieces of software that are offered as applications, services or systems to end-users of the platform. At the interface between the platform owner and third-party developers, platforms may offer resources with which to facilitate the use of core platform functionality to build applications. As noted in the introduction section, we refer to these resources as platform *boundary resources*, i.e. the software tools and regulations that serve as the interface for the arm's-length relationship between the platform owner and the application developer. In software platform settings, such resources typically consist of a software development kit (SDK) and a multitude of related APIs. The power of such resources is that they give easy access to core modules of the platform, stimulating generativity, i.e. 'a capacity to produce unprompted change driven by large, varied and uncoordinated audiences' (Zittrain, 2006, p. 1980). For example, the release

of an API for facilitating the use of the global positioning functionality of a smartphone platform would provide the capacity to produce location-based services across multiple and uncoordinated third-party developers.

The boundary resources model

We propose a model of boundary resource design in third-party development. Even though the role of platform boundary resources has been highlighted in previous research (Ghazawneh & Henfridsson, 2010), little attention has so far been paid to developing a theoretical account that provides a coherent basis for further research in the area. In this regard, this paper embraces Yoo *et al*.'s (2010) call for more grounded research on the role of boundary resources in digital innovation by looking closer at the special case of third-party development.

Figure 1 depicts the main constructs of the boundary resources model and their relationships. *Boundary resources design* refers to the platform owner's act of developing new, or modified, boundary resources as a response to external contribution opportunities and control concerns perceived by the software platform owner. Boundary resources design is typically initiated when a platform owner recognises that existing boundary resources are insufficient for developing the platform, including its applications, in a favourable way. First, boundary resources may be designed to empower the community of third-party developers and their



CONSTRUCTS

Platform: "The extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they operate" (Tiwana *et al.* 2010, p. 676)

Boundary Resources: The software tools and regulations that serve as the interface for the arm's-length relationship between the platform owner and the application developer

Third-Party Applications: Executable pieces of software that are offered as applications, services, or systems to end-users of the platform

Boundary Resources Design: The platform owner's act of developing new, or modified, boundary resources as a response to perceived external contribution opportunities and control concerns

Boundary Resources Use: The third-party developer's act of developing end-user applications drawing on boundary resources offered by the software platform owner

Resourcing: The process by which the scope and diversity of a platform is enhanced

Securing: The process by which the control of a platform and its related services is increased

Figure 1. The boundary resources model.

possibility to extend the platform with applications. The recognition of such external contribution opportunities may be anticipated or triggered by feedback from third-party developers. Second, boundary resources may also be (re)designed to address control concerns. Such control concerns may emerge when third-party developers launch, or announce the intention to launch, applications that represent potential threats to the platform.

We refer to *boundary resources use* as the third-party developer's act of developing end-user applications drawing on boundary resources offered by the software platform owner. Third-party developers use boundary resources such as APIs for utilising the platform's capabilities in developing and deploying applications that serve end-users' needs. Some boundary resources may be mandatory to use. However, most boundary resources are optional in the application development and their use depends on the third-party developers' design choices in seeking to serve their customers, the end-users.

In cases where boundary resources are designed in response to external contribution opportunities and third-party developers find them useful, the platform benefits from heterogeneity in knowledge resources (Yoo *et al*., 2010). Boundary resources then increase such heterogeneity by providing access to new resources of the digital platform in question. We refer to *resourcing* as the process by which the scope and diversity of a platform is enhanced. This process builds on third-party developers' generation of applications with new types of functionality. For instance, the introduction of a new digital hardware feature makes it possible to design a new boundary resource that provides access to that feature in the third-party developer's application development. Resourcing therefore typically helps expanding the ecosystem of actors around the platform and therefore secures the supply of new resources, knowledge and capabilities (Van de Ven, 2005).

The community of third-party developers typically involves multiple and heterogeneous actors, each pursuing their interests (Van de Ven, 2005; Boland *et al*., 2007; Yoo *et al*., 2010). While boundary resources are imperative in cultivating that heterogeneity, they are simultaneously used to address control concerns through a process that we refer to as securing. *Securing* denotes the process by which the control of a platform and its related services is increased. Typically, this process prevents the development of applications that risk infringing the platform. For instance, cancelling or modifying existing developer agreements, and/or issuing new ones may be ways to secure the platform. Securing typically helps establishing a control level for the platform and its community of third-party developers. In this regard, control levels can vary along a continuum ranging from the quite centralised control found in the traditional industry sectors such as the automotive industry (Henfridsson *et al*., 2009), to the relatively decentralised control that would be found in open source software settings.

The boundary resources model provides an intellectual structure with which to understand the role of platform boundary resources design and use in third-party development. The logical formulation of the model follows that of process theory, which means that 'causation consists of necessary conditions in sequence' (Markus & Robey, 1988, p. 590) and that the absence of such necessary conditions will hinder the outcome from occurring. For instance, if a platform owner would not develop boundary resources, third-party developers can not use boundary resources to develop third-party applications. Contrary to variance theories,

however, causation is contingent, meaning that even with a necessary condition present, the outcome may not occur. For example, even if a platform owner would design boundary resources, there is no guarantee that third-party developers would use them for developing applications. Concurring with well-known theories in information systems (IS) such as the structurational model of technology (Orlikowski, 1992), the causality type of the boundary resources model is bidirectional, i.e. it exhibits feedback mechanisms and mutual shaping (cf. Langley, 1999). For instance, the emergence of new boundary resources can be initiated both by the platform owner's perception of external contribution opportunities and by third-party developer's use of boundary resources. Essentially, the platform owner's design of boundary resources can be both proactive and reactive.

METHODOLOGY

We conducted detailed case study research (George & Bennett, 2005; Gerring, 2007) of Apple's iPhone platform since its inception. Using multiple data sources, we adopted Romano *et al*.'s (2003) methodology for analysing internet-based qualitative data to sensitise the boundary resources model. Compared with social sciences such as economics and political science, the use of secondary data in case study research is unusual in IS. It represents a mode of research that differs from the traditional qualitative study where the bulk of data typically is collected first-handedly by the researcher.

While a typical concern would be a perceived distance between the researcher and the context in which the data originate, an important motivation for relying on secondary data collection is the large volumes of data that would be impossible using data collection techniques such as the qualitative interview (Romano *et al*., 2003). In addition, it provides a perspective that covers key stakeholders, whose input often is necessary for sensitising why particular initiatives were taken as a response to environmental changes (cf. Hargadon & Douglas, 2001).

Case selection

The selection of Apple's iPhone platform as the single case for our empirical investigation reflects the use of an extreme case selection technique (Yin, 2009). Extreme cases typically correspond 'to a case that is considered to be prototypical or paradigmatic of some phenomenon of interest' (Gerring, 2007, p. 101). As Gerring (2007) argues, such cases are useful for theory generation because extremes or ideal types typically define theoretical concepts. Compared with a representative case selection technique, our objective is to present ideal types, 'formed by the one-sided accentuation of one or more points of view' (Weber, 1949, p. 90). In this regard, the model offers conceptual constructs that manifest theorising through idealisation (Lopreato & Alston, 1970) rather than theorising intended to be valid across many cases.

There are at least two reasons why Apple's iPhone platform is suitable for this type of theorising. First, the platform's short and so far successful history (see Appendix for data on the tremendous growth of applications, developers and downloads) involves a number of transitions in the strategy by boundary resources were established by Apple and used by third-party developers. This allowed us to isolate specific episodes of third-party development for tracing underlying patterns of each episode. This enabled so-called temporal decomposition, which is important to establish process variance in process studies (Langley, 1999). Second, there exist substantial amounts of publicly available data on Apple's platform, making detailed study of its nature possible.

Data collection and analysis

As suggested above, our data collection reminds of historical studies where insights from first-hand observation can not be obtained (Kieser, 1994; Hargadon & Douglas, 2001). However, it should be noted that secondary data collection avoids possible biases introduced in real-time case studies where different forms of impression management on behalf of the stakeholders may distort collected data about ongoing events. As Silverman (2006) points out, the data collected through the open-ended interview should preferably be seen as mutually constructed by the researcher and the respondent, rather than as fact-gathering where respondents' views are treated as representations of reality. Similarly, our data collection and analysis should be seen as a hermeneutic process involving iterations between the researchers' emerging sensemaking and the data material collected from various sources (Klein & Myers, 1999).

Without the possibility to create new knowledge through focused interaction between researcher and respondent (cf. Denzin, 1970), however, the historical researcher needs to create a similar sense of the research setting by relying on the sensitising power of secondary data. As in any qualitative study, the researcher relying on such data should avoid taking on a journalist role, reporting about what is new but not necessarily providing rich insights that build on an in-depth account (cf. Silverman, 2006). Secondary data collection is powerful for building the extensive and longitudinal database needed for contextualisation of the historical background and plot of the research setting. Such contextualisation is an important element in interpreting data in interpretive research (Klein & Myers, 1999).

Data from six primary sources informed this research. The data sources included Apple press releases, news and announcements; archival data; conference, workshop and special events data; e-mails; interviews; and online articles. First, we collected all Apple's press releases, news and announcements related to the iPhone platform and third-party development. One important objective of collecting this data was to establish a timeline of key events that occurred over time and support the decomposing of the process into three episodes for structuring the analysis (see Appendix for detailed timelines). Moreover, these data were important for gathering Apple's official data on the number of downloads, applications and third-party developers at different points in time. Second, we collected most, if not all, publicly available case documents for understanding the boundary resources and their change over

Table 1. Data analysis

Steps	Tasks	Outputs
Elicitation	Elicited data from the six data sources	Research database
	Initial coding	Numerous concepts
Reduction	Used the boundary resources model to reduce and code the massive data material	Timeline of events
	Traced the historical process	Identified a sequence of events repeated over time
Visualisation	Identified and visualised three case episodes	Three case episodes
	Used the boundary resources model to analyse the case findings	The boundary resources model Four specialised constructs

time. This data collection included different versions of the Registered iPhone Developer agreement, iPhone human interface guidelines and SDK agreements. Furthermore, we collected videos and transcriptions documenting keynotes, speeches and special events, mainly including Apple officials. Since anyone who has seen Steve Jobs at Apple's developers' conference, Apple Worldwide Developers Conference (WWDC), will note that almost everything has been carefully orchestrated,[1] this data source mainly served as a documentation of Apple initiatives and how the firm officially framed these initiatives.

Fourth, the e-mail data source included more than 15 messages between Apple and developers, as well as developers and the public. As these messages were typically published by developers on their website, they often made a point that not necessarily was in Apple's interests. Typically, they provided a counterview to Apple's official line. Fifth, our case database included four interviews of Apple executives, conducted by respected journalists at, e.g. *Time* magazine and *Business Week*, and one interview of a notable iPhone developer. Lastly, more than 480 online articles were collected. These data generated a comprehensive understanding of the case and was particularly useful for providing multiple interpretations of single events.

We adopted Romano *et al*.'s (2003) data analysis method for making sense of the collected data material. This method provides a structured approach to analysing the rich, interesting and dynamic data existing on Apple's platform. Our data analysis was conducted as a three-step process: elicitation, reduction and visualisation (see Table 1). First, we used the six data sources to elicit relevant data to be included in our case database. Covering January 2007 to April 2010, our initial searches included keywords such as iPhone, AppStore and iPhone platform. Then, we engaged in intensive and carefully review of all collected data sources, where the initial coding (Charmaz, 2006) resulted in numerous concepts relating to categories such as third-party developers, SDK, API, platform governance, developers' community and iPhone ecosystem. The second step of our data analysis involved using the boundary resources model's constructs to reduce and focus the massive material collected. We also time-stamped data to help us trace the historical process and secure a correct timeline of

[1]We thank the Associate Editor for pointing this out.

events. Finally, we visualised the findings as a process decomposed into three episodes, which served as the backbone for structuring our illustration of the boundary resources model. The application of the model to each of the episodes generated four specialised concepts for understanding the actions taken by stakeholders in third-party development.

THE IPHONE PLATFORM CASE

The launch of the iPhone in June 2007 proved to be a professionally orchestrated event, where several months of marketing and storytelling had built significant interest in Apple's attempt to enter the handset market. Even before anyone officially had seen the device, the iPhone was considered a groundbreaking handset in its attempt to bring a new form of user experience to the mobile internet. Yet, already from the outset, Apple realised that the device alone would not be sufficient to excel outside the most devoted enthusiasts. Apple needed a strategy for enabling third-party applications and contents on the iPhone.

Early on, Apple decided to use the Safari web browser as a boundary resource for setting up a 'lean' interface between the consumer electronics firm and its anticipated third-party developers. Our data show that one of Apple's main motivations for using the Safari web browser was the huge installed base of existing applications and competent developers. This installed base represented a major external contribution opportunity, as little new learning would be required before developers would be able to contribute to the iPhone community. In addition, the Safari web browser was fully controlled by Apple. In this regard, the decision resonated well with Apple's history of preferring proprietary and tightly integrated solutions.

In parallel with Apple's sanctioned Safari strategy (triggering applications from firms such as Google, Flicker, New York Times, YouTube, Twitter and Facebook), however, third-party developers started to self-resource the platform by designing their own boundary resources such as new frameworks and sample codes. In the absence of third-party developers' native applications, some developers 'jailbreaked' the iPhone. This practice made it possible to install third-party native applications via unofficial installers. As an example, the 'iPhone Dev Team' became a well-known group of software engineers, launching their first jailbreak method 'JailbreakMe 1.0' for the iPhone firmware in 10 October 2007. In less than a year, more than 1.6 million jailbreaked devices were reported. These devices used the most known application installer 'Cydia', which also later offered an application store for jailbreaked devices.

The Safari web browser strategy stimulated the development of around 1000 iPhone applications. However, the strategy illustrated some of the difficulties of balancing external contribution and platform control. It is somewhat ironic that the control concerns, motivating the Safari web browser in the first place, generated new control concerns. These concerns included so-called 'jail-breaking' and unsanctioned application installers and stores, all intended to bypass Apple's sanctioned boundary resource. We refer to this unsanctioned resourcing as self-resourcing, i.e. third-party developers' act of developing new boundary resources as a response to perceived limitations in existing boundary resources.

Episode I: addressing self-resourcing through an SDK (March 2008–March 2009)

In view of the self-resourcing related to the Safari web browser strategy, Apple decided to rebalance the relationship between resourcing and securing in the hope that it would take away the incentives to infringe the platform. Apple responded to third-party developers' unanticipated self-resourcing by rethinking their boundary resources and introducing the 'iPhone Software Roadmap' as a new strategy for stimulating third-party contributions (as a powerful complement to the Safari web browser). Steve Jobs announced the release of the iPhone SDK on Apple's website:

> 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 [2008]. We are excited about creating a vibrant third party developer community around the iPhone and enabling hundreds of new applications for our users.

Platform resourcing

The iPhone Software Roadmap included a set of entirely new boundary resources including SDK, APIs and a distribution channel. First, the new SDK provided a development environment, a graphical user interface builder, a comprehensive analysis tool for the assessment of application performance and an iPhone simulator tool that facilitated application testing. The provided components and features were intended to serve as a robust environment for the integration of APIs. Second, Apple also provided a set of APIs for core services such as collections, address book, networking, file access, core location, net services, threading and URL utilities. Finally, in July 2008, Apple introduced an application distribution channel called the 'App Store', which provided users with functionality for search, browse, purchase and download. Steve Jobs explained the App Store idea:

> 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.

Platform securing

In view of the new boundary resources, Apple developed an application review process to secure the platform. This review process entailed an assessment of every App Store application by an Apple review team. The assessment involved securing an application's compatibility with Apple's regulations, guidelines and rules. For instance, restricted application and content types included porn, bandwidth hogs and malicious ones. The outcome of the assessment determined if an application would be made available for user download, or if it would be rejected and returned to third-party developers for redesign. Demonstrating that Apple was serious about its new form of governance, the review team rejected several applications and even withdrew originally accepted applications. Not surprisingly, the application review

process was criticised by a notable number of third-party developers. It was seen as irrational and too single-handedly focused on Apple's business interests.

In sum, Apple's balance act of both resourcing and securing the platform was successful in terms of ecosystem growth. In early 2009, there were 25 000 applications, 50 000 registered developers and 800 million user downloads. Even though self-resourcing did not stop during the episode, the release of the SDK and related boundary resources made it a marginal aspect of the ecosystem compared with Apple-controlled resourcing. In terms of securing, the introduction of the SDK initiated an application review process, manifesting regulation-based rather than technology-based securing. We refer to regulation-based securing as a platform owners' act of exercising control over the platform and its related services through administrative legislation.

Episode II: resourcing for diversity (March 2009–January 2010)

The introduction of the SDK as a response to self-resourcing proved successful for Apple and its community of developers. However, even with the space of possible resourcing enabled by the SDK and related APIs, the pressure to satisfy a growing and increasingly heterogeneous third-party community was increasing. Third-party developers still voiced criticism over the perceived unwillingness of Apple to open up the platform.

Given this criticism, the launch of the new iPhone device (iPhone 3GS) and the radically updated SDK including a multitude of APIs were important components in Apple's efforts to resource the new device capabilities and stimulating the increasing diversity of its third-party developers.

Platform resourcing

The new SDK provided more than 100 new features and 1000 new APIs, where some of them clearly addressed third-party developers' concerns about the narrow scope of the original boundary resources. By providing support for push notifications, map APIs, serial I/O and payment APIs, Apple diversified the iPhone scope into new application areas such as navigation and e-business. For instance, announced in June 2009, the awaited In-App Purchase API provided functionality for selling, e.g. subscriptions and content within third-party applications. After its update in October 2009, third-party developers were able to sell expansion packs and additional contents after first hooking up users with free applications. This was not least important for media companies such as newspapers.

Platform securing

Apple regularly revisited their application review policies. During this episode, the company introduced new restrictions on submitted applications. As an example, Apple regulated the use of the In-App Purchase API heavily, determining the type of content, functionality, services and

subscriptions that would be available through such applications. Another example concerned location-aware ads. As Apple's development centre 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.

In sum, during this episode, the 25 000 developed applications in March 2009 had more than doubled by July the same year (65 000). In a similar way, the number of registered developers doubled from 50 000 to 100 000 during the same four months. These figures suggest that Apple's diversification strategy, intended to expand the scope of the ecosystem was successful. We refer to this type of resourcing as diversity resourcing, i.e. deliberate action taken by a platform owner to diversify the platform in a way that stimulates new application areas.

Episode III: securing platform sovereignty (January 2010–April 2010)

As indicated above, diversity resourcing was a successful strategy. This strategy was reinforced in January 2010, as Apple announced the iPad, which was a tablet computer compatible with the iPhone platform. Offering a new type of device, still using the same platform, promised to expand the platform into new application areas.

Platform resourcing

The tight coupling between the iPad and the iPhone platform enabled the company to build on iPhone's installed base consisting of 140 000 developed applications, a huge number of users and an SDK including many APIs and development practices. In April 2010, Apple released SDK 4.0 for iPhone and iPad devices. This release enabled third-party developers to use multiple enhanced APIs and features. The most significant features of this release were: multitasking, folders, enhanced mail, iBooks, enterprise features, game centre and iAd (a mobile advertising platform developed by Apple Inc).

Platform securing

Once these new features for further increasing diversity were in place, Apple's attention shifted to the perceived problem of meta/cross platforms injected between the iPhone platform and third-party developers. The company did not want to facilitate simultaneous development for competitors' platforms. Such development would reduce Apple's control over the iPhone ecosystem. Consequently, as Apple introduced their SDK and APIs, the company also hindered meta-platforms to interoperate with them.

As a response, Adobe announced a 'Packager for iPhone' application intended for its next Flash developer tool version, the Creative Suite 5 (CS5). The CS5 simply turned Flash

applications into iPhone applications automatically. This move made Apple release a new developer license agreement with major implications for third-party developers. Section 3.3.1 in the iPhone SDK agreement read:

> 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).

Arguably, the revised developer license agreement directly targeted Adobe's new features. It also effectively addressed other actors' potential attempts to use meta-platforms for bypassing Apple's SDK and APIs.

In sum, the episode, which started as Apple launched the iPad, involved both diversity resourcing and regulation-based securing. Even though the resourcing strategy was successful (from 140 000 applications to 185 000 between January 2010 and March 2010), the securing actions dominated as the company made every attempt to counteract attempts to introduce meta/cross platforms. We label this type of securing as sovereignty securing, i.e. actions taken by a platform owner to achieve or maintain single control over the platform's evolution.

DISCUSSION

This paper has explored a set of issues that challenge the way we think about third-party development as an alternative form of systems development. In particular, it has highlighted the role of platform boundary resources as an interface between the platform owner and third-party developers. Boundary resources have typically been underestimated as elements in the cultivation of third-party development around a platform. This is unfortunate because boundary resources play a crucial role in the platform owner's balancing act of stimulating external contributions and maintaining platform control. A more comprehensive understanding of the role of boundary resources in third-party development is therefore vital.

This paper proposes the boundary resources model as a framework with which to make sense of third-party development and the processes by which the actions of a platform owner and developers are mediated by boundary resources. The model depicts two drivers of boundary resources design. Resourcing is the process by which the scope and diversity of a platform is enhanced. Securing is the process by which the control of the platform is increased. The model suggests that attention to both these drivers is important to cultivate a platform ecosystem. One-sided attention to one of these two drivers is a dead end, because it would jeopardise the performance on the other aspect of cultivating the platform and its ecosystem. In this regard, third-party development is a balance act.

We argue that the boundary resources model is useful for understanding this balance act, because it provides a coherent model with well-defined constructs for examining the actions of stakeholders involved in third-party development. We applied our model to a detailed analysis of three episodes of third-party development in the case of the iPhone platform. This application yielded four insights related to resourcing and securing, manifesting how variants of each of these actions can be manifested in particular stages of the evolution of an ecosystem. We refer to these variants as self-resourcing, regulated securing, diversity resourcing and sovereignty securing. They exemplify actions relevant in attempts to make the cooperation, yet competition, in third-party development play out in the way of mutual benefit form the involved stakeholders.

Self-resourcing, defined as third-party developers' act of developing new boundary resources as a response to perceived limitations in existing boundary resources, offers a construct to analyse third-party developers' own initiatives to resource the platform in ways that benefit their application development. This is a construct that may be useful for investigating cases where the governance of a popular platform leans too much in a securing direction. Regulation-based securing, defined as a platform owners' act of exercising control over the platform and its related services through administrative legislation, captures control actions that rely on regulative measures rather than technical restrictions to resourcing. This is a construct useful for understanding application review procedures in third-party development.

Diversity resourcing denotes deliberate action taken by a platform owner to diversify the platform in a way that stimulates new application areas. This type of resourcing stretches the product boundary (cf. Yoo *et al*., 2010), enabling new meaning-making (Yoo, 2010). This construct is useful for exploring how third-party development can enable a platform owner to transform its enterprise beyond its traditional industrial settings. Lastly, sovereignty securing refers to actions taken by a platform owner to maintain control of the platform's evolution and avoid becoming a substitute platform for application developers. This construct is useful for investigating the responses of platform owners when other platform owner's attempts to scale their own platform and ecosystems.

The insights derived from the boundary resources model enhance previous research on platforms and third-party development. First, it enhances the platform literature by providing a focused theoretical account of the interfaces between the platform and components developed by external parties. Traditional platform literature tends to examine the generation of derivative applications as an in-house process where reuse of existing components is the main emphasis (Meyer & Lehnerd, 1997; Robertson & Ulrich, 1998). While the design of interfaces usually is considered to be at the heart of scaling a platform (Katz & Shapiro, 1994; Baldwin & Clark, 2000), it is typically seen as a standardisation process in which interfaces serve as 'specifications and design rules that describe how the platform and modules interact and exchange information' (Tiwana *et al*., 2010, p. 676). In third-party development, we argue for conceptualising the interfaces between the platform and its developers as boundary resources. The notion of boundary resources, as it is framed in the boundary resources model, provides a means to more closely study the actions taken by stakeholders in third-party development. It

enables analysis that is geared towards appreciating boundary resources design as strategising rather than understanding interfaces as standards.

Second, as demonstrated in our case analysis, the boundary resources model can be specialised for examining specific aspects of third-party development that emerges in the wake of careful data analysis. Viewing, for instance, resourcing as a generic construct that can be situated to specific problems enables the application of an analytical lens for emerging research challenges in platform-centric systems development. This is an important aspect of our work, because the platform literature originates from economics and technology management (Baldwin & Clark, 2000), where the research emphasis is on aggregated phenomena rather than micro-level actions taken by stakeholders.

Third, platforms research typically shows little interest for opposing logics (cf. Robey & Boudreau, 1999), where seemingly contradicting forces coexist in innovation processes. Our model recognises such forces. It provides a basis with which tensions between resourcing and securing can be studied, perhaps by incorporating some form of dialectical model of organisational change (Van de Ven & Poole, 1995).

Finally, we contribute to the continued investigation of digital innovation (Henfridsson *et al*., 2009; Svahn *et al*., 2009; Kallinikos *et al*., 2010; Tilson *et al*., 2010; Yoo *et al*., 2010; Eaton *et al*., 2011) by zooming in on boundary resources as digital technology with the capacity to trigger innovation driven by multiple and uncoordinated third-party developers. Our research addresses Yoo *et al*.'s (2010) call for more research on the strategic roles of boundary resources as, e.g. API design choices can have a tremendous impact on the evolution of a software ecosystem and therefore also the fate of the platform owner.

Future studies could address limitations in our work. First, this paper draws on an extreme case study, where Apple's iPhone platform was chosen as an ideal case for theory generation. However, in increasing generality rather than plausibility, it would be useful to conduct studies using a representative case selection technique. Indeed, the boundary resources model should be tested for generality across empirical settings. Second, future studies could adopt a variance perspective that would develop a causal model for testing different boundary resource strategies and their influence on growth of applications, applications developers and knowledge heterogeneity. Finally, another direction for future work would be to investigate the role of boundary resources from the perspective of third-party developers.

CONCLUSION

In this paper, we contribute a model specifying relationships between constructs useful for analysing the role of boundary resources in third-party development. The model pinpoints two key drivers behind the design and use of boundary resources in third-party development: resourcing and securing. Resourcing happens when: (a) platform owners design boundary resources for extending the scope and diversity of the platform; and (b) third-party developers use these resources in their application development. Securing occurs when a platform owner designs boundary resources for controlling the platform. As there is an inherent tension

between resourcing and securing, the boundary resources model can be seen as a useful lens with which to understand this balancing act.

Our application of the model also contributes four specialised constructs for understanding the actions taken by stakeholders in third-party development: self-resourcing, regulation-based securing, diversity resourcing and sovereignty securing. Each of these constructs can be used as a starting-point for adapting the boundary resources model to specific research problems in third-party development.

On a final note, it can be observed that platforms have received recent attention in the field of IS. We believe this growing interest is useful, not least for understanding new modes of systems development. In this vein, our work suggests one path for future research on systems development.

ACKNOWLEDGEMENTS

The Swedish Research School of Management and Information Technology (MIT) and the Sustainable Transport Initiative partially funded this work. We are also deeply indebted to the special issue editors, Karl-Heinz Kautz and Peter Axel Nielsen, and two anonymous reviewers for their useful comments on our research.

REFERENCES

Baldwin, C.Y. & Clark, K.B. (2000) *Design Rules – The Power of Modularity*. MIT Press, Cambridge, MA, USA.

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

Bosch, J. (2009) From software product lines to software ecosystems. *The 13th International Software Product Line Conference (SPLC 2009)*, San Francisco, CA, USA.

Boudreau, K.J. (forthcoming) Let a thousand flowers bloom? An early look at large numbers of software 'apps' developers and patterns of innovation. *Organization Science*, doi:10.1287/orsc.1110.0678.

Boudreau, K.J. & Lakhani, K.R. (2009) How to manage outside innovation. *Sloan Management Review*, **50**, 69–76.

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

Denzin, N.K. (1970) *The Research Act: A Theoretical Introduction to Sociological Methods*. Aldine, Chicago, IL, USA.

Eaton, B., Elaluf-Calderwood, S., Sørensen, C. & Yoo, Y. (2011) Dynamic structures of control and generativity in digital ecosystem service innovation: the cases of the Apple and Google mobile app stores. LSE Working Paper.

El Sawy, O.A., Malhotra, A., Park, Y. & Pavlou, P.A. (2010) Seeking the configurations of digital ecodynamics: it takes three to tango. *Information Systems Research*, **21**, 835–848.

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

Gawer, A. & Cusumano, M. (2002) *Platform Leadership: How Intel, Microsoft and Cisco Drive Industry Innovation*. Harvard Business School Press, Boston, MA, USA.

George, A.L. & Bennett, A. (2005) *Case studies and theory development in the social sciences*. MIT Press, Cambridge, MA, USA.

Gerring, J. (2007) *Case Study Research: Principles and Practices*. Cambridge University Press, Cambridge, UK.

Ghazawneh, A. & Henfridsson, O. (2010) Governing third-party development through platform boundary

resources. *International Conference on Information Systems (ICIS)*, St. Louis, MO, USA.

Hargadon, A.B. & Douglas, Y. (2001) When innovations meet institutions: Edison and the design of the electric light. *Administrative Science Quarterly*, **46**, 476–501.

Henfridsson, O. & Lindgren, R. (2010) User involvement in developing mobile and temporarily interconnected systems. *Information Systems Journal*, **20**, 119–135.

Henfridsson, O., Yoo, Y. & Svahn, F. (2009) Path creation in digital innovation: a multi-layered dialectics perspective. *Sprouts: Working Papers on Information Systems*, **9**, 1–26. [WWW document]. URL http://sprouts.aisnet.org/9-20/.

von Hippel, E. & Katz, R. (2002) Shifting innovation to users via toolkits. *Management Science*, **48**, 821–834.

Kallinikos, J., Aaltonen, A. & Marton, A. (2010) A theory of digital objects. *First Monday*, **15**, [WWW document]. URL http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/3033/2564.

Katz, M. & Shapiro, C. (1994) Systems competition and network effects. *Journal of Economic Perspectives*, **8**, 93–115.

Kenney, M. & Pon, B. (2011) Structuring the smartphone industry: is the mobile internet OS platform the key? *Journal of Industry Competition and Trade*, **11**, 239–261.

Kieser, A. (1994) Why organization theory needs historical analyses – and how this should be performed. *Organization Science*, **5**, 608–620.

Klein, H.K. & Myers, M.D. (1999) A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, **23**, 67–93.

Langley, A. (1999) Strategies for theorizing from process data. *Academy of Management Review*, **24**, 691–710.

Lopreato, J. & Alston, L. (1970) Ideal types and the idealization strategy. *American Sociological Review*, **35**, 88–96.

Markus, M.L. & Robey, D. (1988) Information technology and organizational change: causal structure in theory and research. *Management Science*, **34**, 583–598.

Mathiassen, L. & Sørensen, C. (2008) Towards a theory of organizational information services. *Journal of Information Technology*, **23**, 313–329.

Messerschmitt, D.G. & Szyperski, C. (2003) *Software Ecosystem: Understanding an Indispensable Technology and Industry*. MIT Press, Cambridge, MA, USA.

Meyer, M.H. & Lehnerd, A.P. (1997) *The Power of Platforms: Building Value and Cost Leadership*. The Free Press, New York, NY, USA.

Orlikowski, W.J. (1992) The duality of technology: rethinking the concept of technology in organizations. *Organization Science*, **3**, 398–427.

Pohl, K., Böckle, G. & van der Linden, F. (2005) *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, Berlin, Germany.

Prügl, R. & Schreier, M. (forthcoming) Learning from leading-edge customers at the Sims: opening up the innovation process using toolkits. *R&D Management*, **36**, 237–250.

Robertson, D. & Ulrich, K. (1998) Planning for product platforms. *Sloan Management Review*, **39**, 19–31.

Robey, D. & Boudreau, M.C. (1999) Accounting for the contradictory organizational consequences of information technology: theoretical directions and methodological implications. *Information Systems Research*, **10**, 167–185.

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

Selander, L., Henfridsson, O. & Svahn, F. (2010) Transforming ecosystem relationships in digital innovation. *International Conference on Information Systems (ICIS)*, St. Louis, MO, USA.

Silverman, D. (2006) *Interpreting Qualitative Data: Methods for Analyzing Talk, Text, and Interaction*, 3rd edn. SAGE, Los Angeles, CA, USA.

Svahn, F., Henfridsson, O. & Yoo, Y. (2009) A threesome dance of agency: mangling the sociomateriality of technological regimes in digital innovation. *International Conference on Information Systems (ICIS)*, Phoenix, AZ, USA.

Teece, D.J. (1986) Profiting from technological innovation: implications for integration, collaboration, licensing and public policy. *Research Policy*, **15**, 285–305.

Tilson, D., Lyytinen, K. & Sørensen, C. (2010) Digital infrastructures: the missing is research agenda. *Information Systems Research*, **21**, 748–759.

Tiwana, A., Konsynski, B. & Bush, A. (2010) Platform evolution: coevolution of platform architecture, governance, and environmental dynamics. *Information Systems Research*, **21**, 685–687.

Van de Ven, A.H. (2005) Running in packs to develop knowledge-intensive technologies. *MIS Quarterly*, **29**, 365–378.

Van de Ven, A.H. & Poole, M.S. (1995) Explaining development and change in organizations. *Academy of Management Review*, **20**, 510–540.

Weber, M. (1949) *The Methodology of the Social Sciences*. Free Press, Glencoe, IL, USA.

West, J. & Mace, M. (2010) Browsing as the Killerapp: explaining the rapid success of Apple's iPhone. *Telecommunications Policy*, **34**, 270–286.

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

Yoo, Y. (2010) Computing in everyday life: a call for research on experiential computing. *MIS Quarterly*, **34**, 213–231.

Yoo, Y., Henfridsson, O. & Lyytinen, K. (2010) The new organizing logic of digital innovation: an agenda for information systems research. *Information Systems Research*, **21**, 724–735.

Zittrain, J.L. (2006) The generative internet. *Harvard Law Review*, **119**, 1974–2040.

Biographies

Ahmad Ghazawneh is a PhD Candidate at the Informatics Department, Jönköping International Business School, Jönköping University, Sweden. He is also a member of the Swedish Research School MIT and the Centre for Information Technology and Information Systems. His research interests include digital and open innovation, software platforms and ecosystems, digital and social networks as well as IS development. His research has been published in international journals in the IS discipline, such as *International Journal of Networking and Virtual Organisations*, and conferences, such as International Conference on Information Systems, European Conference on Information Systems and Hawaii International Conference on System Sciences.

Ola Henfridsson is a Professor of Applied Information Technology at Chalmers University of Technology, Sweden, and an Adjunct Professor at Department of Informatics, University of Oslo, Norway. His research interests include digital innovation, technology management, organisational adaptation of information technology as well as process- and design-oriented research. The outcome of this research has been published in *MIS Quarterly*, *Information Systems Research*, *Information and Organization*, *Information Systems Journal*, *Information Technology and People*, *Journal of Strategic Information Systems* and other journals in the IS discipline. He is a Senior Editor Emeritus of the *MIS Quarterly* and serves on the editorial boards of *Information Technology and People* and *Journal of the Association for Information Systems*

APPENDIX

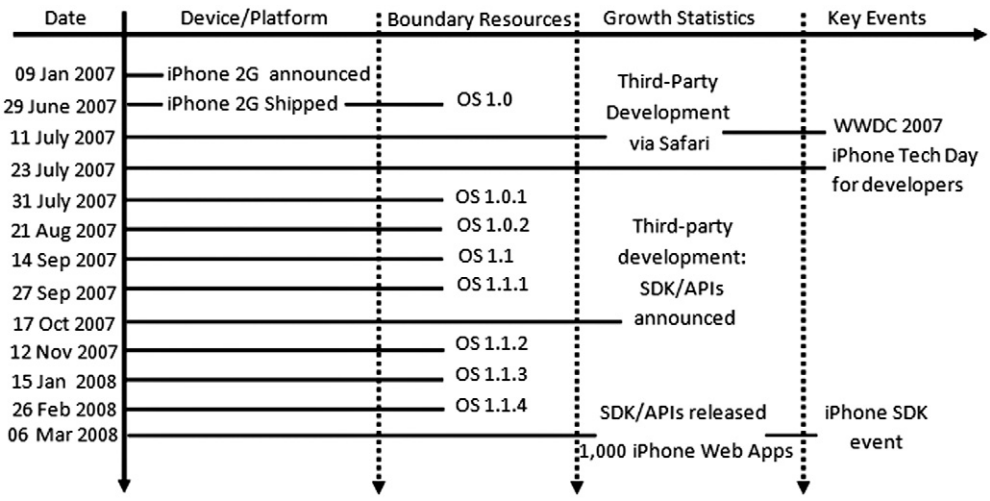


Figure A1. Timeline of pre-SDK episode.

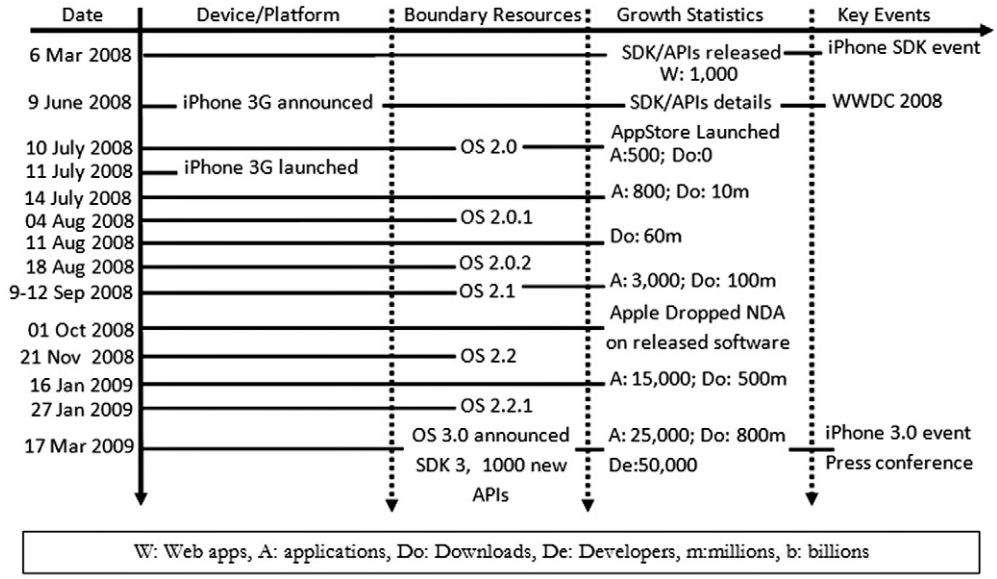


Figure A2. Timeline of episode I.

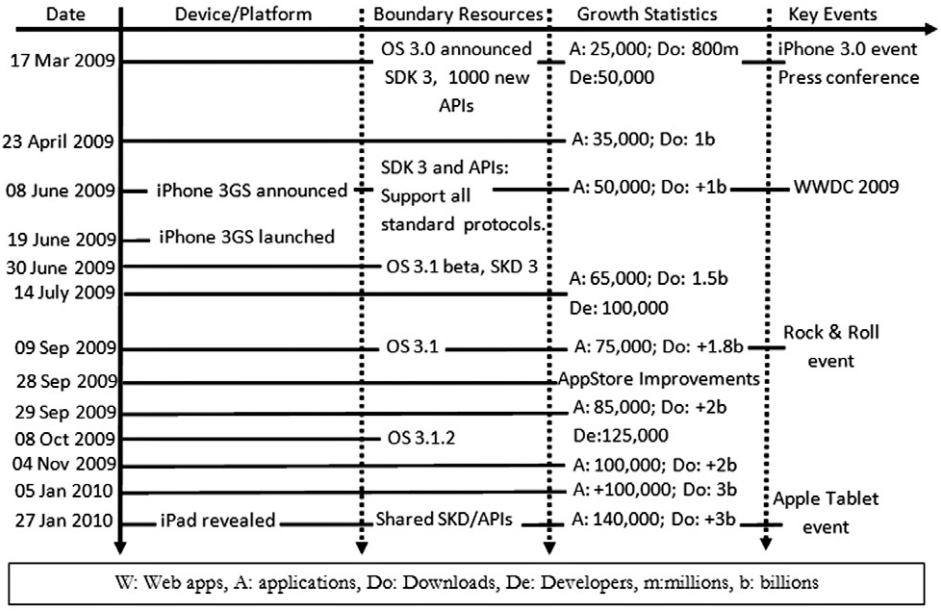


Figure A3. Timeline of episode II.

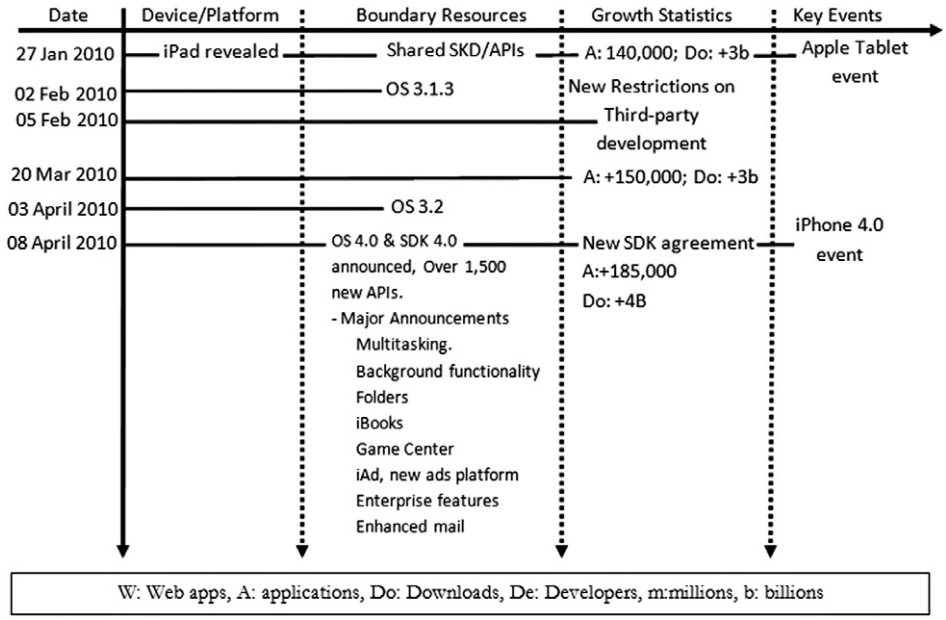


Figure A4. Timeline of episode III.