The tuning of data platform boundary resources: insights from Twitter

Joseph Rubleske
Northern Kentucky University

ABSTRACT

Digital platforms facilitate transactions between buyers and sellers, content consumers and producers, and other market participants. Providers of digital platforms also leverage third-party application developers (3PADs) in order to add functionality to their platform (McIntyre and Srinivasan, 2017). To attract 3PADs to their platform, many providers adopt governance practices that are favorable to 3PADs (Tiwana, 2014). One such practice is the provision of platform boundary resources (PBRs) – such as application programming interfaces (APIs) and developer agreements – that are used mostly to support third-party development, but also may be used to restrict it.

Existing research into PBRs is limited. One notable gap is the absence of a conceptualization of the provision and regulation of platform data. To address this gap, this study introduces the concept of "data PBRs" which aid 3PADs in the collection of platform data. This study also extends the concept of PBR "tuning" (Eaton et al., 2015) by exploring how data PBRs are contested by providers and 3PADs in order to further their respective interests. To this end, Twitter's digital platform is examined as a revelatory case. Using grounded-theory techniques (Glaser, 1998), 91 pertinent technology articles are analyzed, revealing four principal data tuning actions (data provision, data outsourcing, data limiting, data monetization) and seven variants (e.g., data provision to add functionality, data limiting to gatekeep apps). A discussion of this proposed typology considers why data tuning attempts by 3PADs were unsuccessful and how Twitter's apparent success may have limited its longer-term growth. Additional conceptual implications also are considered.

Keywords: digital platforms, platform boundary resources, application programming interfaces, platform data, data tuning

INTRODUCTION

Today, many of the companies with the highest market capitalization (e.g., Alphabet Inc., Amazon.com, Apple) are providers of digital platforms, and markets such as personal transport, wayfinding, shipping, and lodging are being radically disrupted by companies that bring together market participants such as buyers and sellers or content consumers and producers (Choudary, 2015; Kohler, 2018). These digital platform providers are characterized in part by their efforts to enable, promote, and leverage the development of third-party software applications (apps) that draw data from and extend their digital platform (Ceccagnoli, Forman, Huang, and Wu, 2014; McIntyre and Srinivasan, 2017). Indeed, recent research has found that external value creation through third-party apps may be just as important for platform growth as value created internally by the digital platform provider (Choudary, 2015; Parker, van Alstyne, and Choudary, 2016; Song, Xue, Rai, and Zhang, 2018).

Accordingly, the competition between digital platform providers to attract third-party application developers (3PADs) is often intense (Benlian, Hilkert, and Hess, 2015). A review of the germane literature suggests that one of the most important means of attracting 3PADs may be the size of the user base, as this factor has been found to correlate positively with revenue generation potential (Song et al., 2018). Many other studies have found, though, that the adoption by digital platform providers of certain platform governance practices may be just as important, if not more so (e.g., Tiwana, 2014; Wareham, Fox, and Cano Giner, 2014). Such practices include sharing revenue with 3PADs (e.g., Oh, Koh, and Raghunathan, 2015), making decisions collaboratively with 3PADs (e.g., Gawer, 2014, Kazan, Tan, and Lim, 2016), and the granting of intellectual property to 3PADs (e.g., Cusumano and Gawer, 2002).

Another platform governance practice that has received less attention – despite the increasing importance of platform data – is the provision by the digital platform provider of platform boundary resources to 3PADs. A platform boundary resource (PBR) is a "software tool [or] regulation" – such as an application programming interface (API), developer agreement, or software development kit (SDK) – that serves as a resource which enables or supports the development of third-party apps for a digital platform (Ghazawneh and Henfridsson, 2013). PBRs are crucial mechanisms with which digital platform providers balance the need for platform generativity and evolvability with the need for platform stability and identity (Eaton, Elaluf-Calderwood, Sørensen, and Yoo, 2015; Wareham et al., 2014). The nature of the relationship between digital platform providers and 3PADs, as shaped by PBRs, is summarized in Figure 1 (see the appendix).

Existing studies that focus on and conceptualize PBRs are limited. Indeed, the review conducted for this article found only seven articles that conceptualize PBRs. In the first and most seminal of these seven articles, Ghazawneh and Henfridsson (2013) drew from a case study of Apple's iOS platform to propose a "boundary resources model" in which a digital platform provider engages in acts of "resourcing" and "securing." Resourcing aims at generating additional functionality for the platform and entails providing PBRs that enable 3PADs to develop third-party apps. Securing aims at establishing greater platform control and entails limiting the utility of PBRs or strategically withholding them.

Every subsequent conceptualization of PBRs draws from the concepts of resourcing and securing. For example, the concept of "boundary resource dependency" (Rafiq, Ågerfalk, and Sjöström, 2013) outlines the ways in which 3PADs may be adversely affected by acts of securing, while Karhu and Gustafsson's (2015) typology of PBR exploits (i.e., cloning, forking,

hacking, substituting) points to the need for securing practices. Arguably the most fruitful extension of resourcing and securing, though, is Eaton et al.'s (2015) concept of PBR tuning. According to Eaton et al. (2015), PBR tuning occurs when a digital platform provider or a 3PAD contest and/or attempt to re-shape a PBR in order to further their respective interests. This PBR-centered conflict drives the evolution of the digital platform; thus, PBRs can be understood as the locus of control over a digital platform.

Given how few studies have conceptualized PBRs, numerous research gaps remain. Perhaps the most conspicuous of these gaps is the absence of a conceptualization of the offering, collection, and regulation of platform data. As noted by Schreieck, Wiesche, and Krcmar (2016, p. 12), no extant study "explicitly analyses the role of data as a boundary resource" despite the fact that "many of today's platform ecosystems are fueled by data." To date, the role played by data in shaping the evolution of a digital platform has only been (at best) implied by studies that examine APIs. To address this gap, this study first introduces the concept of a "data PBR," which is defined as a PBR that enables or aids 3PADs in the collection of platform data. Second, this study explores how two types of data PBRs (i.e., APIs and data regulations) are contested and re-shaped (i.e., tuned) by digital platform providers and 3PADs. In doing so, insights are generated into how data PBRs help shape a digital platform's evolution.

This article is organized into six main sections. Following this Introduction, the digital platform and PBR literatures are reviewed, with a focus on conceptualizations of PBRs. The third section comprises the study's methodology, which includes a rationale for the case study approach, a rationale for the selection of Twitter as the digital platform provider (case), and details about how data were collected and analyzed. The study's fourth section presents a rich, detailed narrative of the role of data PBRs in the evolution of Twitter's digital platform, while the fifth section draws from this narrative to provide an analysis of data tuning actions by Twitter and 3PADs. The sixth and final section presents key findings, conceptual implications, and directions for future research.

DIGITAL PLATFORMS AND BOUNDARY RESOURCES

For nearly two decades, the economic growth and rising market dominance of digital platforms and ecosystems has been well documented (e.g., Cusumano and Gawer, 2002; Fu, Wang, and Zhao, 2017; Gawer, 2009; McIntyre and Srinivasan, 2017; Tiwana, Konsynski, and Bush, 2010). Today, many of the companies with the highest market capitalization (e.g., Alphabet Inc., Amazon.com, Apple, Facebook, Microsoft) are providers of digital platforms. Moreover, markets ranging from personal transport (ClickBus, Lyft, Uber) and wayfinding (Moovit, Waze) to shipping (Shyp, TruckerPath) and lodging (AirBnB) are being radically disrupted by companies that bring together users and/or market participants (Choudary, 2015; Kohler, 2018).

This study refers to such companies as digital platform providers. In short, these companies provide digital platforms that facilitate interactions between users (e.g., Facebook, Instagram, Twitter) or market transactions between buyers and sellers (e.g., Etsy, Uber), content consumers and producers (e.g., Quora, Yelp, YouTube), or users and third-party application developers (e.g., Google Android, Apple iOS, MS Windows). Digital platform providers also are characterized by their efforts to enable, promote, and leverage the development of third-party software applications (apps) that draw data from and extend their digital platform (Ceccagnoli et al., 2014; McIntyre and Srinivasan, 2017).

Digital platform providers enable and even promote third-party app development for two reasons. First, by making third-party apps available to platform users, they expand the scope of offerings to users, which in turn may attract more users to the platform (Benlian et al., 2015). This tendency of users to adopt the platforms with the most complementary offerings is referred to as indirect network effects or positive cross-side effects (e.g., Katz and Shapiro, 1986; Parker et al., 2016). Second, most platform providers are unable to innovate internally at a rate that is needed to thrive or even survive in a highly competitive technology market (Mantovani and Ruiz-Aliseda, 2017). Accordingly, most platform providers rely increasingly on new, innovative apps produced by third-party developers. By leveraging these third-party apps, platform providers attempt to decrease costs while forging a market position in which their platform's value proposition can be distinguished from that of rivals (Cennamo and Santalo, 2013).

For their part, third-party application developers (3PADs) participate in digital platform ecosystems chiefly because most digital platforms offer low-cost access to an established base of users (Ceccagnoli et al., 2014; Kim, Kim, and Lee, 2016). From a 3PAD's perspective, a platform with a large number of users typically offers greater revenue potential (Venkatraman and Lee, 2004). In addition, other studies have found that many 3PADs are "power users" who are driven to produce desired apps not offered by the platform provider (Koch and Guceri-Ucar, 2017; Rivard and Huff, 1985).

For digital platform providers, competitive advantage still depends to some degree on internal value creation in the form of a modular and stable platform architecture (Gawer, 2014) and a compelling core service (Saarikko, 2016). However, recent research has found that external value creation through third-party apps may be just as important, particularly for platform growth (e.g., Choudary, 2015; Parker et al., 2016; Song et al., 2018). Not surprisingly, then, the competition between digital platform providers to attract 3PADs is often intense. Which factors, then, lead 3PADs to contribute to one digital platform and not another? A review of the literature suggests that one of the most important factors may be a large number of platform users, as this factor has been found to correlate positively with revenue generation potential (Song et al., 2018). In their seminal study of the U.S. video game industry from 1976 to 2002, for example, Venkatraman and Lee (2004) found that the number of customers was the main determinant of platform adoption by video game developers.

The size of the user base is not the only factor that attracts 3PADs to a digital platform, though. Indeed, numerous studies have argued that the adoption by digital platform providers of platform governance practices that are favorable to 3PADs may be even more important (e.g., Tiwana, 2014; Wareham et al., 2014). The literature on platform governance has identified several such policies and practices, including the adoption of widely accepted technology standards (e.g., Thomas, Autio, and Gann, 2014), the sharing of revenue with 3PADs (e.g., Oh et al., 2015), transparent and accountable decision making (e.g., Benlian et al., 2015), collaborative governance (e.g., Gawer, 2014; Kazan et al., 2016), the strategic management of platform updates (Song et al., 2018), and the granting of intellectual property (IP) to 3PADs (Ceccagnoli, Forman, Huang, and Wu, 2012; Cusumano and Gawer, 2002).

Another platform governance practice that has received less attention – despite its increasing importance in data-driven markets – is the provision by the digital platform provider of platform boundary resources to 3PADs (Ghazawneh and Henfridsson, 2013). In short, a platform boundary resource (PBR) is a digital artifact (e.g., an application programming interface, a developer agreement, a code library, documentation) that allows 3PADs to access digital platform resources and/or aids 3PADs in making use of these resources. In the following

section, this author describes PBRs and their strategic use, identifies and describes the four main types of PBRs, reviews the extant literature on them, and identifies the research gap that this article begins to fill.

Platform Boundary Resources

A platform boundary resource (PBR) has been defined as the "software tools and regulations that serve as the interface for the arm's length relationship between the platform owner and the application developer" (Ghazawneh and Henfridsson, 2013, p. 174). The term "boundary resource" originates from sociological research on boundary objects used by actors to bridge two or more knowledge domains (Star and Griesemer, 1989). Such objects are said to be adaptable across contexts while sufficiently robust to maintain a stable identity (Bergman, Lyytinen, and Mark, 2007). Thus, a PBR can be understood as an object (in this case a digital artifact) designed for use by two distinct groups, namely, digital platform providers and 3PADs (Ghazawneh and Henfridsson, 2013).

Extant research on PBRs has framed them as key mechanisms with which digital platform providers balance the need for platform evolvability and generativity with the need for platform stability and identity (Eaton et al., 2015; Wareham et al., 2014). To satisfy the former need, digital platform providers may use PBRs to incentivize participation by 3PADs, support the development efforts of 3PADs, and/or permit 3PADs to market their apps to platform users (Ghazawneh and Henfridsson, 2013). To satisfy the latter need, digital platform providers may withhold or limit the use of PBRs, avoid overcommitting internal resources, and/or prevent third-party apps from replicating a core service offered by the digital platform provider (Mohagheghzadeh and Svahn, 2016). By preventing "replicate apps," the digital platform provide attempts to prevent a 3PAD from becoming a rival.

But which types of digital artifacts qualify as a PBR? Ghazawneh and Henfridsson (2013, p. 175) characterize a PBR as a "resource that provides access to core modules of the platform," with only application programming interfaces (APIs) and software development kits (SDKs) receiving explicit mention. Rafiq et al. (2013) refer to APIs, documentation, and "legal documents" as PBRs, while Mohagheghzadeh and Svahn (2016) identify APIs, SDKs, documentation, and "license agreements." Similarly, dal Bianco, Myllärniemi, Komssi, and Raatikainen (2014) include APIs, SDKs, and technical support as PBRs. In sum, four PBRs are common to most PBR-based studies, each described here in turn.

- An application programming interface (API) is a uniform resource locator (URL) or "endpoint" through which a 3PAD can access platform data (Puschmann and Ausserhofer, 2017). Included in the URL – which functions as a call or request to a server – are request conditions (e.g., authentication) and parameters (e.g., start and end dates, record identifiers). If executed successfully, a request returns data encoded (typically) as JSON or XML objects.
- A developer agreement may specify (1) how 3PADs may (and may not) use the platform's data, (2) how the digital platform provider may use any third-party app, and (3) how third-party apps may be marketed (Mohagheghzadeh and Svahn, 2016). Some developer agreements may also include a set of guidelines that stipulate how platform integrity must be maintained and how user privacy must be respected.
- A software development kit (SDK) may comprise code libraries, developer tools (e.g., debugging and testing tools, component management tools), and sample code (dal Bianco et al., 2014). In most cases, a code library consists of a set of language-specific subroutines

around a certain function (e.g., image processing). In essence, SDKs are app development aids.

- Technical support refers to digital artifacts that (1) describe the APIs and/or data available to 3PADs and (2) enable the sharing of knowledge about how to successfully execute API requests or make use of the data objects yielded by API requests (dal Bianco et al., 2014). Technical support content is typically static (e.g., documentation), but some digital platform providers may assign employees to answer questions posed by 3PADs in discussion forums.

Conceptualizations of Platform Boundary Resources

Existing studies that focus on and conceptualize PBRs are limited. For example, while there are several articles that focus on knowledge communities as digital platforms for users (e.g., Barrett, Oborn, and Orlikowski, 2017), none of these articles frames an online knowledge base as a PBR. Similarly, there are numerous articles that examine some platform governance mechanism that may be addressed in a developer agreement (e.g., intellectual property rights, revenue sharing), but few of these articles focus on developer agreements per se, and none of them frame the governance mechanism as a PBR.

Indeed, the review conducted for this article found only seven articles that conceptualize PBRs. In the first of these seven articles (chronologically), Ghazawneh and Henfridsson (2013) conducted a case study of Apple's iOS platform in order to explain how a digital platform provider uses PBRs to govern a platform ecosystem. Specifically, they draw from their case study to propose a "boundary resources model" in which a digital platform provider designs PBRs to (1) establish greater platform control and/or (2) enhance platform scope and diversity. The former activity is referred to as "securing," which "prevents the development of applications that risk infringing on the platform" (p. 177). The latter activity is referred to as "resourcing," and entails providing PBRs that enable 3PADs to develop third-party apps (as platform complements).

The acts of resourcing and securing by digital platform providers figure centrally in every subsequent conceptualization of PBRs. In adopting the perspective of a 3PAD, Rafiq et al. (2013, p. 208) argued that 3PADs tend to be "boundary resource dependent" such that "changes in the [PBR] directly affect third party application development and maintenance, which in extreme cases can affect the implementation and/or working of the third-party applications." Accordingly, Rafiq et al. (2013) argue for the relative importance of resourcing, and in turn propose a set of design principles that can help prevent or moderate "breakdowns" stemming from changes made to PBRs by digital platform providers.

Ghazawneh and Henfridsson's (2013) concept of resourcing has been extended by two other articles. Mohagheghzadeh and Svahn (2016, p. 11) drew from case study findings to conclude that resourcing may be done more effectively through collaborative, "continuous interactions between platform owners and external developers." In other words, digital platform providers should enlist 3PADs as co-designers of PBRs. Wulf and Blohm (2017) drew from a cluster analysis of randomly sampled APIs to propose three API "archetypes," each representing a distinct way in which an API is used for resourcing purposes. Integrator APIs are mostly used for enterprise app development, and enable 3PADs to integrate API functionality into existing information systems. Free Data Provider APIs are typically offered free of charge by public or non-profit organizations with the aim of stimulating open innovation, while Mediator APIs enable third parties to develop apps that draw from platform data and, per adherence to certain

conditions, may be marketed to platform users. This type of API receives the bulk of attention from PBR research, including this study.

To date, somewhat less attention has been paid to Ghazawneh and Henfridsson's (2013) concept of securing. Karhu and Gustafsson (2015) identified four ways in which 3PADs exploit PBRs: forking; hacking; cloning; and substituting. In doing so, they highlighted the need for certain securing practices by digital platform providers. Mohagheghzadeh and Svahn (2016) cautioned that "it is important to recognize that existing organizational resources often make up the primary material" for PBRs. Moreover, making these internal resources publicly available – through APIs, in particular – "may destroy any competitive advantage that had been achieved in part by protecting them" (p. 11).

The ongoing tension between resourcing and securing the digital platform is further conceptualized in Eaton et al.'s (2015) research into how PBRs evolve over time. Through an embedded case study of Apple's iOS platform, Eaton et al. (2015) applied to PBR design the conception of "tuning" (Barrett, Oborn, Orlikowski, and Yates, 2012), where innovators try to "exercise agency over materials through a dialectic of resistance and accommodation" (Eaton et al., 2015, p. 218). In one embedded case, Eaton et al. (2015) detail attempts by Apple and 3PADs to contest and/or re-shape (i.e., tune) Apple's iPhone. When it launched the iPhone in July 2007, Apple stipulated that third-party apps could only run within Apple's mobile Safari browser – a condition imposed in order to "protect telecom partners' networks and the security of the iPhone itself" (p. 224). Apple enforced this condition by designing its kernel module to prevent 3PADs from installing apps on it.

Apple's restrictions frustrated 3PADs and consumers alike, and numerous hackers attempted to reengineer the kernel module. Apple worked to resist these attempts, but in late August 2007 a means of hacking the kernel module was published, thereby enabling iPhone users to "jailbreak" their device. One month later, Apple released an update which prevented the published means of jailbreaking. Within a few weeks, though, hackers figured out how to jailbreak the updated iPhone. In response, Steve Jobs announced in late October 2007 that by April 2008 Apple would allow the "creation of apps in iOS native code" (p. 225). In February 2008, Jobs added another condition, namely, that apps permitted to run on the iOS platform could only be marketed through an App Store to be administered by Apple. Frustrated by this additional requirement, a group of 3PADs collaborated to launch (in March 2009) an alternative App Store called Cydia. Apple responded to Cydia in April 2009 by prohibiting the distribution of apps "from any other source than the official App Store" (p. 225). According to Eaton et al. (2015, p. 226), this "cat and mouse game between hackers and Apple with the [kernel] module persists to the present day."

Platform Data as a Boundary Resource: A Research Gap

In sum, extant research into PBRs has introduced the concepts of resourcing and securing (Ghazawneh and Henfridsson, 2013), identified best practices around resourcing (Mohagheghzadeh and Svahn, 2016; Rafiq et al., 2013) and securing (Karhu and Gustafsson, 2015), and argued for the relative importance of resourcing or securing. In addition, Eaton et al. (2015) have argued that because PBRs are the locus of control over a platform, conflict between digital platform providers and 3PADs occurs through them. In turn, their respective efforts to "tune" PBRs drive the digital platform's evolution.

Given how few studies have conceptualized PBRs, numerous research gaps remain. For example, while a few studies have shed empirical light on resourcing and securing practices, there is a need for studies that determine the extent to which such practices are employed and for studies that describe practices in greater detail. The absence of a conceptualization of the offering, collection, and regulation of platform data through PBRs constitutes another gap. As Schreieck et al. (2016, p. 12) noted, despite the fact that "many of today's platform ecosystems are fuelled by data," no extant study "explicitly analyses the role of data as a boundary resource in platform ecosystems."

While digital platform providers typically leverage user-generated platform data to develop core platform services (e.g., real-time traffic information, Facebook's Friend Finder) and/or deliver personalized advertisements, they also try to capitalize on platform data by making some portion of it available to 3PADs (Gawer, 2014). In turn, 3PADs collect and use the data to develop apps that attract more users to the platform, thus benefiting the digital platform provider. To date, though, the role played by platform data in shaping a digital platform's evolution has only been implied by studies that examine APIs. Simply put, no study to date has explicitly defined "data PBRs," identified their types, or explored how they affect digital platform evolution.

Data Platform Boundary Resources

Given this conceptual need, this study explores how certain "data PBRs" – that is, PBRs that enable or aid 3PADs in the collection of platform data – are contested and re-shaped (i.e., tuned) by digital platform providers and 3PADs. In doing so, this study generates insights into how data PBRs help shape a digital platform's evolution. But which PBRs can be understood as data PBRs? Drawing from the definitions (provided above) of each of the four main types of PBRs – APIs, SDKs, developer agreements, and technical support – several data PBRs can be proposed, including APIs, data regulations, data object descriptions, technical support for data collection, and example endpoints. The latter can be defined as parameterized URLs that offer examples of the kinds of queries that a 3PAD can use to collect data.

Of these proposed data PBRs, only APIs and data regulations are likely to be contested and/or re-shaped by digital platform providers and 3PADs in an attempt to further their respective interests. This assertion is supported by the observation that APIs and data regulations may be used for securing as well as resourcing purposes. With regard to the act of resourcing (which promotes platform generativity), data regulations permit the collection of a specified amount and variety of platform data, while APIs enable the collection of these data. With regard to the act of securing (which maintains platform integrity or prevents the replication of core services), data regulations may effectively limit the amount and/or variety of data that 3PADs are permitted to collect, while APIs may be withheld from 3PADs in order to prevent the collection of some platform data. In effect, the use of APIs and data regulations for securing purposes makes them instrumental as tuning objects.

The other proposed data PBRs may be used for resourcing purposes, but the notion of securing does not well apply to them. For example, if a digital platform provider withholds data object descriptions or example endpoints from 3PADs, then it may be more difficult for a 3PAD to figure out, in a technical sense, how to collect platform data. The 3PAD is not prevented from collecting platform data, though. Thus, it is unlikely, or at least less likely, that these data PBRs

are used for tuning purposes by digital platform providers or 3PADs. Accordingly, this study's data collection and analysis focuses on two data PBRs, namely, APIs and data regulations.

METHODOLOGY

The exploratory question of how certain data platform boundary resources (PBRs) may be tuned (i.e., contested and re-shaped) by digital platform providers and third-party application developers (3PADs) demands a method that helps the researcher better understand a phenomenon and, in turn, create new concepts or extend existing ones. The case study method satisfies these demands (Eisenhardt, 1989; Yin, 2009). For the purpose of this study, the desired case was a digital platform for which some portion of platform data are accessible. Twitter was chosen as the case because (1) it has always provided public access to some of its data and (2) it remains one of the more popular social media platforms.

Launched in 2006, Twitter provides microblogging and social networking services to roughly 330 million active global users (Statista, 2018). Twitter's registered users can post and read short messages called "tweets," and can also search tweets, "follow" other Twitter users, and make use of various services such as Twitter Moments, which allows them to develop and share a "curated story" based on tweets. In addition, and as detailed herein, Twitter's services have always been augmented to some extent by an ecosystem of third-party applications that draw from data made available by Twitter. Thus, Twitter's end users have benefited from services offered by both Twitter and 3PADs.

First Stage of Data Collection and Analysis

In order to explore how data PBRs may be tuned by digital platform providers and 3PADs, this author adopted an approach similar to that used in Eaton et al. (2015). In exploring how PBRs evolve, Eaton et al. (2015) identified and analyzed "tech blogs" containing content that helped them develop a theory around the evolution of PBRs. For this study's research question, this author sought to identify and collect articles that (1) focus on the use of Twitter data by 3PADs and/or (2) highlight conflicts or tensions between Twitter and the 3PADs who build on its platform.

To identify such articles, ProQuest's ABI/INFORM Global database was used first. This database enables access to content from indexed magazines, newspapers, trade publications, and scholarly journals. This search, conducted in September 2018, yielded 280 English-language, full-text articles containing "Twitter" in the title and any of the following strings in the abstract: "data"; "API"; "application programming interface"; "develop*"; and "policy" or "policies." These search parameters aimed at producing an inclusive corpus of articles. The resulting corpus spanned 2009 to 2018.

The next step entailed filtering out impertinent articles. Specifically, this author read each article's abstract – and in some cases most or all of the article – to determine whether the article discusses (1) Twitter's use of APIs or data regulations to affect data collection by 3PADs, (2) beliefs, attitudes, or opinions held by 3PADs about the collection of Twitter data, and/or (3) efforts made by 3PADs to influence how Twitter data are collected and/or how much Twitter data may be collected. Based on this review, only 33 of the 280 articles were deemed pertinent. None of the 247 other articles satisfied the aforementioned criteria; in most cases, these non-germane articles focused on the use by non-developers of Twitter data for analytics, on the use of

tweets by advertisers, on the use of Twitter for marketing purposes, or on various security and/or legal issues surrounding Twitter.

In accordance with this study's exploratory nature, a method designed to analyze Internet-based qualitative data was employed to analyze the 33 pertinent articles (Romano Jr., Donovan, Chen, and Nunamaker Jr., 2003). Drawing from this methodology, in the first (selection) phase each article in the corpus was examined in order to extract pertinent passages (i.e., multi-sentence strings of text). Next, each extracted passage was "tagged" with a label aimed at describing it at a very high level (e.g., "Twitter opened its platform," "Demand for third-party apps"). Each extracted passage also was tagged with the year in which it was produced.

Romano et al.'s (2003) second (coding) phase closely resembles the open and axial coding employed in grounded theory studies (Glaser, 1998). Through open coding, each passage extracted during the selection phase is categorized in a way that is mindful of categories already created (Gleasure and Feller, 2016). This process is commonly referred to as constant comparison (Glaser and Strauss, 1971) and aims at developing a comprehensive set of categories that captures the meaning of every passage. When open coding reaches categorical saturation, axial coding commences. Axial coding entails establishing hierarchical relationships between categories; thus, superseding categories represent 'axes' from which 'spokes' (sub-categories) extend (Glaser, 1988). For this study, top-level categories represented evolutionary stages of Twitter's digital platform, with second-level categories representing key events, decisions, or outcomes. For example, one top-level category was labeled "Aligning APIs with market strategy (2011-2013)." This category contained six second-level categories: "Twitter posts open letter to 3PADs"; "Twitter eliminates its whitelist"; "Twitter offers a Streaming API"; "API v1.1 is introduced"; "Implications of API v1.1 for 3PADs"; and "Outsourcing of data access management." Each of these categories, in turn, contained multiple third-level categories, some of which contained fourth-level categories.

Second Stage of Data Collection and Analysis

Romano et al.'s (2003) third (clustering) phase resembles the "theoretical coding" phase of the grounded theory method (Glaser, 1998). In the clustering phase, the analyst ties together all categories and sub-categories through one or more concepts aimed at addressing the study's research question. To accomplish this, all axial categories and their relationships must be well understood. However, because the corpus yielded by the first search was too small (n=33), this author was not able to perform clustering until certain informational gaps were filled. More specifically, further details around certain phenomena were needed, such as Twitter's whitelisting program, the exact means by which Twitter engaged in app gatekeeping, the response of 3PADs to API v1.1, API tiers (i.e., public vs. Premium vs. Enterprise), and Twitter's motives behind data throttling at different evolutionary stages.

Accordingly, Google's vast database of web documents was leveraged – as well as Google's custom date-range search functionality – to conduct a more fine-tuned search that could fill these gaps. Many search strings were prepared, each with the aim of filling a particular informational gap or answering some unresolved question. This second-phase search yielded 58 open-web articles. (A full reference for each of the 91 articles analyzed for this study is provided in Table 1 in the appendix.) These articles allowed this author to complete the coding phase and, in turn, perform and complete the clustering phase. During the clustering phase, the study's core

concepts emerged – namely, four principal data tuning actions and their seven variants – thus allowing this author to tie together (and cluster) the axial categories. The expanded corpus also improved informational accuracy by enhancing cross-checking for facts. To this end, every major, germane assertion contained in the following narrative is accompanied by a citation linked by number to its full reference in Table 1 in the appendix.

THE ROLE OF DATA PLATFORM BOUNDARY RESOURCES IN THE EVOLUTION OF TWITTER'S DIGITAL PLATFORM

Twitter Opens Its Platform to Generate More Functionality

Two months after its public launch on 15 July 2006, Twitter opened its platform to third-party application developers (3PADs) by offering a set of public (free) application programming interfaces (APIs) based on a representational state transfer (REST) architecture which enables the provision of web services through hypertext transfer protocols [55, 88]. These public APIs allowed 3PADs to execute GET, POST, PUT (i.e., update), and DELETE methods against user statuses, user timelines, lists of users, list of followers, saved searches, and more [67]. In addition, 3PADs could include various parameters in these methods in order to focus a search (e.g., by keyword or location), perform operations on behalf of a single user, view trending topics, and more [49]. Because of these affordances, 3PADs gained the ability to develop applications that could offer more and better functionality to Twitter's users. Thus, Twitter's public APIs promised to facilitate the expansion and enrichment of Twitter's platform by 3PADs.

Twitter chose to open its platform to 3PADs because it needed to focus its efforts on "making sure the core service worked and was scaling," where core service provision entailed the execution of basic functions such as the posting of tweets, replies, and retweets [5]. Core service provision was made difficult because of Twitter's rapidly growing scale of operations: over the period ranging from December 2007 to December 2008, the number of active Twitter users increased from 500,000 to 4.5 million [89]. Thus, for the first three or four years of its existence, at least, Twitter's need to focus on scaling operations meant that it lacked the internal capacity to develop new, innovative apps for its platform [54].

However, Twitter's executives knew that (1) Twitter's native interface could be improved and (2) better, more innovative apps were crucial to Twitter's growth [5]. Accordingly, "for the first four years or so of its existence, Twitter relied heavily on [3PADs] to provide extra functionality around its bare-bones microblogging service" [54]. During these first four years, Twitter encouraged 3PADs to develop any app that could "build on and improve the user experience," including apps that could provide an alternative interface for core services [21].

Despite its need for new, innovative third-party apps, Twitter's public API did not enable 3PADs to access to all of Twitter's data [14]. Specifically, older data (including tweets, retweets, replies, followers, searches, etc.) were simply not available to 3PADs, though just how far back in time a 3PAD could go depended on the API method and Twitter's operational capacity. According to one source, Twitter did not "make available tweets older than three weeks, [and] in some cases only one week" [43]. Another source stated that Twitter's public Search API "enabled searches of tweets [from] the past six to nine days" [14], while another stated that it returned "only the 20 most recent tweets" [58]. Twitter also prevented the overloading of its computing infrastructure by imposing "per-endpoint rate limits" [62]. Specifically, and per a

letter posted by Twitter to 3PADs, the public API limited "the number of requests [third-party] applications can make to 350 calls per hour, regardless of the type of information the application was requesting" [62].

Many 3PADs were frustrated with these limitations of Twitter's public API, which came to be known as "API v1.0.". Some technology writers argued that the amount of data that a 3PAD could retrieve was insufficient [2, 58, 61], with one writer stating that it "isn't going to be of much use" for app development [2]. Other tech writers were critical of limitations associated with API methods. As one tech writer pointed out, all of the replies to a tweet could not be retrieved, direct messages could not be threaded, and tweets could not be retrieved in bulk [61]. Taken together, these criticisms suggested that the limitations of API v1.0 diminished the creative potential of third-party apps.

Recognizing that these limits may be curtailing platform growth to some extent, Twitter offered "whitelisting" as an alternative to API v1.0. In short, a 3PAD wanting to exceed the aforementioned limits could submit a request to Twitter [47]. If granted, an app provided by the 3PAD could make up to 20,000 calls per hour (up from 350) [59]. In addition, whitelisted organizations had access to an API that enabled the collection of real-time (streaming) data that Twitter would not make publicly available until 2012 [59]. To the consternation of many 3PADs, though, Twitter did not enumerate the conditions needed for whitelist inclusion, and the reasons for granting some requests while denying others were not provided [47]. While the number of 3PADs who were whitelisted is not available, one source suggests that relatively few organizations were whitelisted [59].

Despite the limitations of API v1.0 and the exclusivity of the whitelisting program, 3PADs had developed and marketed 100,000 apps by August 2010 [8], compared to only 7,000 apps in 2007 [21]. Third-party apps available to Twitter users included apps that support user analytics (e.g., Twitterholic, Twist), search engines for tweets (e.g., Summize), URL shortening apps (e.g., TinyURL, bit.ly), picture sharing and management apps (e.g., TwitPic), apps that integrate Twitter with RSS feeds (e.g., TwitterFeed), and interfaces for core services such as profile management, tweet posting and replying, and tweet reading and scrolling (e.g., Twhirl, TweetDeck, Twitterrific, Dabr) [5, 37, 45, 46, 79]. Moreover, every major operating system – including Windows, Mac, iOS, and Android – was served by one or more apps from these categories [5].

As a result of the proliferation of new and innovative third-party apps, the Twitter platform had been extended and enriched. In short, 3PADs had, through their efforts, generated a great deal of additional functionality for the platform. Indeed, one of the more popular third-party apps, Twitterrific, was the creator of the renowned bird icon, the Twitter conversations feature, and the first iPhone client [65]. One tech writer noted that 3PADs "allowed [Twitter] to be on every mobile platform before it had an internal team building mobile apps. Even Twitter's search engine was built [using] Twitter's API" by a 3PAD [68]. This rich ecosystem of third-party apps also generated indirect network effects, meaning that it increased the rate of Twitter adoption by users [57].

While Twitter acknowledged the valuable contributions of third-party apps [55], Twitter also had become increasingly concerned about its ability to maintain a computing infrastructure capable of executing a very rapidly growing number of API calls while still providing core services effectively [39]. According to one Twitter engineer, by September 2010 Twitter was processing *six billion* API calls per day, or 70,000 calls per second [50, 51]. As a result of this growing concern, Twitter implemented two changes: first, it reduced the number of public API

calls that third-party apps can make per hour to 175 (from 350); and second, it outsourced the management of its streaming data, and its vast stores of historical data, to a company called Gnip [39]. As a result, whitelisted 3PADs and 3PADs willing to pay for premium data services had to work with Gnip, not Twitter. In addition to reducing the burden on its computing infrastructure, Twitter also received revenue from Gnip, which paid Twitter to be a "certified data reseller" [40].

Twitter Discourages the Development of Core User Experience Apps

On 11 March 2011, Twitter's Platform Director posted an open letter to 3PADs, titled "Consistency and Ecosystem Opportunities," on a Google Group forum for third-party Twitter developers [55]. In this letter the Platform Director stated the following: "Our research shows that consumers continue to be confused by the different ways that a fractured landscape of third-party Twitter clients display tweets and let users interact with core functions... If there are too many ways to use Twitter that are inconsistent with one another, we risk diffusing the user experience" [55].

Given this need for a "consistent user experience," Twitter determined that it would provide, "by itself or in exclusive partnerships" [54], the "primary mainstream consumer client experience on phones, computers, and other devices" [55]. Accordingly, the open letter warned 3PADs to not develop applications that "mimic or reproduce" this experience [55]. To discourage such development, the Platform Director stated that Twitter reserves the right to "revoke API tokens" – which allow a third-party app's users to use the app – from offending apps [55].

Twitter proceeded to develop some *core user experience apps* internally while acquiring others. For example, in 2011 Twitter introduced a new native feature called Lists (which lets users segment their followers) [7] and developed its own URL shortener while "threatening to deny all others" [79]. And in 2012 Twitter introduced its own filters and editing tools for pictures and a new feature that "allows users to e-mail a tweet to someone who is not on Twitter" [17]. From 2010 through 2012, Twitter's acquisitions of core user experience apps included Summize (which became Twitter Search) [79], Tweetie (the most popular iPhone client at the time) [8], and an application from atebits that became Twitter for iPad [60]. By the end of 2011, Twitter had also launched official apps for the Android, Blackberry, and Windows Phone operating systems [60].

A spate of articles from tech writers argued that Twitter's move to gain the bulk of the market share for the core user experience was driven by Twitter's push to become a publicly-owned company [8, 12, 19, 54] – a push made possible by explosive user growth from 2008 to 2011. In order to appear attractive to public investors, though, Twitter needed to demonstrate that it could generate substantial revenues through advertisement sales [55]. Ad revenues were contingent on users visiting the Twitter site or using the native app, though. As one tech writer put it, "It's pretty clear why Twitter is doing this. If you're not using their application, they're not getting revenue from promoted tweets. They want you to come onto their site, whether it's through a mobile app or the web. That's how they monetize the platform" [17].

Thus, by discouraging or preventing the development of core user experience apps by 3PADs – by throttling the amount of data they can collect and by acquiring and promoting the most popular third-party apps – Twitter effectively eliminated much of its third-party competition [13, 26]. In turn, many 3PADs "complained [that] Twitter had pivoted from

benevolent platform to aggressive competitor" [8]. Many of them also vowed to stop developing for Twitter; as one lead developer stated, "Why would I invest time and effort in enhancing an app that Twitter is going to disallow?" [54]. Twitter was not apologetic, though. As Twitter's CEO, Evan Williams, put it: "To developers nervous that the platform for which they are building software might compete with them... that's part of the game" [8].

In April 2011, one month after posting its open letter to developers, Twitter eliminated its whitelist [59]. As a result, all 3PADs wanting to exceed the public API rate limits or access historical data now had to purchase the data through Gnip. One *Wall Street Journal* writer speculated that Twitter eliminated its whitelist because (1) the process of reviewing whitelisting requests consumed too much of its time, (2) some rejected organizations became angry with Twitter, and (3) Twitter was beginning to recognize that its user data was potentially a source of great value [10]. As this writer put it, Twitter had become "wary about sharing its data for free" [10].

Twitter Introduces API v1.1

In August 2012, more than 16 months after Twitter posted its open letter to developers, it announced the deployment of "API v1.1." [17]. With this update, Twitter's public, REST-based APIs offered the same methods used with API v1.0, for the most part [63]. Unlike API v1.0, however, Twitter also enabled public access to streaming (real-time) data, although studies conducted at the time found that 3PADs received only 1 percent to 40 percent of near real-time tweets, depending on the API method and demand [67]. Somewhat confusingly, Twitter referred to this new public API as its Streaming API. The service that had (prior to API v1.1) been referred to as the Streaming API – a service which allowed paying 3PADs (and whitelisted organizations before that) to access 100 percent of streaming tweets – was now referred to as Twitter's Enterprise API [67]. Gnip continued to administer this premium service, but Twitter also authorized a second data reseller (DataSift) to administer it as well [63].

Twitter's API v1.1 differed from API v1.0 in that it required open authentication (OAuth) on every API endpoint [62]. With API v1.0, 3PADs "simply needed to authenticate [by] using the username-password combination of their regular Twitter account" [79]. With API v1.1, though, access to data required adherence to OAuth protocols that mask the password of the accessing agent. While the adoption of OAuth by Twitter aimed ostensibly at "preventing the malicious use of the API" [62], at least one tech writer argued that Twitter adopted OAuth in order to "know exactly who is accessing [the API]" [72].

API v1.1 also further reduced the number of requests that a third-party app could make. As stated in official Twitter documentation, "most API endpoints [in v1.1] will be rate limited at 60 calls per hour," down from 175 to 350 calls per hour in API v1.0 [62]. With regard to historical data, Twitter allowed 3PADs to obtain only the most recent 3,200 tweets, which in some cases was an improvement over API v1.0 [58]. Nevertheless, organizations still had to pay relatively large sums of money – and work with directly Gnip or DataSift – in order to access all of Twitter's historical data.

Finally, API v1.1 was accompanied by a set of "Rules of the Road" for 3PADs [57]. The first major function of these rules was to specify the types of apps that 3PADs were encouraged to develop. More specifically, 3PADs were once again – as with the open letter Twitter posted one year before – explicitly warned against developing core user experience apps [17]. Given the uncertainty around what, exactly, constitutes a core user experience app, Twitter provided some

clarity by identifying five broad types of acceptable apps: social CRM (i.e., customer relationship management); enterprise; media integration; analytics; and social influence ranking [57, 62]. Per Twitter's CEO, 3PADs that develop these types of apps demonstrated that they "want to work with Twitter, not against it" [55].

The second major function of these "Rules of the Road" was to identify another mechanism for deterring 3PADs from developing core user experience apps [17]. Specifically, Twitter stated that developers of such apps would be limited to "100,000 tokens on connected users," at which point the 3PAD "must get permission and work with Twitter directly" [64]. Any app that violated this rule would not be allowed to "take on any more users" [64]. Along these lines, Twitter introduced the Twitter Certified Program shortly after launching API v1.1. With this program, a 3PAD would submit its app ("as soon as [the app] gets serious") to Twitter for approval. Twitter's rejection of the app – most likely because it replicates the core user experience – would lead Twitter to "limit how much data it can retrieve and/or how many users (tokens) it can serve" [74]. As one tech writer put it, Twitter was now "playing emperor and giving thumbs up or down to any third-party app" [72].

Twitter stated that its desire to "prevent the malicious use of its service" was the chief impetus for promulgating these "Rules of the Road" [62, 74]. Many tech writers argued that this was not the case, though; as they contended one year before, Twitter's move to control the core user experience through its own apps was based on its need to demonstrate an ability to generate ad revenue [28, 65]. Fourteen months after launching API v1.1, Twitter's initial public offering was accepted on 7 November 2013 at US$26 per share.

Many 3PADs once again expressed their disappointment and frustration with Twitter's actions, mostly through tech articles and discussion forums. Many 3PADs were compelled to revise the value proposition of their apps (e.g., Echofon, Storify, Tweetbot) while others opted to cease operations (e.g., Favstar.fm) [26, 62]. One tech writer wrote that "being a [third-party] share-cropper is a fool's paradise – fun while it lasts but ultimately doomed" [19], while one investor stated that he is "hesitant to invest in a company that is linked to Twitter because of [Twitter's] treatment of developers" [21].

Twitter Further Monetizes Its Platform Data

Roughly the same time that Twitter was rolling out API v1.1, it expanded its list of certified data resellers to include Dataminr and NTT Data [15, 73]. As explained in one tech article, "each month [these resellers] pay Twitter for access to its firehose... then turn around and sell access to companies like Klout, Dell, IBM, and Oracle" [69]. In 2013, Twitter earned nearly US$50 million by licensing its data to certified data resellers [66].

Shortly thereafter, in early 2014, Twitter acquired Gnip, which had been administering Twitter's Enterprise API since August 2012 [88]. In doing so, Twitter internalized Gnip's data processing and analytics capabilities, which in turn enabled it to "mine insights from its data and monetize the insights... [thus] giving Twitter the ability to sell targeted advertising" [73]. Thus, Twitter was investing in its belief that targeted advertising represented a key means of generating substantial revenues [88].

A bit later, in October 2014, Twitter launched a mobile app development kit called Fabric [21]. Fabric's features – aimed at making it easier for 3PADs to build Twitter-based apps – included a crash monitoring and reporting tool (Crashylitics), an app distribution and tracking tool (Beta), an analytics dashboard (Answers), and a tool that lets 3PADs embed tweets in their

apps (TwitterKit) [22, 24, 79]. Twitter's publicly stated reason for offering Fabric was to "regain developers' trust and attract more app makers" [21]. Many tech writers questioned whether it could ever accomplish the former, given the history of Twitter's treatment of 3PADs [32]. Twitter's actual objective with Fabric, they argued, was to "insert Twitter" in as many apps as possible After all, if many 3PADs were to adopt Fabric as their primary mobile SDK, then its Twitter-centric features may lead to the integration of Twitter in apps that were not Twitter apps per se [90].

Demand for Twitter's data grew rapidly over the ensuing months. In the first quarter of 2015, Twitter "earned US$47 million in data licensing revenue, a 107 percent increase" from the first quarter of 2014 [29]. As a result, data licensing now comprised 15 percent of Twitter's revenues, up from 11 percent one year before [83]. Over the same period, Twitter's financial projections for targeted advertising revenues were not met [77]. Accordingly, by the summer of 2015 there were reports that Twitter was reconsidering its plan to focus more on targeted advertising than on data licensing [75, 83]. Indeed, one reporter wrote that, "In place of agreements with resellers such as DataSift and NTT Data, Twitter instead plans to sell access to its firehose data directly via its own API set... Twitter's motives are plain enough. The company is determined to generate more revenue by turning its data into a licensable resource for real-time sentiment analysis" [77].

Reports on Twitter's plans to focus more on data sales proved to be accurate, although it was not until early 2017 that Twitter announced plan to offer "Premium APIs." Initially, only one Premium API – a "Search Tweets Premium API" – was introduced to the market [82, 83]. This API provided access to "the past 30 days of Twitter data," offered "more tweets per request and higher rate limits" than the public Search API, and "supported more complex queries" [82]. Twitter announced that it planned to offer, "in the near future," another Premium API that "will enable access to the full history of Twitter data" [82].

Twitter's launch of this Premium API meant that three tiers of APIs were now available to 3PADs: free, public APIs; the new Premium API; and the Enterprise API, which had been administered by Twitter since 2014. The Enterprise API was Twitter's "top of the line" API, and was cost-prohibitive for most organizations. As noted in one tech article, 3PADs had to "pony up thousands of dollars per month to get on board. BlitzMetrics' CTO blogged this month that his company had to pay US$5,000 to US$20,000 per month for access" [78]. Twitter's new Premium API represented a "market somewhere in the middle," between the Enterprise API and the public APIs: "Pricing for the Premium API starts at US$149/month [up to US$2,499/month] and includes a free [development] sandbox... We [i.e., Twitter] are also introducing a new self-serve developer portal that gives you more transparent access to your data usage" [82].

At the same time, Twitter expanded its public API offerings to include the Media, Collections, Curator, Ads, Account Activity, and Direct Message APIs [80, 85]. Twitter's Streaming API, which provided only "a random sampling of one out of every 10 [real-time] tweets," remained publicly available, thus signaling that "Twitter's API monetization will remain tiered, with the most useful tiers also being the more expensive ones" [77]. Finally, in 2017 Twitter sold Fabric (i.e., its mobile SDK) to Google. Fabric had not been adopted to the extent that Twitter had hoped [85]. As a result, the monetization of data via Premium and Enterprise APIs became an even higher priority for Twitter [91].

ANALYSIS OF DATA TUNING BY TWITTER AND THIRD-PARTY APPLICATION DEVELOPERS

Drawing from the preceding narrative of the role played by data platform boundary resources (PBRs) in Twitter's evolution, Table 2 (in the appendix) details fifteen (15) instances in which Twitter or third-party application developers (3PADs) attempted to tune a data PBR. For each of these instances, Table 2 identifies the data tuning agent and data tuning action, describes in brief the data tuning instance, states the goals of the data tuning action, and indicates whether the goals were accomplished or not.

An analysis of Table 2 suggests that Twitter evolved from start-up to digital platform behemoth through four main stages. Moreover, attempts by Twitter to tune data PBRs figured prominently in this evolution. In the first stage (2006 to 2009), Twitter tuned data PBRs through data provision (see Tuning Instances 1 and 3 from Table 2). Specifically, Twitter opened its platform by offering public (free) APIs that allowed 3PADS to retrieve a portion of data generated by Twitter's users. Twitter's hope was that 3PADs would use the data to develop new, innovative apps, thus allowing Twitter to focus less on internal app development and more on delivering and scaling core services. By mid-2009 there was little doubt that Twitter's aims had been achieved, with roughly 60,000 third-party apps [8] – many of them providing functions that would come to define Twitter – to a base of more than 10 million global users (Statista, 2018).

Twitter had anticipated growth, but not to that extent and not so quickly. Indeed, by mid-2010, third-party apps were making more than five billion calls per day to Twitter's APIs [51], and a rapidly growing number of 3PADs were submitting requests to be whitelisted (i.e., approved for access to Twitter's historical data and real-time data stream). The second stage of Twitter's evolution (2010 to 2011) can be understood in terms of its twofold response to these demands on its internal resources. First, Twitter outsourced the management of access to its historical data and real-time data stream to a company called Gnip (Tuning Instance 4). This meant that whitelisted 3PADs and 3PADs who were willing to pay for premium data services had to start working with Gnip and not Twitter; accordingly, Twitter was able to continue focusing on the delivery and scaling of core services. Thus, Twitter tuned data PBRs through data outsourcing. Twitter also responded to demands on its internal resources by tuning data PBRs through data limiting (Tuning Instance 5). In this instance, Twitter limited the amount of data that third-party apps could access through its public API – from 350 calls per hour to 175.

During the second stage of Twitter's evolution, Twitter realized that (1) its data were more valuable than it had thought and (2) its remarkable growth meant that becoming a publicly-traded company was a viable possibility. These realizations shaped the third stage of Twitter's evolution (2011 to 2012), which centered on its drive to appeal to public investors. In this stage, Twitter engaged in data limiting across four instances of data tuning. In the first instance (Tuning Instance 6), Twitter informed 3PADs that it would henceforth provide the "core user experience" through its own apps, which would be acquired or developed internally. (Twitter reasoned that it could generate advertising revenues by providing such apps, thus improving its initial public offering.) To discourage the development of such apps by 3PADs, Twitter began to throttle the amount of data that a violating app could retrieve.

In the second instance of data limiting in the third stage (Tuning Instance 8), Twitter eliminated its whitelist, which meant that all 3PADs wanting to access historical data or exceed the public API rate limits had to purchase data through a certified reseller. In the third instance (Tuning Instance 10), Twitter further limited the number of calls per app per hour (from 175 to

60) in order to encourage 3PADs to purchase data through Twitter's Gnip-managed Enterprise API. And in in the fourth instance (Tuning Instance 11), Twitter placed a cap (100,000 connected users) on the degree to which a third-party app could scale if it was found to be replicating a core user experience. As with Tuning Instance 6, the goal of this tuning action was to discourage the development of such apps by 3PADs.

Thus, in the third stage of its evolution, Twitter engaged in data limiting twice for the purpose of app gatekeeping and twice for the purpose of data monetization. Twitter also engaged in data provision for the purpose of data monetization (Tuning Instance 9) when it enabled public access to a portion of its real-time data stream. In doing so, Twitter furthered its tiered data access strategy, with APIs segmented into a basic (free) tier and one or more paid tiers. All of the tuning attempts in this stage helped Twitter demonstrate its ability to generate an ongoing stream of revenues and, in turn, appeal to public investors.

In the fourth and final stage of Twitter's evolution (2013 to 2017), Twitter engaged in data monetization in three major instances of data tuning. In the first instance (Tuning Instance 13), Twitter licensed its data to two more companies (i.e., Dataminr and NTT Data). In the second instance (Tuning Instance 14), Twitter acquired Gnip in order to internalize Gnip's data mining and analytics capabilities and, in turn, create and market more personalized advertisements. In the third instance (Tuning Instance 15), Twitter introduced Premium APIs that provide access to more data than the public (free) API but much less data than the Enterprise APIs. (Premium APIs were priced accordingly.) In doing so, Twitter further advanced its strategy to offer tiered data access to 3PADs.

Data Tuning by Third-Party Application Developers

As Table 2 (see appendix) shows, third-party application developers (3PADs) made three major data tuning attempts during the study period. In all three instances, their attempt was tied to the publication of numerous pro-3PAD technology articles. Nearly all of these articles – which were published in numerous outlets, including The Wall Street Journal, Information Week, Network World, TechCrunch, Wired, and Mashable (see Table 1 in the appendix) – criticized tuning actions by Twitter. In their first attempt (Tuning Instance 2), 3PADs criticized Twitter's API v1.0 for being too limiting in terms of the amount of data one can access and the ways in which it can be accessed. The goal of this tuning attempt was to persuade Twitter to (1) provide access to its historical data, (2) ease data rate limits, and (3) offer more API methods. While Twitter responded by adding some API methods and creating the whitelisting program, it did not ease rate limits for non-whitelisted 3PADs or offer them access to historical data.

In their second attempt to tune Twitter's data PBRs (Tuning Instance 7), 3PADs used tech articles to criticize Twitter for discouraging the third-party development of core user experience apps. The upshot of these articles was that Twitter's actions were grossly unfair to 3PADs who had invested in developing such apps. As with their first tuning attempt, 3PADs hoped to persuade Twitter to make changes to its APIs and its data regulations. In addition, though, 3PADs hoped to persuade Twitter to not undermine 3PADs in the future while persuading other 3PADs to stop developing for Twitter. None of these goals was accomplished, though: Twitter carried out its plan to provide the bulk of the core user experience, and many 3PADs continued to develop complementary apps for Twitter.

Finally, in their third tuning attempt (Tuning Instance 12), 3PADs used tech articles to express their frustration – after Twitter launched API v1.1 – at taking large losses on investments

on apps deemed by Twitter to replicate the core user experience. By this point, 3PADs seemed to acknowledge that public pleading would not lead Twitter to reverse its position; accordingly, 3PADs' goal with this tuning attempt was to persuade other 3PADs to stop developing for Twitter. Once again, their goal was not accomplished, though, as many 3PADs simply shifted their focus to the types of sanctioned apps that Twitter had identified.

This analysis raises an important question: Why did 3PADs only attempt to tune data PBRs through the publication of tech articles?[1] The answer is that there may have been no other legitimate means of tuning Twitter's data PBRs. In Eaton et al.'s (2015) case study of the tuning of Apple's iPhone, hackers worked to jailbreak the iPhone's kernel module after each and every hack-defending update from Apple. With regard to this Twitter case study, though, no efforts by 3PADs to hack Twitter's API and download more data were reported. If any such hacking attempts were actually made, they were stopped by Twitter. To date, it appears that no script for the hacking of Twitter's API was ever publicized.

Moreover, while some 3PADs created programming language-specific libraries for use in retrieving data through Twitter's public API, such resources did nothing to reduce rate limits or provide access to historical data. Instead, they merely made it easier for 3PADs to retrieve such data. Thus, such efforts cannot be understood as attempts to tune data PBRs. Ultimately, given that Twitter did not make its public API fully open, only Twitter could take such actions (hacking attempts excepted). Indeed, the other three ways in which 3PADs have been known to exploit PBRs – forking, cloning, and substituting (Karhu and Gustafsson, 2015) – pertain only to open PBRs. The problem for 3PADs, then, centered on how to convince Twitter to make its APIs beneficial to 3PADs even when doing so might not be as beneficial to Twitter.

Finally, it is important to emphasize that in two instances (Tuning Instances 2 and 3) 3PADs used tech articles to persuade other 3PADs to stop developing for Twitter. For the disgruntled 3PADs, then, the hope was that enough 3PADs would leave the Twitter platform to cause Twitter to meet 3PADs' demands, or at least to reach a compromise with 3PADs. Thus, a withdrawal from the Twitter platform by a large number of 3PADs could have proved to be a fruitful means of tuning Twitter's data PBRs, but such a withdrawal never happened. Instead, a sufficient number of 3PADs continued to develop for Twitter despite repeated tuning actions by Twitter that affected many 3PADs adversely.

KEY FINDINGS AND CONCEPTUAL IMPLICATIONS

The preceding narrative and analysis of Twitter's digital platform illustrates how Twitter's attempts to tune its data platform boundary resources (PBRs) – that is, its PBRs that enable and aid 3PADs in the collection of platform data – helped shape the evolution of its digital platform. More specifically, Twitter performed four principal data tuning actions – namely, data provision, data outsourcing, data limiting, and data monetization – across 12 data tuning instances and four evolutionary stages. Based on these four principal data tuning actions, seven variants of data tuning can be identified. Each of these seven data tuning variants,

[1] The analysis also reveals that nearly all germane tech articles advocated for the interests of 3PADs and not Twitter. Based on this author's reading of more than one hundred such articles, three main factors may explain this phenomenon: first, many tech writers also develop third-party apps, either professionally or recreationally; second, most tech publications promote a market ideology of open innovation; and third, a typical tech publication's audience includes substantially more 3PADs than tech executives. Consider, for example, that 3PADs had developed and marketed 100,000 Twitter apps by August 2010, while Twitter employed roughly 3,000 workers in 2010 (Statista, 2018).

identified here in turn, consists of a principal data tuning action (e.g., data provision) and a goal (e.g., to add functionality).

1. Data provision to add platform functionality. Through its API v1.0 and its whitelisting program, Twitter engaged in data provision to generate third-party functionality for its digital platform (Tuning Instances 1 and 3).
2. Data limiting to reduce demands on resources. Rapidly growing demands on its human and computational resources compelled Twitter to limit the amount of data that 3PADs could retrieve (Tuning Instance 5).
3. Data limiting to gatekeep apps. In order to demonstrate to public investors its ability to generate revenues through advertisements, Twitter needed to control the core user experience. Accordingly, Twitter targeted violating third-party apps by throttling the amount of data they could retrieve and by limiting their number of connected users (Tuning Instances 6, 10, and 11).
4. Data monetization to generate revenue. Twitter engaged in data monetization by licensing its to resellers (Tuning Instance 13); by selling it directly to 3PADs (Tuning Instance 15); and by selling customized ads made possible by data mining (Tuning Instance 14).
5. Data outsourcing to monetize. In Stage 2, Twitter needed to reduce the growing demand on its human and computational resources. To do so, Twitter outsourced the management of access to its most computationally-intensive data (i.e., its real-time data stream and historical data) to Gnip (Tuning Instance 4). Even though Twitter did so mostly out of operational necessity, it was able to monetize these data by collecting licensing fees from Gnip.
6. Data provision to monetize. In Stage 3, Twitter needed to demonstrate to public investors its ability to generate revenues. One way it did so was by actually providing free access to a sampled portion of its real-time data stream (Tuning Instance 9). By doing so, Twitter introduced a program of tiered data access in which each major API type was segmented into a free tier and one or more paid tiers. Twitter reasoned that some 3PADs retrieving a sample of data would want to retrieve more of it, and would thus pay for more of it.
7. Data limiting to monetize. Twitter also predictably demonstrated its ability to generate revenues by monetizing its data. In Tuning Instance 8, though, it monetized data by actually limiting it. Specifically, by eliminating its whitelisting program Twitter effectively forced all 3PADs – including those who had been whitelisted – to purchase data through a certified reseller.

How Twitter's Data Tuning Actions Fueled Its Success

The analysis demonstrates that Twitter effectively controlled and improved its platform ecosystem throughout the period of study (2006 to 2017). Indeed, due in large part to its data tuning actions, Twitter was successful inasmuch as it:

- Generated substantially more platform functionality via third-party apps;
- Reduced the demands on its human and computational resources in order to reduce costs and focus on delivering and scaling its core service;
- Demonstrated to public investors its ability to generate revenues; and
- Monetized data by licensing it, mining it to create customized ads, and selling it directly to third parties.

The analysis suggests that Twitter's success was a function of the timing of its data tuning actions, which in turn was a function of its evolving needs. For example, from 2006 through 2010, Twitter needed to generate more functionality on its platform but lacked the internal resources to do so. Thus it made sense for Twitter to open its platform and let 3PADs develop and market Twitter-based apps, even if such apps provided the core user experience. Indeed, it appears that during this period Twitter prevented 3PADs from accessing all of its data *not* because it wanted to limit the value and appeal of third-party apps, but rather because it lacked the requisite computational resources.

As another example, Twitter likely determined in 2011 that its need for new, innovative third-party apps was not as strong as it had been during the preceding five years. After all, 3PADs had already created much of the essential functionality. As a result, Twitter started taking steps to exclusively provide the core user experience – including, and most notably, limiting the amount of data that a violating app could retrieve – even though such steps would most likely lead to a general decrease in the production of third-party apps. By this time, though, Twitter had been able to develop its human and computational infrastructure and thus could devote more resources to internal app development.

It is important to consider, though, that while Twitter's data tuning actions helped fuel its success, the same actions could have backfired on Twitter if many 3PADs had responded by withdrawing from the Twitter platform. As discussed in the preceding section, numerous tech articles called for 3PADs to stop developing for Twitter in response to the negative impacts (e.g., data limiting, app gatekeeping) caused by Twitter's data tuning actions. Indeed, Twitter's success during the study period may be attributed in large part to not having to reverse or even moderate its data tuning actions, which they might have been compelled to do if 3PADs had withdrawn in sufficiently large numbers.

However, it is worth asking whether Twitter's success in the short term came at the expense of longer-term success. As one tech writer put it, Twitter's app gatekeeping and data monetization actions may have done "long-term damage to Twitter itself as much as to any of its partners... This was an innovation-destroying idea, as less experimentation took place... All of which decreased value to customers" [77]. In other words, the data tuning actions that fueled Twitter's short-term success may have effectively placed a ceiling on Twitter's growth, thus raising the possibility that Twitter's platform could have been comparable to Facebook's platform in terms of size and market value.

Conceptual Implications

This study extends the PBR literature in two main ways: first, by proposing the concept of "data PBRs" (i.e., PBRs that enable or aid 3PADs in the collection of platform data); and second, by exploring how certain data PBRs (i.e., APIs and data regulations) are contested and re-shaped by digital platform providers and 3PADs in order to further their respective interests and shape the digital platform's evolution. The second contribution extends the concept of PBR tuning (Eaton et al., 2015) by introducing the concept of "data tuning" and proposing four principal data tuning actions: data provision; data outsourcing; data limiting; and data monetization. These four actions serve as the basic components of data tuning.

Based on these four principal data tuning actions, many variants of data tuning may be identified. Seven such variants were identified from this case study of the evolution of Twitter's

digital platform. Of these seven variants, four are generally aligned with the PBR literature in terms of the behaviors of digital platform providers:

- Data provision to add functionality to the platform (e.g., Ghazawneh and Henfridsson, 2013);
- Data limiting to reduce demands on resources (e.g., Karhu and Gustafsson, 2015);
- Data limiting to gatekeep apps (e.g., Mohagheghzadeh and Svahn, 2016); and
- Data monetization to generate revenue (e.g., Wulf and Blohm, 2015).

The other three variants, on the other hand, represent behaviors by digital platform providers that have not been addressed in the PBR literature. First, Twitter engaged in data outsourcing in order to monetize data when it outsourced the management of access to its historical and streaming data to Gnip. Twitter did so primarily to reduce demands on its human and computational resources, but licensing these data for resale meant that Twitter also monetized these data.

Second, Twitter engaged in data provision in order to monetize data (data provision to monetize) when it made available small samples of its real-time data stream for free. By doing so, Twitter introduced a program of tiered data access aimed at persuading 3PADs to pay for premium data services after sampling the free (basic) data services. Third, Twitter engaged in data limiting in order to monetize (data limiting to monetize) when it eliminated its whitelisting program. As a result of this action, Twitter forced all 3PADs, including those who had been whitelisted, to purchase data through a reseller such as Gnip or Dataminr.

Other findings around data PBRs also have conceptual implications. First, the analysis revealed that Twitter engaged in acts of securing (through data limiting) and resourcing. With regard to resourcing, Twitter made some of its data accessible to 3PADs via APIs and whitelisting (data provision). Twitter also engaged in resourcing through data monetization, though. Specifically, historical data and the real-time data stream were made available to 3PADs, but not for free; instead, 3PADs had to pay a certified data reseller or (later on) pay Twitter directly in order to retrieve these data. Accordingly, one can distinguish between free (basic) and for-pay (premium) APIs as data PBRs. An important implication of this distinction is that unless a 3PAD pays for more data, the degree to which it can scale its app(s) may be limited. Depending on the intensity of competition, a 3PAD that depends on free data may have to discontinue its app(s). Thus, a digital platform provider may also engage in data monetization for securing purposes.

Second, the findings suggest that this case study provides another empirical example of boundary resource dependency. According to Rafiq et al. (2013, p. 208), boundary resource dependency refers to how the modification of a PBR by a digital platform provider may "directly affect third-party app development and maintenance, which in extreme cases can affect the implementation and/or working" of affected apps. As this analysis revealed, Twitter's data tuning actions had a profound and detrimental impact on many 3PADs, particularly those that developed core user experience apps. Affected 3PADs had two options: first, settle for and deal with the limitations imposed by Twitter (i.e., ceilings on the number of users and the amount of retrievable data); or second, discontinue the development of violating apps. In many cases, affected 3PADs shifted their operations to the development of apps that Twitter sanctioned.

Third, Twitter's efforts to weaken core user experience apps support Mohagheghzadeh and Svahn's (2016) contention that resourcing can pose substantial risks to digital platform providers. Specifically, Mohagheghzadeh and Svahn (2016) argued that because PBRs comprise internal resources made public, the provision of PBRs by a digital platform provider creates risk

related to the loss of competitive advantage. Twitter recognized that by allowing 3PADs to retrieve larger amounts of data, some 3PADs would try to provide the core user experience and, in turn, become a competitor. Accordingly, Twitter throttled data to such apps and limited the number of users who could retrieve data from them.

Fourth, and finally, Mohagheghzadeh and Svahn (2016) argued that resourcing may be more effective – in terms of promoting participation by 3PADs – when digital platform providers collaborate with 3PADs through "continuous interactions" aimed at meeting 3PADs' data retrieval needs. In this case, though, there is no evidence that Twitter ever formally enlisted 3PADs as co-designers of data PBRs. Of course, helping 3PADs get all their desired affordances from data PBRs creates risk for digital platform providers. Thus, Twitter had to consider whether the risks (i.e., potential loss of competitive advantage) of open data PBRs outweighed the rewards (i.e., more innovative third-party apps for Twitter's users). The findings suggest that Twitter tended to protect its data (as a strategic asset) much more often than it made it freely available.

Directions for Future Research

Future research into data PBRs can extend knowledge about data PBRs *per se* or about data tuning actions and their variants. With regard to the former, there may be more data PBRs beyond the six identified here (i.e., free APIs, premium APIs, data regulations, data object descriptions, tech support for data collection, example endpoints). Thus, additional case studies of digital platform providers could expand the typology of data PBRs.

Due to the nature of this study's research question – how are data PBRs tuned? – only APIs and data regulations were investigated, as the other data PBRs are not instrumental for data tuning. However, other research questions could center on "support data PBRs" such as data object descriptions, tech support for data collection, and example endpoints. For example, how important are these and other support data PBRs to the promotion and generation of third-party apps? Do qualitative differences across digital platforms affect participation by 3PADs?

Future case studies could also expand the typology of data tuning actions and their variants. Such case studies could examine (1) digital platform providers (like Twitter) that wield tight control over their APIs and data regulations and/or (2) digital platform providers that offer open APIs and permissive data regulations. A case study that compares these two types of digital platform providers may reveal significant differences in terms of the data tuning actions taken. For instance, digital platforms that are more open may enable cloning, forking, and/or substituting efforts by 3PADs (Karhu and Gustafsson, 2015), which could in turn have a profound impact on the digital platform's evolution. With regard to this study, Twitter's tight control over APIs and data regulations greatly limited the ways in which 3PADs could engage in data tuning.

Further, survey-oriented research could extend this study's findings by determining the frequency with which data tuning actions and their variants are performed by providers of open and closed digital platforms. Such research could also address the question of whether certain data tuning actions tend to be performed at similar stages of the digital platform's evolution or under similar circumstances. Ultimately, though, the greatest research need may be to relate data tuning actions to outcomes desired by digital platform providers and 3PADs. For example, which data tuning actions, performed at certain stages or in certain contexts, are most effective at increasing participation by 3PADs, or at involving 3PADs as co-designers of data PBRs? This

case study demonstrated that Twitter's data tuning actions were largely successful – at least in the short term – but it could be that the same data tuning actions, taken by another digital platform provider operating in a different context, would not yield success or would shape the digital platform's evolution in a very different way.

CONCLUSION

Most digital platform providers enable and leverage third-party application development in order to add functionality to their platform and, in turn, attract users. Competition for third-party application developers (3PADs) is often intense, though. One way that digital platform providers try to attract 3PADs to their platform is by providing them with platform boundary resources (PBRs) such as application programming interfaces (APIs), software development kits, and favorable developer agreements. These PBRs are used mostly to support the work of 3PADs, but digital platform providers also may use them restrictively.

This study aimed at filling a notable gap in PBR research, namely, the absence of a conceptualization of the provision and regulation of platform data. To do so, the concept of a "data PBR" (i.e., a PBR that aids 3PADs in the collection of platform data) was introduced. This study then extended Eaton et al.'s (2015) concept of PBR "tuning" by exploring how digital platform providers and 3PADs attempt to further their respective interests by contesting and re-shaping (i.e., tuning) data PBRs. Twitter's digital platform and ecosystem (2006 to 2017) was examined in detail as a revelatory case.

An analysis based on grounded-theory techniques revealed four principal data tuning actions performed by Twitter: data provision; data outsourcing; data limiting; and data monetization. From these four principal actions, seven specific variants were identified, including data provision to add functionality, data monetization to generate revenue, and data limiting to monetize. Three of these seven variants represent behaviors not identified in the PBR literature. Additional conceptual implications were considered, such as why attempts by 3PADs to tune data PBRs were unsuccessful, and how Twitter's apparent success may have limited its longer-term growth.

In a recent article, de Reuver, Sørensen, and Basole (2018) called for further theorizing in digital platform research, including theorizing into the design and delivery of services such as PBRs. By introducing the term "data PBR," and by proposing a typology of data tuning actions by digital platform providers and 3PADs, this study serves as a response to this call and provides a foundation for future research into data PBRs.

REFERENCES

Barrett, M., Oborn, E., Orlikowski, W., and Yates, J.A. (2012). Reconfiguring boundary relations: robotic innovations in pharmacy work. *Organization Science*, 23(5), 1448-1466.

Barrett, M., Oborn, E., and Orlikowski, W. (2017). Creating value in online communities: the sociomaterial configuring of strategy, platform, and stakeholder engagement. *Information Systems Journal*, 27(4), 704-723.

Benlian, A., Hilkert, D., and Hess, T. (2015). How open is this platform? The meaning and measurement of platform openness from the complementor's perspective. *Journal of Information Technology*, 30, 209-288.

Bergman, M., Lyytinen, K., and Mark, G. (2007). Boundary objects in design: an ecological view of design artifacts. *Journal of the Association for Information Systems*, 8(11), 546-569.

Ceccagnoli, M., Forman, C., Huang, P., and Wu, D.J. (2012). Cocreation of value in a platform ecosystem: the case of enterprise software. *MIS Quarterly*, 36(1), 263-290.

Ceccagnoli, M., Forman, C., Huang, P., and Wu, D.J. (2014). Digital platforms: when is participation valuable? *Communications of the ACM*, 57(2), 38-39.

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

Choudary, S.P. (2015). *Platform Scale: How an Emerging Business Model Helps Startups Build Large Empires with Minimum Investment*. Cambridge, MA: Platform Thinking Labs Pte. Ltd.

Cusumano, M. and Gawer, A. (2002). The elements of platform leadership. *Sloan Management Review*, 43(3), 51-58.

Dal Bianco, V., Myllärniemi, V., Komssi, M., and Raatikainen, M. (2014). The role of platform boundary resources in software ecosystems: a case study. *Proceedings of the 2014 IEEE/IFIP Conference on Software Architecture*, Sydney, New South Wales, Australia, April 7-11, 2014.

De Reuver, M., Sørensen, C., and Basole, R.C. (2018). The digital platform: a research agenda. *Journal of Information Technology*, 33(2), 124-135.

Eaton, B., Elaluf-Calderwood, S., Sørensen, C., and Yoo, Y. (2015). Distributed tuning of boundary resources: the case of Apple's iOS service system. *MIS Quarterly*, 39(1), 217-234.

Eisenhardt, K.M. (1989). Building theories from case study research." *Academy of Management Review*, 14(4), 532-550.

Fu, W., Wang, Q., and Zhao, X. (2017). The influence of platform service innovation on value co-creation activities and the network effect. *Journal of Service Management*, 28(2), 348-388.

Gawer, A. (2009). *Platforms, Markets, and Innovation*. Cheltenham, UK: Edward Elgar.

Gawer, A. (2014). Bridging differing perspectives on technological platforms: toward an integrative framework. *Research Policy*, 43, 1239-1249.

Ghazawneh, A. and Henfridsson, O. (2013). Balancing platform control and external contribution in third-party development: the boundary resources model. *Information Systems Journal*, 23(1), 173-192.

Glaser, B.G. (1988). *Doing Grounded Theory: Issues and Discussions*. Mill Valley, CA: Sociology Press.

Glaser, B.G. and Strauss, A.L. (1971). *The Discovery of Grounded Theory Strategies for Qualitative Research*. Chicago: Aldine-Atherton.

Karhu, K. and Gustafsson, R. (2015). Open platform boundary resources as arenas for strategic exploitation. *Proceedings of the 2015 Annual Meeting of the Academy of Management*, Vancouver, British Columbia, Canada, August 7-11, 2015.

Katz, M. and Shapiro, C. (1986). Technology adoption in the presence of network externalities. *Journal of Political Economics*, 94(4), 822-841.

Kazan, E., Tan, C., Lim, E.T.K. (2016). Towards a framework of digital platform competition: a comparative study of monopolistic and federated mobile payment platforms. *Journal of Theoretical and Applied Electronic Commerce Research*, 11(3), 50-64.

Kim, H.J., Kim, I., and Lee, H. (2016). Third-party mobile app developers' continued participation in platform-centric ecosystems: an empirical investigation of two different mechanisms. *International Journal of Information Management*, 36, 44-59.

Koch, S. and Guceri-Ucar, G. (2017). Motivations of application developers: innovation, business model choice, release policy, and success. *Journal of Organizational Computing and Electronic Commerce*, 27(3), 218-238.

Kohler, T. (2018). How to scale crowdsourcing platforms. *California Management Review*, 60(2), 98-121.

Mantovani, A. and Ruiz-Aliseda, F. (2017). Equilibrium innovation ecosystems: the dark side of collaborating with complementors. *Management Science*, 62(2), 534-549.

McIntyre, D.P. and Srinivasan, A. (2017). Networks, platforms, and strategy: emerging views and next steps. *Strategic Management Journal*, 38, 141-160.

Mohagheghzadeh, A. and Svahn, F. (2016). Transforming organizational resources into platform boundary resources. *Proceedings of the 24th European Conference on Information Systems (ECIS)*, Istanbul, Turkey, June 12-15, 2016.

Oh, J., Koh, B., and Raghunathan, S. (2015). Value appropriation between the platform provider and app developers in mobile platform mediated networks. *Journal of Information Technology*, 30, 245-259.

Parker, G., van Alstyne, M., and Choudary, S.P. (2016). *Platform Revolution: How Networked Markets Are Transforming the Economy and How To Make Them Work for You*. New York: W.W. Norton & Company.

Parker, G., van Alstyne, M., and Jiang, X. (2017). Platform ecosystems: how developers invert the firm. *MIS Quarterly*, 41(1), 255-266.

Puschmann, C. and Ausserhofer, J. (2017). Social data APIs: origins, types, issues. In M.T. Schäfer and Karin van Es (Eds.), *The Datafied Society: Studying Culture Through Data* (pp. 147-154), Amsterdam University Press, Amsterdam, Netherlands.

Rafiq, A., Ågerfalk, P.J., and Sjöström, J. (2013). Boundary resources dependency in third-party development from the developer's perspective. *Proceedings of DESRIST 2013*, Helsinki, Finland, June 11-12, 2013.

Rivard, S. and Huff, S.L. (1985). An empirical study of users as application developers. *Information & Management*, 8(2), 89-102.

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

Saarikko, T. (2016). Platform provider by accident: a case study of digital platform coring. *Business Information Systems Engineering*, 58(3), 177-191.

Schreieck, M., Wiesche, M., and Krcmar, H. (2016). Design and governance of platform ecosystems – key concepts and issues for future research. *Proceedings of the 24th European Conference on Information Systems (ECIS)*, Istanbul, Turkey, June 12-15, 2016.

Song, P., Xue, L., Rai, A., and Zhang, C. (2018). The ecosystem of software platform: a study of asymmetric cross-side network effects and platform governance. *MIS Quarterly*, 42(1), 121-142.

Song, P., Xue, L., Zhang, C., and Rai, A. (2017). APIs in software platform: implications for innovation and imitation. *Proceedings of the 38th International Conference on Information Systems (ICIS)*, Seoul, South Korea, December 10-13, 2017.

Star, S.L. and Griesemer, J.R. (1989). Institutional ecology, 'translations', and boundary objects: amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19(3), 387-420.

Statista (2018). "Most famous social network sites worldwide as of April 2018, ranked by number of active users." Accessed on 16 July 2018 from https://www.statista.com/272014/global-social-networks-ranked-by-number-of-users.

Thomas, L.D.W., Autio, E., and Gann, D.M. (2014). Architectural leverage: putting platforms in context. *The Academy of Management Perspectives*, 28(2), 198-219.

Tiwana, A. (2014). *Platform Ecosystems: Aligning Architecture, Governance, and Strategy*. Waltham, MA: Morgan Kaufmann.

Tiwana, A., Konsynski, B., and Bush, A.A. (2010). Platform evolution: coevolution of platform architecture, governance, and environmental dynamics. *Information Systems Research*, 21(4), 675-687.

Venkatraman, N. and Lee, C. (2004). Preferential linkage and network evolution: a conceptual model and empirical test in the U.S. video game sector. *Academy of Management Journal*, 47(6), 876-892.

Wareham, J., Fox, P.B., and Cano Giner, J.L. (2014). Technology ecosystem governance. *Organization Science*, 25(4), 1195-1215.

Wulf, J. and Blohm, I. (2017). Service innovation through application programming interfaces – towards a typology of service designs. *Proceedings of the 38th International Conference on Information Systems (ICIS)*, Seoul, South Korea, December 10-13, 2017.

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

Zimmerman, S., Müller, M., and Heinrich, B. (2016). Exposing and selling the use of web services – an option to be considered in make-or-buy decision making. *Decision Support Systems*, 89, 28-40.

APPENDICES

Figure 1: Summary of the Relationship Between Digital Platform Providers and Third-Party Application Developers (3PADs), as Shaped by Platform Boundary Resources (PBRs)

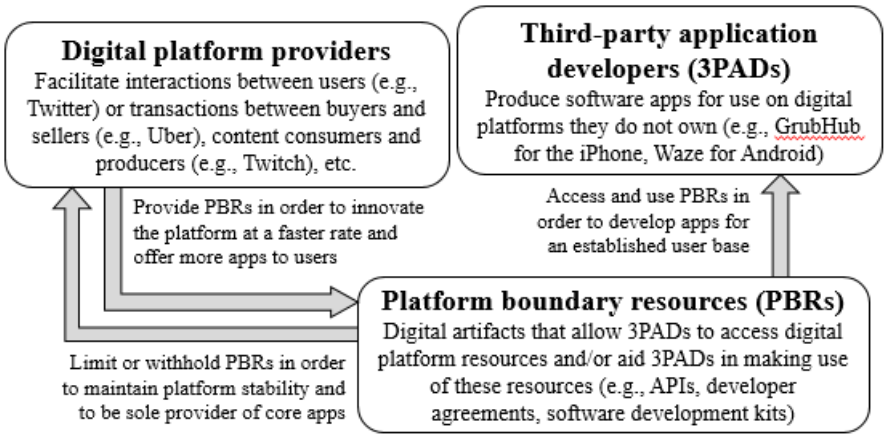

Table 1: References for Analyzed Articles

Code	Reference
Article numbers 1 to 33 were retrieved from ABI/Inform's Global Database	

1	M. Gibbs, "Analyzing Twitter with Excel, Part 3," *Network World*, 20 April 2009.
2	M. Gibbs, "Analyzing Twitter with Excel, Part 4," *Network World*, 27 April 2009.
3	N.D. Kho, "Ten Things You Need To Know About Twitter," *Information Today*, June 2009.
4	M. Rodier, "Algo Traders Hear Twitter's Call," *Wall Street & Technology*, August/September 2009.
5	L. Goldie, "Twitter Apps: Tuning Into the Chatter," *New Media Age*, 13 August 2009.
6	M. Gibbs, "The Twitter Sentimeter," *Network World*, 17-24 August 2009.
7	R. Karpinski, "Twitter Tools for Teams," *B to B*, 18 January 2010.
8	D. Gelles, "Twitter Builds On Its Character," *Financial Times*, 16 April 2010.
9	D. Talbot, "Can Twitter Make Money?," *MIT Technology Review*, March/April 2010.
10	R.L. Hotz, "Want To Monitor an Earthquake, Track Political Activity, or Predict the Ups and Downs of the Stock Market? Researchers Have Found a Bonanza of Real-Time Data in the Torrential Flow of Twitter Feeds," *Wall Street Journal*, October 2011.
11	Anonymous, "Twitter Leaves Business Market to Others," *Wall Street Journal*, 26 July 2012
12	L. O'Reilly, "Twitter Cracks Down on Third-Party App Developers to Keep Data for Itself," *Marketing Week*, 23 August 2012.
13	T. Claburn, "Twitter Rule Change Riles Developers," *Information Week*, 3 September 2012.
14	K. North, "From Medlars to Twitter," *Information Week*, 23 April 2012.
15	E. Dwoskin, "Data Mining Thanks to Twitter," *Wall Street Journal*, 7 October 2013.
16	A. Timms, "Mining Twitter for Market-Moving News," *Institutional Investor*, November 2013.
17	M.J. Lopresti, "Twitter Comes of Age in 2012," *EContent*, January/February 2013.
18	C. Delo, "Who Are Twitter and Facebook's Gatekeepers?," *Advertising Age*, 18 March 2013.
19	P. McNamara, "Cautionary Tale from a Twitter Share-Cropper," *Network World*, 25 March 2013.
20	E. Dwoskin, "Twitter Agrees to Buy Data Partner Gnip," *Wall Street Journal*, 15 April 2014.
21	Y. Koh, "Twitter to Offer New Tools for App Developers: Social Network Seeks to Build Good Will with Programmers at Conference," *Wall Street Journal*, 19 October 2014.
22	M. Bergen, "Twitter Can Help You Build an App for That," *Advertising Age*, 27 October 2014.
23	D. Clark and Y. Koh, "IBM and Twitter Forge Partnerships on Data Analytics," *Wall Street Journal*, 29 October 2014.
24	Anonymous, "Twitter Wooing Indian Developers with Fabric App Development Suite," *The Economic Times*, 5 May 2015.
25	Anonymous, "Twitter Needs to Unlock Its Data to Drive Growth," *Campaign*, 3 July 2015.

26	Y. Koh, "Twitter CEO Apologizes to App Developers," *Wall Street Journal*, 22 October 2015.
27	S. Agarwal, "Twitter Opens Free Mobile App Development Suite 'Fabric' to Partners," *The Economic Times*, 23 October 2015.
28	V. Kayser and A. Bierwisch, "Using Twitter for Foresight: An Opportunity?", *Proceedings of the XXVI ISPIM Conference*, June 2015.
29	M. Bergen, "Twitter to CMOs: We're for Data, Not Just Ad Impressions," *Advertising Age*, 23 March 2015.
30	C.S. Stewart, "Twitter Bars Intelligence Agencies from Using Analytics Service," *Wall Street Journal*, 8 May 2016.
31	S. Shankar, "Fork Media, Twitter Join Hands for Data," *The Economic Times*, 12 July 2016.
32	M. Kapko, "Is Twitter's Dead Developer Conference Another Nail in Its Coffin?," *CIO*, 14 October 2016.
33	D. Seetharaman and E. Minaya, "Twitter Aims to Cash in on Growth," *Wall Street Journal*, 10 February 2017.
Article numbers 34 to 91 were obtained via searches against Google's database of web documents	
34	B. Stone, "Introducing the Twitter API," *Twitter Developer Blog*, 20 September 2006. [https://blog.twitter.com/official/en_us/a/2006/introducing-the-twitter-api.html]
35	R. Miller, "Scaling Twitter for 600 Requests per Second," *DataCenter Knowledge*, 18 September 2007. [https://www.datacenterknowledge.com/archives/2007/09/18/scaling-twitter-for-600-requests-per-second]
36	K. Greene, "What Is He Doing," *MIT Technology Review*, 15 October 2007. [https://www.technologyreview.com/s/408856/what-is-he-doing/]
37	J. Gruber, "The Unsatisfying State of Twitter Web Clients for the iPhone," *Daring Fireball*, 17 April 2008. [https://daringfireball.net/2008/04/twitter_web_clients_for_the_iphone]
38	TweetDeck, "What Does 'Rate Limit Exceeded' Mean?," *Twitter Developer Blog*, 28 October 2008. [https://blog.twitter.com/official/en_us/a/2008/what-does-rate-limit-exceeded-mean-updated.html]
39	L. Gray, "Gnip CEO's Goal: Make Twitter's Data Flow Suck Less," *louisgray.com*, 18 July 2008. [https://blog.louisgray.com/2008/07/gnip-ceos-goal-make-twitter-suck-less.html]
40	M. Arrington, "Twitter Plays Nice: XMPP Firehose Data Feed to Gnip," *TechCrunch*, 18 July 2008. [https://techcrunch.com/2008/07/18/twitter-plays-nice-xmpp-firehose-data-feed-to-gnip/]
41	D. Frommer, "Twitter Confirms Paid Pro Accounts on the Way," *Business Insider*, 25 March 2009. [https://www.businessinsider.com/twitter-confirms-paid-pro-accounts-on-the-way-2009-3]
42	M. Volpe, "New Data on Top Twitter Applications and Usage," *Hubspot*, 27 February 2009. [https://blog.hubspot.com/blog/tabid/6307/bid/4584/new-data-on-top-twitter-applications-and-usage.aspx]

43	Anonymous, "Getting Historical Data from Twitter," *Stack Overflow*, 2 November 2009. [https://stackoverflow.com/questions/1662151/getting-historical-data-from-twitter]
44	B. Gaines, "Implementing Twitter Data Tracking in Omniture SiteCatalyst," *Adobe Blog*, 24 February 2009. [https://theblog.adobe.com/implementing-twitter-data-tracking-in-omniture-sitecatalyst/]
45	R. Gonda, "Best Tools to Analyze, Aggregate, and Visualize Twitter Data," *RobGonda.com*, 18 February 2009. [http://www.robgonda.com/2009/02/18/best-tools-to-analyze-aggregate-and-visualize-twitter-data/]
46	B. Nelson, "Using Google Spreadsheets to Extract Twitter Data," *brelson.com*, 20 November 2009. [http://www.brelson.com/2009/11/using-google-spreadsheets-to-extract-twitter-data/]
47	B.M. Carey, "Using the Twitter REST API," *IBM Developer Works*, 9 June 2009. [https://www.ibm.com/developerworks/library/x-twitterREST/index.html]
48	P. Statz, "Using the Twitter API," *Wired*, 15 February 2010. [https://www.wired.com/2010/02/get_started_with_the_twitter_api/]
49	Siddarth, "Diving Into the Twitter API," *Envato Tuts+*, 30 March 2010. [https://code.tutsplus.com/articles/diving-into-the-twitter-api--net-10607]
50	L. Rao, "Twitter Seeing 6 Billion API Calls Per Day, 70K Per Second," *TechCrunch*, 17 September 2010. [https://techcrunch.com/2010/09/17/twitter-seeing-6-billion-api-calls-per-day-70k-per-second/]
51	M.G. Siegler, "Twitter Slashes API Rate Limits in Half Across the Board to Deal with Capacity Issues," *TechCrunch*, 29 June 2010. [https://techcrunch.com/2010/06/29/twitter-api-limit/]
52	R. Singel, "Twitter Throttling Takes Toll on Tools," *Wired*, 6 July 2010. [https://www.wired.com/2010/07/twitter-throttling/]
53	M. Bryant, "Apigee Makes Life Easier for Twitter Developers," *The Next Web*, 14 April 2010. [https://thenextweb.com/apps/2010/04/14/apigee-twitter-api/]
54	J.C. Perez, "Twitter Clamps Down on Client Apps," *Macworld*, 14 March 2011. [https://www.macworld.com/article/1158531/twitter_client_apps.html]
55	R. Sarver, "Consistency and Ecosystem Opportunities," *Twitter Development Talk*, 11 March 2011. [https://groups.google.com/forum/#!topic/twitter-development-talk/yCzVnHqHIWo]
56	M. Brown, "Twitter Clamps Down on Third-Party Clients," *Wired*, 14 March 2011. [https://www.wired.com/2011/03/twitter-third-party-clients/]
57	J. O'Dell, "Twitter to Devs: Don't Make Twitter Clients... or Else," *Mashable*, 11 March 2011. [https://mashable.com/2011/03/11/twitter-api-clients/]
58	Anonymous, "Rate Limits on Twitter," *Texifter*, 18 August 2011. [https://texifter.com/2011/08/19/rate-limits-twitter/]
59	M. Melanson, "Twitter Kills the API Whitelist: What It Means for Developers and Innovation," *ReadWrite.com*, 11 February 2011. [https://readwrite.com/2011/02/11/twitter_kills_the_api_whitelist_what_it_means_for/]
60	J. Kincaid, "Loren Brichter, Creator of Official Twitter Apps for Mac and iPhone, Leaves Twitter," *TechCrunch*, 4 November 2011.

	[https://techcrunch.com/2011/11/04/loren-brichter-creator-of-official-twitter-apps-for-mac-and-iphone-leaves-twitter/]
61	O. Marx, "Dear Twitter: Here Is How To Fix Your API," *Observer*, 9 September 2011. [https://observer.com/2011/09/dear-twitter-here-is-how-to-fix-your-api/]
62	M. Sippey, "Changes Coming in Version 1.1 of the Twitter API," *Twitter Developer Blog*, 16 August 2012. [https://blog.twitter.com/developer/en_us/a/2012/changes-coming-to-twitter-api.html]
63	A. Hanna, "Collecting Real-Time Twitter Data with the Streaming API," *Bad Hessian*, 16 October 2012. [http://badhessian.org/2012/10/collecting-real-time-twitter-data-with-the-streaming-api/]
64	D. Bohn, "Twitter Dictates Third-Party App Form and Function in New API, Givers Six Months to Comply," *The Verge*, 16 August 2012. [https://www.theverge.com/2012/8/16/3248079/twitter-limits-app-developers-control]
65	M. Panzarino, "Developers, Bracing Themselves for Twitter API Restrictions, Call Today's Post Ominous," *The Next Web*, 29 June 2012. [https://thenextweb.com/twitter/2012/06/30/developers-bracing-themselves-for-twitter-api-retrictions-call-todays-post-ominous/]
66	E. Dwoskin, "Twitter's Data Business Proves Lucrative," *Wall Street Journal*, 7 October 2013. [https://www.wsj.com/articles/twitter8217s-data-business-proves-lucrative-1381104711]
67	Anonymous, "Twitter Firehose vs. Twitter API: What's the Difference and Why Should You Care," *BrightPlanet*, 15 June 2013. [https://brightplanet.com/2013/06/twitter-firehose-vs-twitter-api-whats-the-difference-and-why-should-you-care/]
68	T. Bucher, "Objects of Intense Feeling: The Case of the Twitter API," *Computational Culture*, 16 November 2013. [http://computationalculture.net/objects-of-intense-feeling-the-case-of-the-twitter-api/]
69	V. Luckerson, "Twitter is Selling Access to Your Tweets for Millions," *Time*, 8 October 2013. [http://business.time.com/2013/10/08/twitter-is-selling-access-to-your-tweets-for-millions/]
70	A. Green, "Twitter Consultant Tip: Get All the Twitter Data You Need for Free," *140Dev*, 3 December 2010. [http://140dev.com/twitter-api-programming-blog/twitter-consultant-tip-get-all-the-twitter-data-you-need-for-free/]
71	A. Green, "Twitter Ecosystem: Care and Feeding of Developers," *140Dev*, 20 October 2010. [http://140dev.com/twitter-api-programming-blog/twitter-ecosystem-care-and-feeding-of-developers/]
72	A.J. Dellinger, "#FirstWorldProblems: Twitter Third Party Clients Continue to Shut Down and its API is Just Getting More Restrictive," *Digital Trends*, 18 March 2013. [https://www.digitaltrends.com/mobile/firstworldproblems-twitter-api-and-third-party-problem/]
73	B. Kepes, "Twitter Buys Gnip – It's All About the Data," *Forbes*, 15 April 2014. [https://www.forbes.com/sites/benkepes/2014/04/15/twitter-buys-gnip-its-all-about-the-data/#2394bc2b7796]

74	F. Irving, "The Story of Getting Twitter Data and Its Missing Middle," *ScraperWiki*, 21 August 2014. [https://scraperwiki.com/2014/08/the-story-of-getting-twitter-data-and-its-missing-middle/]
75	J. Garside, "Twitter Puts Trillions of Tweets Up for Sale to Data Miners," *The Guardian*, 18 March 2015. [https://www.theguardian.com/technology/2015/mar/18/twitter-puts-trillions-tweets-for-sale-data-miners]
76	W. Ahmed, "Comparison of Twitter APIs and Tools for Analyzing Tweets Related to the Ebola Virus Disease," *Data Driven Journalism*, 11 February 2015. [http://datadrivenjournalism.net/news_and_analysis/comparison_of_twitter_apis_and_tools_for_analyzing_tweets_related_to_the_eb]
77	S. Yegulalp, "The Shut Off of Twitter's Firehose is a Hazard of the API Economy," *InfoWorld*, 13 April 2015. [https://www.infoworld.com/article/2908869/big-data/twitters-firehose-shut-off-is-the-newest-hazard-of-the-api-economy.html]
78	M. Anderson, "Twitter's Withdrawal of Reliable Share Count API is a Bold Monetising Move," *The Stack*, 5 October 2015. [https://thestack.com/cloud/2015/10/05/twitters-withdrawal-of-reliable-share-count-api-is-a-bold-monetising-move/]
79	V. van Der Mersch, "Twitter's 10 Year Struggle with Developer Relations," *Nordic APIs*, 23 March 2016. [https://nordicapis.com/twitter-10-year-struggle-with-developer-relations/]
80	K. Lane, "Taking a Fresh Look at the Twitter API," *API Evangelist*, 14 October 2016. [https://apievangelist.com/2016/10/14/taking-a-fresh-look-at-the-twitter-api/]
81	E. Anuff, "Almost Everyone is Doing the API Economy Wrong," *TechCrunch*, 21 March 2016. [https://techcrunch.com/2016/03/21/almost-everyone-is-doing-the-api-economy-wrong/]
82	A. Tornes, "Introducing Twitter Premium APIs," *Twitter Developer Blog*, 14 November 2017. [https://blog.twitter.com/developer/en_us/topics/tools/2017/introducing-twitter-premium-apis.html]
83	K. Wagner, "Twitter Is Selling a Cheaper Version of Its Enterprise Product to Try and Boost Sales," *recode*, 14 November 2017. [https://www.recode.net/2017/11/14/16647106/twitter-data-licensing-sales-revenue-gnip]
84	W. Ahmed, "Twitter as a Data Source: an Overview of Tools for Journalists," *Data Driven Journalism*, 23 May 2017. [http://datadrivenjournalism.net/news_and_analysis/twitter_as_a_data_source_an_overview_of_tools_for_journalists]
85	T. Claburn, "Twitter's Motto: If at First You Screwed Developers Over, Try, Try Again, Eh?," *The Register*, 6 April 2017. [https://www.theregister.co.uk/2017/04/06/twitter_rethinks_developer_relations_again/]
86	S. Perez, "Twitter Unveils a New API Platform, Roadmap, and Vision for Its Developer Community," *TechCrunch*, 6 April 2017. [https://techcrunch.com/2017/04/06/twitter-unveils-a-new-api-platform-roadmap-and-vision-for-its-developer-community/]

87	K. Yeung, "Twitter Simplifies API Platform, Publishes 2017 Developer Roadmap," *Venture Beat*, 6 April 2017. [https://venturebeat.com/2017/04/06/twitter-simplifies-api-platform-publishes-2017-developer-roadmap/]
88	A. Piper, "Building the Future of the Twitter API Platform," *Twitter Developer Blog*, 6 April 2017. [https://blog.twitter.com/developer/en_us/topics/tools/2017/building-the-future-of-the-twitter-api-platform.html]
89	A. Ostrow, "Twitter's Massive 2008: 752 Percent Growth," *Mashable*, 9 January 2009. [https://mashable.com/2009/01/09/twitter-growth-2008/#3MZ8I4KvKGqO]
90	M. Honan, "Twitter's Audacious Plan to Infiltrate All Your Apps," *Wired*, 22 October 2014. [https://www.wired.com/2014/10/twitter-fabric-sdk/]
91	J. Constine, "Google Acquires Fabric Developer Platform and Team from Twitter," *TechCrunch*, 18 January 2017. [https://techcrunch.com/2017/01/18/google-twitter-fabric/]

Table 2: The Tuning of Data Platform Boundary Resources by Twitter and Third-Party Application Developers (3PADs), 2006 to 2017

Instance No.	Tuning Agent	Data Tuning Action	Data Tuning Description	Data Tuning Goals	Was Goal Accomplished?
1	Twitter	Data provision	In September 2006 Twitter launched a set of free, public APIs that enabled 3PADs to retrieve data generated by Twitter's users.	1) Promote platform generativity (to add more functionality) 2) Allow Twitter to focus on delivering and scaling core services	Yes. As Twitter grew rapidly from 2006 to 2010, 3PADs developed many new, innovative apps, and Twitter successfully delivered and scaled its core services.
2	3PADs	Criticism of data limits	Through numerous tech articles spanning 2007 through 2009, 3PADs argued that Twitter's API v1.0 is too limiting in terms of the amount of data one can access and the ways in which the data can be accessed.	Persuade Twitter to: 1) Provide access its historical data; 2) Ease data rate limits; and 3) Offer more API methods.	No. Twitter responded by adding some methods and creating the whitelisting program in 2009 (see #3), but did not ease rate limits for non-whitelisted 3PADs or offer them access to historical data.
3	Twitter	Data provision	In June 2009 Twitter created a whitelisting program in which approved (whitelisted) 3PADs were able to access the real-time data stream and make more API calls per hour.	Further promote platform generativity and address 3PADs' calls for easing rate limits related to public API v1.0	Yes. While Twitter was criticized for not providing criteria for whitelisting approval, and for not whitelisting most 3PADs, whitelisting enabled some 3PADs to develop more data-intensive Twitter apps.
4	Twitter	Data outsourcing	In July 2009 Twitter outsourced the management of access to its historical data and real-time data stream to Gnip. As a result, whitelisted 3PADs had to work with Gnip to access these data, while some 3PADs began to pay for it.	1) Reduce the demands on its human and computing infrastructure 2) Continue to focus internal efforts on delivery and scaling of core services 3) Monetize the data	Yes, all three goals were accomplished.
5	Twitter	Data limiting	In September 2009 Twitter reduced the number of public API calls that a third-party app can make per hour (from 350 to 175).	Reduce the demands on its human and computing infrastructure	Yes. Reducing the overuse of its infrastructure helped Twitter commit more resources elsewhere, including core services and internal app development.
6	Twitter	Data limiting	In March 2011 Twitter posted an open letter to 3PADs stating that it would henceforth provide the core user experience through its own apps. By	Discourage the development of core user experience apps by 3PADs, thus hampering competition	Yes. Twitter throttled the amount of data that a violating app could retrieve, and acquired the most dominant core user experience apps, thus causing

			doing so, Twitter hoped to generate more ad revenue and, in turn, appeal to public investors.		many 3PADs to shift focus to the development of niche and complementary apps.
7	3PADs	Criticism of app gatekeeping and data limits	Through numerous tech articles spanning 2011 through 2012, 3PADs criticized Twitter for discouraging the third-party development of core user experience apps, arguing that it is unfair to 3PADs who invested in such apps.	1) Persuade Twitter to change its position 2) Persuade Twitter to not undermine 3PADs in the future 3) Persuade other 3PADs to stop developing for Twitter	No. Twitter carried out its plan to provide the core user experience, and continued to protect its interests vis-a-vis 3PADs. Moreover, many 3PADs continued to develop for Twitter.
8	Twitter	Data limiting	In April 2011 Twitter eliminated its whitelist. As a result, all 3PADs wanting to exceed the public API rate limits or access historical data had to purchase data through a certified data reseller.	1) Save labor costs by not having to review whitelisting requests 2) Monetize data	Yes. Twitter was no longer burdened with the administration of the whitelisting program. The elimination of this program also helped Twitter further monetize its data.
9	Twitter	Data provision	In August 2012 Twitter enabled public access to its real-time data stream, though public users could only receive a sample of real-time tweets. Twitter called this API its Streaming API.	Begin to introduce tiered data access to 3PADs, with each API segmented into a free tier and one or more paid tiers	Yes, inasmuch as Twitter offered tiered APIs to 3PADs through the period examined for this study (i.e., through 2017). Reports suggest that tiered access has been successful.
10	Twitter	Data limiting	In August 2012 Twitter launched API v1.1. With API v1.1, Twitter further limited most API calls (per app) to 60 per hour, down from 175 per hour.	Leverage tiered data access. If the basic (free) offering is diminished, then 3PADs are more likely to purchase the Enterprise API.	Yes, inasmuch as Twitter offered tiered APIs to 3PADs through the period examined for this study (i.e., through 2017). Reports suggest that tiered access has been successful.
11	Twitter	Data limiting	As part of API v1.1, Twitter reiterated that 3PADs should not develop core user experience apps. To this end, Twitter proposed five types of acceptable apps (e.g., enterprise apps, analytic apps). Violating apps were limited to 100,000 connected users.	Discourage the development of core user experience apps by 3PADs, thus hampering competition (same as #6)	Yes. Twitter's cap on the degree to which a violating third-party app could scale led many 3PADs to develop the apps sanctioned by Twitter. In turn, Twitter was able to fill market niches while continuing to provide its own apps for the core user experience.
12	3PADs	Criticism of app gatekeeping	Through numerous tech articles spanning 2012 through 2013, 3PADs expressed their frustration at taking large losses on app development investments. 3PADs also argued that Twitter's app gatekeeping is a short-sighted strategy and will harm Twitter's ability to innovate.	Persuade other 3PADs to stop developing for Twitter	No. Many 3PADs shifted their focus to Twitter's sanctioned apps. At this point, most 3PADs seemed to acknowledge that public pleading would not lead Twitter to reverse its position.

13	Twitter	Data monetization	In late 2013 Twitter expanded its list of certified data resellers (which already included Gnip and DataSift) to include NTT Data and Dataminr. Licensing data to NTT Data gave Twitter an inroads to the Japanese market.	Monetize data while outsourcing data access management	Yes. In 2013 Twitter earned nearly US$50 million by licensing its data to certified data resellers; by 2015, Twitter earned nearly US$200 million from data licensing. In addition, costs related to data access management were substantialy reduced.
14	Twitter	Data monetization	In early 2014 Twitter acquired Gnip, which had been administering Twitter's Enterprise API since August 2012. By doing so, Twitter once again was responsible for managing access to its historical and streaming data. The move also allowed Twitter to internalize Gnip's data mining and analytics capabilities.	Generate more advertising revenue by selling customized ads derived from data mining	No. By mid-2015 it had become clear that Twitter's revenues from advertising were not as large as it had hoped. As a result, Twitter focused more on selling and administering premium APIs.
15	Twitter	Data monetization	In 2017 Twitter began to offer a set of "Premium APIs" that provided access to more data than the free APIs but much less data than the Enterprise APIs. These Premium APIs were priced accordingly. And by terminating partnerships with certified resellers, Twitter took control of data access management.	Monetize data through a program of tiered data access (see also #9 and #10)	Yes. Although hard data are not available, tech articles suggest that Twitter's strategic move to control the sale and management of data access has been lucrative. At present, Twitter still offers Premium APIs and Enterprise APIs.