Organization Science
Vol. 23, No. 5, September–October 2012, pp. 1409–1427
ISSN 1047-7039 | EISSN 1526-5455 | 12 | 2305 | 1409
http://dx.doi.org/10.1287/orsc.1110.0678

Let a Thousand Flowers Bloom? An Early Look at Large Numbers of Software App Developers and Patterns of Innovation

Kevin J. Boudreau
London Business School, Regent's Park, London NW1 4SA, United Kingdom, kboudreau@london.edu

In this paper, I study the effect of adding large numbers of producers of application software programs ("apps") to leading handheld computer platforms, from 1999 to 2004. To isolate causal effects, I exploit changes in the software labor market. Consistent with past theory, I find a tight link between the number of producers on platform and the number of software varieties that were generated. The patterns indicate the link is closely related to the diversity and distinct specializations of producers. Also highlighting the role of heterogeneity and nonrandom entry and sorting, later cohorts generated less compelling software than earlier cohorts. Adding producers to a platform also shaped investment incentives in ways that were consistent with a tension between network effects and competitive crowding, alternately increasing or decreasing innovation incentives depending on whether apps were differentiated or close substitutes. The crowding of similar apps dominated in this case; the average effect of adding producers on innovation incentives was negative. Overall, adding large numbers of producers led innovation to become more dependent on population-level diversity, variation, and experimentation—while drawing less on the heroic efforts of any one individual innovator.

Key words: competition and innovation; multisided platforms; distributed and open innovation; network effects; software and digital innovation
History: Published online in *Articles in Advance* September 28, 2011.

1. Introduction

Multisided platforms (hereafter, simply "platforms") support interactions across multiple sets of actors and can facilitate technical development (Parker and Van Alstyne 2005, Boudreau 2010, Rochet and Tirole 2003, Weyl 2010). Decisions to adopt and use a platform on one side can benefit the other, potentially giving rise to cross-platform network effects. The literature now points to a long list of middlemen, bottlenecks, and intermediaries that follow this characterization.[1] A key theme emphasized in research on platforms is the importance of bringing large numbers of participants "on board" in order to foment network effects.[2] This paper reconsiders this argument for a particular class of multisided platform: computer platforms on which large numbers of independent software producers[3] develop application software. For this class of platform, the presumption has been that a large pool of independent application software producers will generate a wide variety of application software. Farrell and Weiser (2003) note the importance of bringing large numbers of producers onto a platform of this kind with the metaphor of letting a thousand flowers bloom.[4]

The practice of bringing large numbers of application software producers on board has become a mainstay of computing and networking industries. This has been particularly true since the early days of personal computing in the 1970s. The availability of a large library of business software, games, and productivity tools is widely agreed to have fuelled adoption of the IBM PC in the early 1980s (Langlois and Robertson 1992). Since that time, and largely with the IBM PC example in mind, nearly all computer platforms of note have included a "third-party developer" strategy.[5] This typically involves a combination of standard-form licensing contracts, development tools, documentation, and support in marketing and distribution. The approach would appear to only be growing in prominence. For example, leading enterprise software provider SAP underwent a program to transform its software products into platforms in the last decade by building facilities and launching programs to promote independent developers (Iansiti and Lakhani 2009). Even as the notion of computer platforms becomes increasingly broad, including such items as Web-based services (e.g., Google and Facebook), browsers (e.g., Firefox), middleware (e.g., Java), navigation systems (e.g., Garmin), and other sorts of platforms, the model of encouraging large numbers of developers to produce complementary applications has only become more widely prevalent. The model has received renewed acclaim with the Apple iPhone. First released in early 2007, the Apple's App Store was launched roughly a year and a half later with about 500 applications. By 2010, this number had already past a staggering

300,000 applications (Lardinois 2010), with the number of developers being roughly a quarter of this number (GigaOM 2010). Competing smartphone platforms, Google Android and Research in Motion (RIM) Blackberry, followed, with tens of thousands of applications and aggressive attempts to encourage still more. Apart from these large iconic firms, even small entrepreneurial firms now often attempt to develop "third-party developer programs" when launching their systems. For example, Livescribe, maker of a digital-pen platform, is attempting to grow a network of external developers to allow its users to access to a library of games, dictionaries, and productivity tools to enhance the value of its platform. Encouraging independent developers to come on board a computer platform has now simply become "business as usual."

Notwithstanding the many examples of platforms with large numbers of independent application software producers, the history of computing provides a number of cautionary tales regarding this liberal approach to granting access and encouraging outsiders to come on board and build on a platform. For example, after leading the early videogame industry, the Atari platform, and, indeed, the broader home videogame industry, lost its momentum following the "crash" of 1983 and 1984, in which the market was saturated with hundreds of low-quality games (Coughlan 2004). The modern videogame industry likewise raises questions regarding the importance of large numbers of developers. In this case, the industry is increasingly consolidated around large publishers such as Electronic Arts and Ubisoft, and console platform owners establish tight bilateral agreements with game publishers. Even in the case of Apple's App Store, recent press coverage has begun to suggest potential trade-offs from large numbers of independent application producers. Burrows (2010) reports that "in the past few months, an increasing number of app developers have complained that they couldn't make money on their work." In an article in *Wired* magazine (Rowan and Cheshire 2010), a successful developer, Kostas Eleftheriou, is quoted as describing the early days of development for the iPhone as follows: "Back then there were about 1,000 apps, so it was much, much easier." In relation to more recent development, he says, "Keep it simple to begin with. Don't invest three months in a project—you might make something very good, but it might not get noticed. It's a lottery now."

The goal of this paper is to gain better understanding of the economic mechanisms shaping application software innovation, particularly as the number of producers grows. To help guide the analysis, I begin by highlighting the particular institutional details of application development and hypothesize implications for innovation. The core of the paper is an empirical analysis of software applications built for mobile computers and personal digital assistants (PDAs) such as Palm, Microsoft Windows CE, Symbian, and Linux platforms. I study data from the period 1999–2004, because at the time, the bulk of all transactions for application software were carried on a single Web-based store, Handango.com. This makes it possible to observe all leading platforms with a single data source. In these data, I observe each software title, by each producer, for each of the eight market-leading platforms, broken down by genre. Given the goals of the analysis, I study innovation patterns using a simple reduced-form panel framework so as to reveal key patterns. I exploit shifts in the labour market for software developers as instrumental variables to estimate causal relationships. (There are a number of other econometric modelling considerations in this context of a two-sided platform, which are discussed at greater length herein.) With these data, I can assess the effects of varying numbers of application developers on variety and investment incentives. The data also provide some indication of variation generated by successive cohorts of entrants.

The rest of this paper proceeds as follows. In §2, I define and describe the generation of application software. In §3, I review key technical and institutional properties of application development that should shape the economics of innovation in these contexts. Section 4 develops guiding empirical hypotheses, based on these properties. Section 5 presents the context and data to be analyzed. Section 6 presents results. Section 7 discusses findings and concludes.

2. Computer Platforms, Application Software, and Development

Software is the set of instructions or "programs" and data that tell a computer what to do. Early electronic computers, such as machine controllers, computational devices, and calculators, were designed to execute focused tasks. These relatively simple and specialized instructions were designed as an integral part of computer systems and were rendered in "low-level" language, providing little abstraction from the computer's digital instruction set. With computers evolving into a "stored program" architecture (von Neumann 1945), software became increasingly flexible and abstracted from the hardware on which it ran. This gradually gave rise to the now-conventional distinction between application software and operating system platform software.

Operating system software is the set of programs and components that manage and regulate a computer's resources and subsystems and mediate access to analogue input–output devices.[6] Application software can be programmed and reprogrammed "on top" of the operating system, in the sense of calling on the functionality of the operating system. This distinction and the "swappable" nature of applications, held flexibly on read-writable media with the ability to load different application programs, was a key condition for transforming computers

into general-purpose technologies capable of performing a wide array of tasks. This basic application–program distinction remains, even as communications and networking now lead to increased nesting and interlinking among platforms.

As a problem-solving endeavour, software design requires a range of skills and abilities, including creativity and invention, trial-and-error learning and experimentation, diligent verification and testing, and attention to precise syntax (Cusumano 2004, Hohmann 1997). In addition, the "architecting" of an operating system involves making numerous subjective trade-offs to reconcile a complex array of technical constraints to utilize limited computer system resources effectively (Tanenbaum and Woodhull 1997). To architect a system is thus to define key characteristics and elements of a system, what a system will do well, what it will do less well, and the design approach to achieving these ends. It is therefore a relatively open-ended problem.

The development of application software is rather different from the development of operating systems. It instead involves devising instructions that call on the *defined* functionalities of the operating system: expert logical manipulation of preexisting design rules and a logical "grammar." Furthermore, application development in modern times is increasingly based on relatively widely taught and standardized programming languages. Nonetheless, there is often at least some degree of specialization in development rules and tools, because the particular strengths and limitations of a given system can necessitate to particularities in how applications call on the operating system. A consequence is that software applications built for one platform are still often unable to run on other platforms (without using add-on interfaces, adding converters or "middleware," or otherwise significantly rebuilding the application). Application software can typically be developed with a personal computer, perhaps using a simulator for the actual type of computer platform used or even the actual hardware and computer itself (e.g., videogame console, smartphone).

Apart from the technical aspects, application software design also often requires sensitivity to aesthetics and the man–machine user interface, as well as a keen understanding of particular user requirements. This latter knowledge often coincides with specialized domain knowledge or assets relevant to the application.[7] Developing novel application software ideas will itself require some measure of entrepreneurial knowledge regarding which applications may be demanded and which applications are currently served.

3. Key Properties of the Application Software Innovation Process

All innovation, creation, and discovery may be to some degree an uncertain and even mysterious process (Scotchmer 2004). In this section, I review several important properties of application development that should at least shape the broad outlines of the innovation process in this context.[8] These properties will serve to inform later hypothesis development.

3.1. Network Effects

In the case of independent application software developers on a computer platform, adding producers is understood to expand the selection/variety of software available for a platform,[9] thereby attracting more users to adopt the platform. This, in turn, leads to still more application developers and variety becoming available (e.g., Chou and Shy 1990, Church and Gandal 1992, Hagiu 2009). In effect, adding more software application producers should encourage still more producers by effectively expanding the market into which application producers sell their software (i.e., the "installed base" of users of the platform)—an "indirect" or "cross-platform" network effect. The literature on platforms therefore stresses the importance of accumulating a "critical mass" of consumers and producers on a platform (Evans 2009, Evans and Schmalensee 2010, Suarez 2004). "Bandwagon" dynamics may then follow, with escalating network effects (Arthur 1994, David 1985, Rohlfs 2003, Schilling 1999).[10] Crucial to this understanding of network effects is the presumption that added productions will lead to a superior selection of applications. Subsequent discussion relates to properties of this innovation process that take place within this broader notion of network effects.

3.2. Virtually Infinite Product Space

Arguably, a most basic distinguishing aspect of application software innovation is the sheer expanse of development possibilities. Software innovation is necessarily representational and informational (Varian et al. 2004) and often possesses cultural, intellectual, or aesthetic elements (Potts et al. 2008). Stoneman (2010) describes such goods as "soft" innovations, a class apart from the output of more usual industrial innovation.[11] Although a software compiler might develop within a finite set of instructions, the possibilities of generation are virtually boundless.[12] In this sense, Zittrain (2006) describes innovation in digital outputs as possessing "generativity," a tendency to expansive novelty. Caves (2000) argues this expansive novelty as inherent to the large, multidimensional product space in which creative endeavors, such as software development, effectively exist. He refers to this as the "infinite variety property" of a network (p. 6). This possibility of a virtually infinite variety is clearly consistent with the network effect characterization, if only because application software variety has a large space into which it may expand.

3.3. Extensive Recombination and Reuse

The expanse of possibilities might be thought of as larger still by virtue of the importance of relatively small differences in design. For example, the computer game *Counter-Strike* was built simply by modifying another game, *Half-Life* (Jeppessen and Molin 2003). In this sense, Parker and Van Alstyne (2008) observe that innovation of this kind is notably "sequential" and "cumulative." Cohen and Lemley (2001, p. 6) place this feature at the heart of the "special nature of innovation within the software industry," similarly stating that software is "characterized by rapid sequential innovation, reuse, and recombination." That "digital artifacts can be flexibly programmed and re-programmed" (Yoo et al. 2010b, p. 5) allows for an unending digital canvas, providing nuanced continuity and a vast expanse for creation. Thus, apart from an infinite expanse with many "distant" designs, the nuanced many-dimensional product space affords ample scope for incremental but meaningful "tweaking," reuse, and recombination to generate palpably novel offerings. Growing compatibility among digital goods and services creates still more scope for recombination of a range of digital programs, subroutines, services, features, and content—as when, say, mapping functions are added to advertising on a social networking application (Yoo et al. 2010a). Summarizing this sentiment, Yoo et al. (2010b, p. 14) emphasize that digital platforms enable "continual reinterpretations, expansions and refinements of products and services" and that the "design characteristics of digital representations . . . foster unbounded innovations through incessant recombination and modification of different elements in digital service architectures." (Also refer to Zittrain 2006.)

3.4. Uncertainty and Skewed Outcomes

A near-infinite technical frontier, by its nature, implies that much of it remains uncharted and unknown. Thus innovation into new applications will often be shrouded by uncertainty. As a consequence, the innovation of application software typically leads to many failed "experiments" and only isolated successes. More generally, Zittrain (2006) observes in digital innovations a high degree of "equivocality" or serendipity as regards what or who will be successful.[13] Echoing this sentiment, Lee and Wu (2009, p. 67) argue that "extreme uncertainty plagues the creation of distinctly new content" as they describe innovation on the Internet more broadly. Consonant with this sentiment, Lessig (2008, p. 14) states that "[t]housands could experiment on this common platform for a better way; millions of dot.com dollars will flow down the tube; but then a handful of truly extraordinary innovations comes from these experiments." Skewed outcomes lead this innovation to be popularly characterized as a search for the "killer app"—one that will achieve mainstream popularity and success. This is akin to Cave's (2000) "nobody knows" property of innovation in creative industries. Moreover, even were there no uncertainty, one might still expect skewed outcomes if a large number of offerings simply reflects the presence of heterogeneous and dispersed niche interests (e.g., see Anderson 2006, Brynjolfsson et al. 2006).

3.5. Low-Cost, Facilitated Development

An important corollary of skewed outcomes, uncertainty, and large numbers of producers is that any one software innovation should have low expected economic payoffs. Essential here, then, is the approach of building applications on top of platforms. This is an approach to system design that is explicitly intended to reduce development costs, speed product creation, reduce coordination costs, and facilitate experimentation by enabling reuse of core design elements (Baldwin and Woodard 2009, Robertson and Ulrich 1998, Simpson et al. 2006). This platform approach therefore entails an initial nontrivial fixed investment but then provides a base for innovation inherently geared toward handling large numbers of variants, supporting varied experiments, and providing sufficient flexibility and economies of scope and scale (Bakos 1991, Yoo et al. 2010a). In the case of software platforms, the close comingling of applications, and the underlying platform within the same digital media, creates still more scope for economizing.[14] For example, application development on a platform is now routinely associated with the provision of an application programming interface, an abstracted vocabulary that enables application programmers to call up, with simple commands, rich sets of the underlying operating system's functionality, with the effect of simplifying development. Other assets that come to be intimately associated with the platform include documentation, debuggers, source code examples, and integrated development environments. The role of providing tools and facilities is so integral to software platforms that these enabling tools are often viewed as synonymous with the platform (e.g., see Taudes et al. 2000). Furthermore, the applications are themselves information goods (Varian et al. 2004), allowing platforms to easily serve as distribution channels or intermediaries to users (i.e., the buyers of application software). This then alleviates the need to develop complementary assets in distribution.[15]

3.6. Ease of Access and Entry

Reducing both resource and skill requirements and "partitioning" the development task (von Hippel 1990) between applications and the platform have the effect of deepening the prospective pool of would-be application software producers (see MacCormack et al. 2006). Application platforms are therefore "inclusive" in that they allow many sorts of individuals to join should they have the motivation and incentives to do so. For example, a very basic application for the Apple iPhone might require just a couple of weeks of part-time development

work (Prochnow 2009).[16] "Digital technology," Lessig (2002, p. 9) writes, "could enable an extraordinary range of ordinary people to become part of a creative process," and von Hippel (2005, p. 13) emphasizes that "even individual hobbyists have access to sophisticated design tools . . . . With relatively little training and practice, they enable users to design new products and services." Thus, digital platforms have been described as a means of potentially "democratizing" digital innovation (Yoo et al. 2010b). Manifestations of this ease of joining sometimes reach extreme proportions, such as when the do-it-yourself Google App Inventor was launched in 2010; intended to enable regular users to develop applications on the Android handheld phone platform, it was tested mainly on sixth graders and nursing students (Lohr 2010).

4. Let a Thousand Flowers Bloom?

This section develops simple hypotheses to help guide the empirical analysis and interpretation of results that follow. Drawing on the properties of application development (in §3), the particular thrust is to begin to better understand the particular conditions that could warrant large numbers of producers and the implications of these conditions. (To this end, a number of more nuanced issues are simply identified categorically rather than fully probed or modelled.)

Despite the emphasis on large numbers of independent application producers, the link between numbers of application producers and the selection of software that is generated has yet to be tested in a systematic fashion.[17] Furthermore, there is no reason a priori why adding producers should add to or improve the selection of products that become available (Sah and Stiglitz 1987). Nevertheless, key properties of the application innovation process (see §3) suggest that, at least under certain conditions, there may be a positive link between numbers of producers and variety, as is generally presumed (see §3.1). To begin, the virtually infinite expanse of possibilities (see §§3.2 and 3.3) should at least allow for ever-expanding variety. Complementary to this, low barriers to feasibly participating in development (see §3.5) create the possibility of a diverse[18] pool of producers. This heterogeneity might plausibly be sustained if there is no compelling tendency toward consolidation (i.e., limited scale and scope economies) or if there is a weak selection environment. Features of the application platform context (see §3) suggest that this is again quite plausible. For example, the use of common development and distribution tools and assets (see §3.5) might itself guard against consolidation by "levelling the playing field" across producers, such that no one firm can achieve an ever-widening advantage. Large numbers of heterogeneous producers appear to be a matter of fact in numerous contexts (Yoo et al. 2008), and the importance of diverse know-how in pools of innovators has been persuasively argued in numerous platform contexts (see Baldwin and Clark 2000; Chesbrough 2006; Yoo et al. 2008, 2010b; von Hippel 2005).[19] Therefore, although the link between producers and variety is not a matter of strict or general conditions, the properties of application platforms (see §3) should make a positive link more likely. The first hypothesis is therefore simply an explicit test of the usual presumption.

HYPOTHESIS 1. *Adding more application software producers to a platform, all else being equal, causes the number of software varieties to increase.*

If the particular conditions outlined above manifest themselves, there is a direct corollary for producer scope, the number of application varieties developed by individual producers. The central role of diversity, as discussed earlier, implies enduring productive differences and comparative advantages across producers (consistent with §3.6). Equivalently, one should observe diseconomies of scope in cases in which producers venture beyond their areas of expertise. Without any such diseconomies of scope, application development would tend to consolidate; large numbers of producers simply would not be sustainable. This leads to the following hypothesis.[20]

HYPOTHESIS 2. *The product scope of individual application software producers tends to remain narrow and specialized.*

Provided that earlier conditions hold, the presence of large numbers of heterogeneous producers should be crucial to the provision of a wide variety of applications—and, more generally, experimentation and variation. However, a remaining question is how the group of application producers will respond to added numbers, in terms of their own choices, conduct, and behaviour in stimulating innovation. Perhaps the simplest issue to focus on is the effect on effort and investment. Several scholars have begun to suggest that one might begin to understand innovation incentives by likening groups of application producers on a platform to *markets* of competing innovators, in which case the varying intensity of competition and rivalry should shape innovation incentives (Parker and Van Alstyne 2008, Rysman 2009).[21] Markovich and Moenius (2009) go furthest in explicating a model with both competition and indirect network effects.[22] To the extent that this is true, added competition has the potential to stimulate innovation; however, especially intensive competition risks crowding out innovation incentives (see Aghion et al. 2005, Aghion and Griffith 2008).

Although the market analogy clearly plays a useful role in theory development on platforms, it might not wholly capture a long list of qualitatively distinct features of the application innovation context. For example, the virtually infinite expanse of the product space

(see §§3.2 and 3.3) implies that producers need not necessarily compete head-to-head with near-identical or even closely positioned offerings. Uncertainty and the search for "hits" (see §3.4) may imply that competition is less about usurping competitors' positions and more about searching for (any) viable solutions on a largely unknown frontier of possibilities. Specialization and heterogeneity of productive knowledge (see §3.6) could have the added consequence of "slotting" producers into distinct niches. Producers on platforms have also been known in instances to become "socialized" in the sense that producers know or know of one another, interact, and develop a sense of common identity or "membership" (Boudreau and Lakhani 2009). Any one of these properties could, in principle, alter the qualitative nature of, and likely "soften," competition.

There are, however, other features of application software competition and innovation that might still open the door to especially intensive competition. Although copyright[23] (see Lemley 1995) and, increasingly, patents (see Cohen and Lemley 2001, Bessen and Hunt 2007) may protect software innovations from outright machine code copying or "de-compilation" (see Cifuentes and Gough 1995, Samuelson and Scotchmer 2002),[24] these protections do not necessarily or easily extend to the protection of visual aspects, functionality, and sequences in a program's use. Once a given concept has been proven in the marketplace, competing producers might then be able to simply vary or recombine this concept (see §3.3). Furthermore, they may engage in this copying with the support of the very same development tools, technologies, and distribution channels that were used by the concept originator (see §3.5). Therefore, despite what may be a virtually boundless frontier, there may be scope for especially intensive pockets of competition, particularly for already market-proven applications. Adding still more nuance to strategic interactions in these cases, apart from crowding in the typical sense of "business stealing" among producers, the sometimes extraordinary numbers of producers and products might also lead to congestion or a "glut" of producers. This glut impairs the fully informed decision making of platform adopters (see Simonsohn 2010, Tucker and Zhang 2010).[25] Each of these features could also alter the qualitative nature of competition in relation to more usual industrial innovation. However, these particular features of interactions suggest that competition could instead be especially intense, at least in certain applications.

Adding further nuance to the question of interactions among producers and implications for innovation incentives is the presence of network effects. The multisidedness of platforms (see §3.1) may lead to complementarities, in the sense that added producers can stimulate innovation incentives. Added producers and associated expanded variety of software (see Hypothesis 1) can attract platform users, thus creating a "market pull" for greater investments. A growing and diverse population of producers could also spark further recombination and synthesis of concepts from a larger catalogue of accumulated innovations (Parker and Van Alstyne 2008). Hence, a number of studies of platform contexts, particularly those related to digital innovation and information technology, have hinted at a "virtuous cycle" between network effects and mounting innovation investments (Bresnahan and Greenstein 1999, Gawer and Cusumano 2002, Grindley 1995, von Burg 2001, Yoo et al. 2010a). A string of theoretical papers further suggests that this virtuous circle and network effects could coexist with simultaneously intensifying competition as more producers are added (e.g., see Church and Gandal 1992, Economides 1996a, Ellison and Fudenberg 2003).[26] There is some preliminary evidence that at least supports the view of ambivalent strategic interactions. Venkatraman and Lee (2004) show that videogame publishers tend to join platforms with relatively few producers on board, controlling for several factors; this is consistent with avoiding more intense competition on a platform with many producers. Augereau et al. (2006) study the case of an Internet service provider using different modem technology platforms and find evidence of both complementary and competitive interactions. In another example outside of software—in electronic markets—Tucker and Zhang (2010) find evidence of both network effects and competitive crowding. Here, I simply extend these views of ambivalent strategic interactions to their possible implications for innovation incentives.

HYPOTHESIS 3. *Adding distinct and differentiated independent application producers to a platform, all else being equal, raises individual producers' investment incentives.*

HYPOTHESIS 4. *Adding independent application producers to already-served application areas, all else being equal, crowds out individual producers' investment incentives.*

The remainder of this paper is devoted to empirically investigating these hypotheses.

5. Empirical Context and Data

5.1. Context

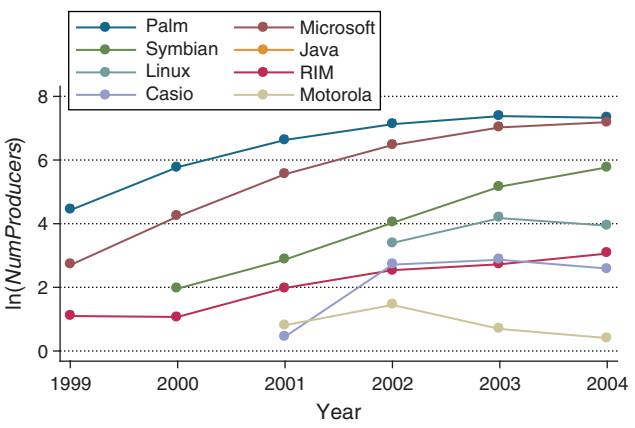
In the empirical analysis, I study point-of-sale data on the thousands of independent application producers for handheld computer platforms in the late 1990s and early 2000s, a period dominated by PDAs. The smartphone had not yet gained prominence, remaining, in most cases, a specialized application of PDA technology. Platform-specific retail channels (such as today's Apple iTunes Store or BlackBerry App World) had not

yet emerged, and the bulk of all transactions across leading platforms were carried on the third-party retailer Handango.com.[27] Handango kindly provided me with monthly data from its point-of-sale database for the period from November 1999 to December 2004, allowing me to observe new title releases, by individual producer and genre, for each leading platform for each month over the sample period. Counts of titles available for the leading platforms are presented in Figure 1.

The building of apps on handheld platforms quite closely followed the textbook conception of a multisided platform. Third-party app development on handhelds has been an important motivating example for theory development on platforms and network effects (see Evans et al. 2006, Rochet and Tirole 2003). McGahan et al. (1997) provide a detailed descriptive account of the positive interactions and network effects between increasing the variety of apps and growing the installed base of platform users. Nair et al. (2004) report econometric estimates of feedback between variety and consumer demand. However, the effect of varying number of producers on the generation of new offerings—the focus of this study—has yet to be systematically examined.

The study period (1999–2004) begins after the jarring changes of the mid-1990s that were precipitated by the entry of the iconic Palm Pilot; the study ends prior to the period in the mid-2000s when the PDA industry converged with digital music players and handheld phones. This was a period of relatively stable growth during which platforms vied for incremental gains in market share. At the beginning of the sample period, these devices already had several tens of millions of users.[28] In 2004, Palm, the early market leader, claimed that to that date more than 20,000 titles (cumulatively) had been developed for its platform (Nagel et al. 2005). Claims of other platform owners, such as Microsoft and Symbian, albeit more modest, still suggested thousands of releases.

Figure 1 Logarithm of the Number of Applications Available for Handheld Platforms (1999–2004)



5.2. Sample

The initial database from Handango included 56,760 distinct products. Dropping the nonsoftware products (hardware, accessories, and peripherals), titles for which a platform was not specified, server-side software, and multiuser contracts reduced the total number of observations by less than 5%. Eliminating all general media products (music, themes and backgrounds, digital content, graphics, e-books, guides, etc.) further reduced the sample to 28,960 titles sold by 5,973 independent producers across eight platforms (see Figure 1). Anecdotal information from the industry suggested that these producers included a varied range of small firms, divisions within larger firms producing mobile versions of their software, individual producers acting as entrepreneurs, hobbyists, students, and so on. The true complexion and range of producers cannot be directly observed in the data analyzed here. Releases, including new titles and new versions of existing titles, averaged 4.32 per producer over the course of the sample period. Because the Handango content was regularly refreshed, I was able to observe "actively sold" titles; each quarter, on average, roughly one-tenth more titles were added, and roughly one-twentieth of the least successful titles were removed.

The Handango database divides titles into distinct software categories. The sample breaks down as follows: games (19.4% of the titles), personal productivity (8.1%), translation and travel (4.2%), education and reference (3.8%), utilities and tools (5.5%), medical (4.4%), business and professional (4.5%), databases (1.2%), and unspecified (48.9%). (A visual inspection reveals that unspecified titles overlap with other categories and therefore cannot be treated simply as "other.") Eight platforms are represented in the data: Palm, Microsoft, Symbian, RIM, Linux, Motorola, Casio, and Java. Roughly 90% of the sample observations relate to market leaders Palm, Microsoft, and Symbian. Linux, Motorola, and Casio (roughly 1.5% of the sample) drop out when individual software categories are analyzed, because titles for these platforms were uncategorized. The categories remained unchanged throughout the sample period.

5.3. Data and Variables

Most closely related to Hypotheses 1 and 2, the measure of producer scope, *Scope*, is the count of the number of genres in which a producer has products. The total number of varieties in a system, *Varieties*, is then measured as the sum of varieties (or scope) offered by a given producer. The unit of analysis for *Scope* is an individual producer in a given month. The unit of analysis for *Varieties* is a platform in a given month.[29] Tables 1 and 2 define the variables and provide descriptive statistics.

To infer innovation incentives, as is most closely related to Hypotheses 3 and 4, I develop measures

Table 1 Variable Definitions

Variable	Unit of analysis	Definition
Scope	Producer–Month	Number of distinct genres (e.g., databases, medical, games) in which a producer offers application software
Varieties	Platform–Month	Sum of scope across all producers on a given platform
TimetoNewVersion	Title version	Number of months between subsequent releases developed by a producer within a given system and genre
NumProducers	Platform–Month	Total number of active producers in a system
NumProducersSameGenre	Genre–Month	Total number of active producers in a system that are selling titles within a given genre of software (e.g., databases, medical, games)
NumProducersOtherGenres	Genre–Month	Total number of active producers in a system that are selling titles outside of a given genre of software as the focal observation
Scale	Producer–Month	Total producer dollar revenues across all products in the time period
Experience	Producer–Month	Total cumulative past releases by the producer
Age	Producer–Month	Number of months that have passed since the producer entered the data
Version	Title–Month	Number (integer) of the current release of title within a given system and genre (i.e., first = 1, second = 2, etc.)
Time trend	Month	Count of months since beginning of the data
Employment	Month	Total software jobs in the United States (1,000s)
Wages	Month	Average wage of those employed in software development in the United States ($1,000s)

that distinguish new titles from releases of new versions of existing titles (versions 1.0, 2.0, 3.0, etc.; varieties such as "home" and "professional"). I construct a measure of the timing between subsequent versions of releases a given producers in a given genre, *TimetoNewVersion*, counted in months. (Double releases in the same time period are counted as a single incidence of a new release.) This variable necessarily disregards the first version, because I do not observe the amount of development time prior to the first version. Altogether, 2,277 uncensored durations correspond to categorized titles in the sample.

The explanatory variables of greatest interest relate to the numbers of producers on a platform, *NumProducers*. (Fewer than 1% of producers multihomed on multiple platforms.) *NumProducersSameGenre* counts the number of producers on a given platform working in a particular genre (e.g., databases, medical and health, games); *NumProducersOtherGenres* counts the number of producers working on the same platform in a genre other than the focal one. Titles for which a genre has not been specified are disregarded in calculating these variables.

The most important controls used in the analysis relate to the panel structure of the data, which includes a series of fixed effects and time trend controls discussed in the analysis. Also controlled for are producer- and product-level covariates (*Scale*, *Scope*, *Age, Experience*, and *Version*).

Additional data were collected to serve as instrumental variables (see §5.4). These included measures of the total employment of software programming jobs in the United States. The variable *Employment* measures the numbers of software programming jobs in thousands. The variable *Wages* measures the average wages in real dollars. These data were acquired from the U.S. Bureau of Labor Statistics.

Table 2 Variable Means, Standard Deviations, and Correlations

Variable	Mean	Std. dev.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
(1) *TimetoNewVersion*	6.1	5.7	1.00									
(2) *NumProducers*	859.6	624.7	0.06	1.00								
(3) *Scope*	1.7	0.8	−0.02	0.05	1.00							
(4) *Experience*	1.2	7.3	−0.11	0.04	0.15	1.00						
(5) *Age*	19.1	13.4	−0.02	0.39	0.29	0.20	1.00					
(6) *Sales*	189.7	1,640.8	0.12	−0.01	0.18	0.17	0.32	1.00				
(7) *Version*	1.6	3.2	−0.14	−0.06	0.31	0.42	0.25	0.14	1.00			
(8) *Time trend*	32.0	12.1	0.15	0.39	0.02	0.11	0.17	0.09	0.24	1.00		
(9) *Employment*	451.9	65.1	0.2	−0.60	−0.12	−0.05	−0.34	−0.12	0.04	−0.62	1.00	
(10) *Wages*	30.5	2.8	−0.1	0.24	0.06	0.03	0.22	0.18	−0.02	0.58	0.22	1.00

Notes. Means, standard deviations, and correlations are based on Producer–Platform–Category–Months observations, except *TimetoNewVersion* and *Sales*. *TimetoNewVersion* is based on Producer–Platform–Category–Version observations. *Sales* is based on individual title-level data. *Varieties* is not included because it is directly constructed from *Scope*.

5.4. Estimation Approach and Instrumental Variables

The objective of the regressions was to determine whether and how variation in a platform's number of producers led to changes in the measures of new software generated. Simple regressions of the dependent variables on measures of the number of producers should be biased for several reasons: omitted variables (e.g., changes in technical returns to innovation or unobserved market opportunities), possible reverse causality if the state of technology itself affects patterns of joining platforms, and a possible changing average composition or quality of producers in cases of low or high membership. It is consequently important to isolate variation in the number of producers that is not correlated with omitted variables, technological advance, or types of producers drawn to a platform.

The omitted-variable problem warrants an additional comment, related to the special case of a multisided platform. In effect, one would like the estimated coefficients on measures of the number of producers to incorporate any stimulating effect related to network effects. For this reason, I do not want to control for the number of users on the "other side" of the platform. However, any changes in the number of users not related to the network effect and not controlled for would create an upward bias, because entry and investment decisions would likely respond in the same direction. The estimation approach must therefore address this estimation challenge, along with the range of other potential sources of bias.

Also regarding omitted-variable bias, I should also note that the analysis relies on the context being one in which platform owners "opened" software development in a relatively stable fashion and without regularly intervening in the structure of markets for applications. To the extent that there were updates in development kits or other features that may have impinged on the functioning of competition and innovation in applications, I rely on time controls. Furthermore, the results I present here are robust to the inclusion of fixed effects for individual versions of the operating system (Palm 1.0, Palm 2.0, Palm 2.1, etc.). This is particularly important given that changes in tools or other structural features that may have shaped the structure of applications development most often coincided with the release of a new version of the operating system.

The estimation approach begins first with controlling for cross-sectional heterogeneity and compositional effects with a series of fixed effects: platform and genre fixed effects in the least stringent specifications and individual software producer- or product-level fixed effects in the most stringent. Transitory variation is then controlled for, at least in part, with time trends, even platform-specific time trends, and a series of control variables (see §5.3). To address the remaining sources of endogeneity bias, I isolate exogenous variation in the numbers of producers using an instrumental variables (IV) approach. The estimation procedure projects the measures of numbers of producers onto the Bureau of Labor Statistics' wage (*Wages*) and employment (*Employment*) data for the broader labour market for software developers in the United States.

The essential idea here is that changes in the labour market should have flushed developers into and out of handheld computing.[30] However, given its tiny size during the sample period (<1% of information technology), changes in the handheld sector should have had no meaningful impact on the broader labour market for software developers. Furthermore, given that producer firms were generally lone developers or small groups of developers that tended to remain constant over time, the formation or dissolution of producers was affected by the entry of individual developers (rather than resulting in the expansion and contraction of firms). Consistent with this reasoning, I find that an increase in 1,000 software jobs was associated with 0.65 fewer handheld software producers per platform and an increase of $1,000 in average wages with 9.1 fewer handheld software producers per platform, controlling for platform and time dummies. This implies that, all else being equal, developers tended to move into handheld computing software development when other information technology sectors were contracting. The estimates of these relationships are significant at $p = 5\%$. As *Employment* and *Wages* are simply time-series variables, I interact them with dummies for each platform when applied as instrumental variables. Therefore, strictly speaking, coefficients on the number of producers are estimated on the basis of differences in how platform numbers of producers responded to changes in the labour market for software developers.[31]

6. Results

6.1. Application Varieties and Producer Scope

I examine in this section how *NumProducers* related to two closely related variables, *Varieties* (per producer) and *Scope*. The results show that variety moved in a lockstep with the numbers of producers added to a platform and that producers' scope decisions were little affected by changes in the number of producers. This is because individual producer scope remained relatively narrow and invariant.

Results related to *Varieties* are reported in Table 3. The simple correlation of *Varieties* on *NumProducers* is 1.72 varieties per producer. Model 3-1 reestimates the relationship with platform fixed effects; Model 3-2 then also controls for time trends. These produce statistically identical estimates as the simple correlation. Model 3-2 then reestimates the relationship with the instrumental variables specification, which exploits *Employment*

Table 3 Results of Variety Fixed Effect (IV) Regressions

	Dependent variable = *Varieties*		
	Model 3-1	Model 3-2	Model 3-3
Explanatory variables	Platform FEs	Time trends	Instrumental variables
NumProducers	1.80*** (0.01)	1.73*** (0.01)	1.50*** (0.21)
Platform FEs	Yes	Yes	Yes
Platform time trends		Yes	Yes

Notes. Robust standard errors are in parentheses. Explanatory variables are lagged; instrumental variables are *Employment* and *Wages*, interacted with platform dummies. Number of observations = 393 platform-months. FEs, fixed effects.

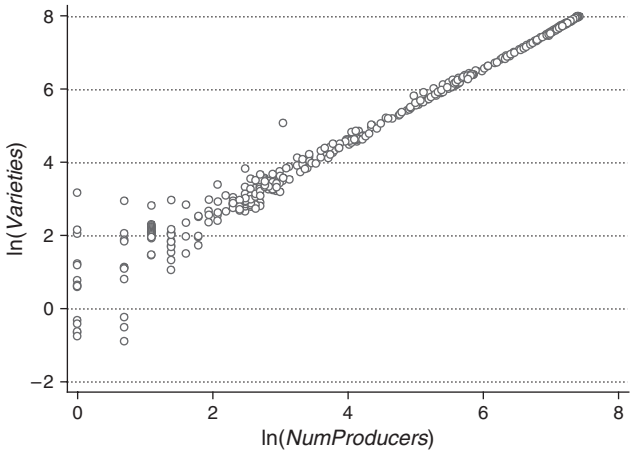
*Significant at the 10% level; **significant at the 5% level; ***significant at the 1% level.

and *Wages* interacted with platform dummies as instrumental variables, and it finds a slightly smaller magnitude of 1.50. The difference with the simple two-way correlation is again statistically insignificant. Nonetheless, 1.5 varieties per producer is the preferred estimate, given that the smaller magnitude may reflect the purging of even a slight upward bias if the factors that lead producers to join a platform also lead them to release more products. Allowing for nonlinear specifications (not reported), I find a statistically significant, but substantively minute, convexity in the relationship between *Varieties* and *NumProducers*. However, the effect is substantively tiny; the relationship is essentially linear, with variety moving in a lockstep with numbers of producers. Figure 2 plots the relationship, plainly illustrating that the relationship should be taken to be linear in any substantive sense. These results support Hypothesis 1.

Results related to *Scope* (in essence, the number of varieties offered by individual producers) are reported in Table 4.[32] Model 4-1 begins by regressing *Scope* on producer characteristics and platform fixed effects and time trends. The results reveal positive and significant relationships on *Scale*, *Experience*, and *Age*. Adding platform-specific time trends, as in Model 4-2, has little effect on the coefficient estimates, suggesting that the model is quite stable.[33] The main variable of interest, *NumProducers*, is introduced in Model 4-3. The estimated coefficient on this variable is significant but rather small (0.0002). Reestimating with the IV specification, as in Model 4-4, generates a similar estimate. Model 4-5 goes further to control for possible selection or compositional effects with producer fixed effects. (These were not included earlier, given that this analysis necessarily focuses on just 42% of producers whose scope varied over the sample.) The coefficients on *Scale*, *Scope*, and *Age* become smaller in magnitude but remain positive and significant. The estimated coefficient on *NumProducers* with these stringent panel controls is slightly larger (0.0006) but still quite small.

To provide a more substantive indication of the magnitude of the effect, Figure 3 presents the distribution of *Scope* in relation to different levels of *NumProducers*. The distribution is fitted to a Weibull distribution. This functional form is highly flexible and allows the data to be summarized in a way so as to detect any slight differences. The plot reveals that the positive effect of the number of producers on scope is not only small but only appears as the number of producers changes between 0 and 500 producers and between 501 and 1,000 producers. There is no discernible change in the distribution above this level. Therefore, individual producers' scope is relatively focused (roughly 1.5 varieties per producer, as mentioned earlier) and unchanging, consistent with Hypothesis 2.

Figure 2 *Varieties* (Corrected for Platform and Time Effects) vs. *NumProducers*



6.2. Innovation Incentives

I now turn to how the varying the number of producers affected individual producers' innovation efforts, most closely related to Hypotheses 3 and 4. I focus on the generation of new versions of existing titles as a way of discerning effects, particularly focusing on the time taken to release a new version, *TimetoNewVersion*. Clearly, any number of factors should affect the amount of time taken to complete a new version. However, the analysis here particularly isolates how variation in the number of producers has a causal impact on this time. A shorter amount of time—so long as all else is held equal—implies higher effort and input. I model the (nonnegative) durations between new versions using an IV Tobit framework.[34] For an instrumental variable, I interact *Employment* and *Wages* with dummies for software genres rather than platforms. Interacting with categories instead of platforms in producer-level data results in much more evenly sized groups and slightly more significant results. I find that the net effect of greater numbers of producers is to slow down the generation of new versions. Results are presented in Table 5.[35]

Table 4 Results of Producer Scope Fixed Effect (IV) Regressions

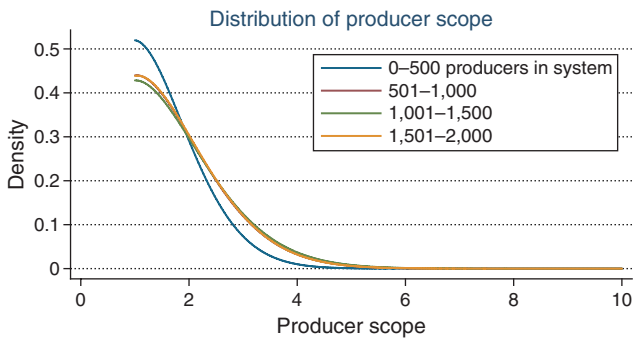
	Dependent variable = *Scope*				
	Model 4-1	Model 4-2	Model 4-3	Model 4-4	Model 4-5
Explanatory variables	Platform covariates and FEs	Time trends	Membership size	IV	IV and Producer FEs
NumProducers			0.0002***	0.0002***	0.0006***
			(0.0000)	(0.0000)	(0.0000)
Producer characteristics					
Scale	0.0135***	0.0130***	0.0130***	0.0130***	0.0068***
	(0.000)	(0.00)	(0.00)	(0.00)	(0.00)
Experience	0.0110***	0.0112***	0.0112***	0.0112***	0.0090***
	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)
Age	0.0122***	0.0155***	0.0157***	0.0157***	0.0104***
	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)
Producer FEs					Yes
Other controls					
Platform FEs	Yes	Yes	Yes	Yes	Yes
Platform time trends		Yes	Yes	Yes	Yes

Notes. Robust standard errors are in parentheses. Explanatory variables are lagged; instrumental variables are *Employment* and *Wages*, interacted with platform dummies. Number of observations = 121,503; number of cross-sectional units (suppliers) = 5,994. FEs, fixed effects.

*Significant at the 10% level; **significant at the 5% level; ***significant at the 1% level.

I first study relationships between *TimetoNewVersion* and the full set of controls. It is particularly important to assess the stability and robustness of the model in this case, given the importance of the earlier-mentioned "all else being equal" caveat. As reported in Model 5-1, coefficients on *Scale*, *Experience*, and *Scope* are positive and significant, and their magnitude and significance is relatively insensitive to model specification. This implies that larger, more successful producers tend to take longer or wait longer to release new versions.[36] The coefficient on *Version* is not significant. To control for possible nonlinear effects in the version number (when, say, earlier or later versions tend to take more or less time), I replace, in Model 5-1, the linear control for *Version* with a series of dummies for different versions, as reported in Model 5-2. The series of dummies are slightly more (jointly) significant (at $p = 15\%$) than the linear control. Therefore, I maintain this more stringent specification in the models that follow, although the results do not depend on choosing this specification.

Figure 3 The Distribution of Producer Scope for Different Numbers of Producers on a Platform



I now turn to the explanatory variables of greatest interest related to the numbers of producers. Here, I distinguish the number of producers within the same genre as a the focal producer, *NumProducersSameGenre*, from those in different genres, *NumProducersOtherGenres*. Adding these variables to the model without exploiting the instrumental variables results in positive coefficients, consistent with some unobserved factors that simultaneously encourage entry and investment in a given genre, platform, and period of time. I then reestimate the model with the instrumental variables specification, reported in Model 5-3. The −0.09 coefficient on *NumProducersOtherGenres* indicates that adding producers in one genre to a platform sped up development in other genres. This is consistent with an expansion of users on a platform and network effects occasioned by added members, resulting in a market pull for development (i.e., Hypothesis 3). The 1.24 coefficient on *NumProducersSameGenre* suggests that adding producers in the same genre to a platform slowed development in that genre. This is consistent with the presence of intensifying competition and the crowding-out of incentives among similar offerings (i.e., Hypothesis 4).[37]

To test the robustness of these results, I introduced, in Model 5-4, fixed effects at the level of individual titles. Including these fixed effects controls for any cross-sectional, nontransitory title, producer, platform, and genre heterogeneity. This is the most stringent possible cross-sectional control. These controls were not introduced in earlier regressions because they virtually reduce the sample by about half. (Producers with just

Table 5 Results of *TimetoNewVersion* Tobit (IV) Regressions

	Dependent variable = *TimetoNewVersion*				
	Model 5-1	Model 5-2	Model 5-3	Model 5-4	Model 5-5
Explanatory variables	Covariates	Version FEs	IV Tobit	Individual title FEs	Overall net effect
NumProducers					0.04*** (0.00)
NumProducersOtherGenres			−0.09*** (0.02)	−0.03 (0.02)	
NumProducersSameGenre			1.24*** (0.21)	0.68*** (0.23)	
Producer and title characteristics					
Scale	0.12*** (0.03)	0.12*** (0.03)	0.07 (0.08)	0.09 (0.18)	0.34*** (0.07)
Experience	0.01** (0.01)	0.02** (0.01)	0.01 (0.02)	0.01 (0.06)	0.10*** (0.02)
Scope	0.48*** (0.10)	0.47*** (0.10)	0.51 (0.27)	0.16 (0.64)	0.07 (0.28)
Version	−0.07 (0.20)				
Version FEs		Yes	Yes	Yes	Yes
Individual title FEs				Yes	Yes
Other controls					
Platform FEs and trends	Yes	Yes	Yes	Yes	Yes
Genre FEs and trends	Yes	Yes	Yes	Yes	Yes

Notes. Robust standard errors are in parentheses. Explanatory variables are averaged over the periods during which a new version was developed; instrumental variables are *Employment* and *Wages*, interacted with genre dummies. Number of observations = 2,263. FEs, fixed effects.
*Significant at the 10% level; **significant at the 5% level; ***significant at the 1% level.

one observation would effectively drop out of the analysis.) Despite the substantial change to Model 5-4 to now focus strictly on longitudinal variation not already controlled by the model, and an effective change in the sample on which coefficients are estimated, the coefficient estimates remain unchanged in sign. I regard these more stringent estimates to be preferred, given the particular importance of all else being equal, to meaningfully interpret coefficients. In Model 5-5 I estimate the overall net effect by replacing the earlier measures with the aggregate measure of *NumProducers*. The 0.04 coefficient is positive, indicating, on average (weighted across all dampening and stimulating effects of all producers), that the addition of producers caused an overall slowdown in the generation of new versions.

6.3. Variation Generated by Subsequent Cohorts of Entrants?

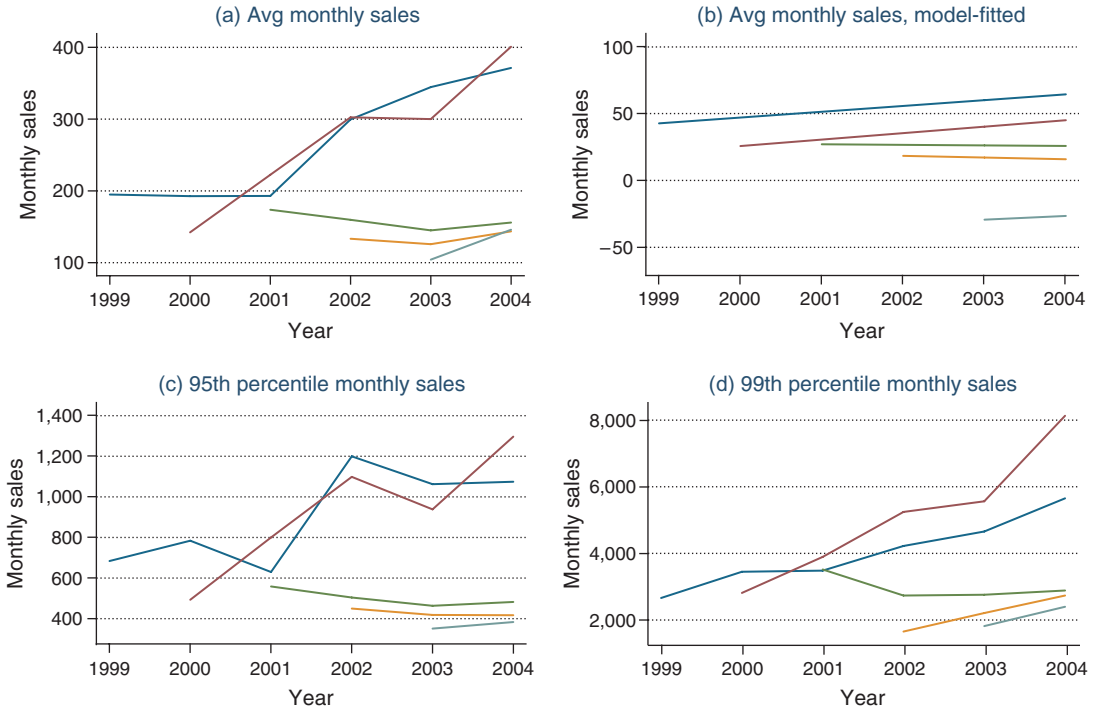
Given that marginal joiners had a negative impact on incumbent producers' innovation efforts, it is natural to then assess whether the later-joining (marginal) producers may have themselves contributed usefully to the basket of software generated. Some indication of how compelling the software of later entrants was can be gleaned by comparing the dollar sales of titles across successive cohorts. Figure 4(a) plots the average sales of titles released by cohorts of producers joining in different years. The patterns are consistent with titles released by later-entering cohorts being far less compelling. To ensure that these patterns are not created by general trending or inherent differences across platform or genre, Figure 4(b) plots linear fits of the average sales of titles, having corrected for platform- and genre-specific fixed effects and time effects. The basic conclusion of less compelling software with each cohort becomes even clearer in this controlled comparison.

It remains possible that the "extreme tail" of titles developed by late-entering producers might still have meaningfully added compelling titles. However, Figure 4(c) (showing sales of the 95th percentile) and Figure 4(d) (showing sales of the 99th percentile) reveal the same falling pattern. The 95th percentile of later-entering cohorts attains merely the average level of sales of earlier-entering cohorts; the 99th percentile does not exceed this much. Underlining this point, in the top three platforms, no single late entrant created an application that entered the list of the top 100 sellers.

7. Discussion and Conclusions

It has become common for owners of computer platforms of all sorts to try to attract large numbers of

Figure 4 Dollar Sales per Title in Successive Cohorts



Note. Cohorts are defined according to the year they begin/enter.

independent producers to their platforms in order to generate an attractive selection of application software. This might be understood as contradictory to usual notions of competition and innovation, where too much competition can quash innovation incentives (see Aghion et al. 2005, Aghion and Griffith 2008). However, in the case of application software, large numbers are understood to stimulate network effects (e.g., Church and Gandal 1992, Hagiu 2009). The key underlying presumption is that large numbers of producers will generate a wider and more attractive selection of application software. In this paper, I studied how a growing platform altered patterns of innovation in the case of software developed on an earlier generation of handheld computing devices. Broadly speaking, the foregoing analysis suggests that varying the number of application software producers does not simply increase or decrease the innovation level, but rather, it qualitatively transforms the nature and sources of innovation. Further, the results suggest a role that platform owners might play in regulating the market for application software (Boudreau and Hagiu 2009), so as to achieve desired innovation outcomes.

Within the analysis, I began by assessing the usual prediction that increasing numbers of developers should cause an increase in the variety of software titles. Consistent with the literature, I found that increasing the number of producers led to increased variety and did so in a linear lockstep fashion. A corollary finding is that the scope (i.e., variety offered) of individual producers remains narrow and unchanging with platform size. This more granular analysis showed that the link between variety and the number of producers was a direct consequence of individual producers maintaining a narrow and mostly unchanging product scope. Therefore individual scope decisions were clearly insensitive in this case to varying competition and strategic incentives. Rather, the steady narrow focus of producers points to the importance of specialization and comparative advantages of individual producers, as well as the collective diversity of the pool of application producers in generating such a wide array of apps. Therefore, not only did the variety of applications benefit from adding producers, but the lockstep relationship meant that to add variety, it was *required* to bring more producers on board. Several other results corroborated this importance of producer heterogeneity and comparative advantages. For example, to the extent that the scope of individual producers did vary, it evolved with enduring, slow-to-change characteristics of producers, such as scale and experience. Also indicative of heterogeneity, stark differences were found across products produced by producers of different cohorts.

Therefore, the evidence, as it relates to variety, is consistent with the bulk of past theorizing on software platforms while clarifying the role placed by specialization and comparative advantages. This pronounced role of heterogeneity in explaining the link between the number of producers and variety is also consistent with research related to various forms of open and distributed innovation (Baldwin and Clark 2000, Chesbrough 2006,

von Hippel 2005). In this case, however, it is important to note that a range of other mechanisms one might expect to be associated with heterogeneity—including spillovers, recombination, and accumulation of ideas—could not be directly observed in these data. Furthermore, I should emphasize that the evidence presented here did not allow direct observation of heterogeneity and the distribution of producers across the product space. The conclusions were simply inferred from a variety of patterns in the data. Clearly, direct observation and systematic mapping of key phenomena are areas for future research.

I then proceeded to examine whether there is evidence that varying the numbers of producers might also have affected innovation incentives (conditional on producers having joined a platform). I found that incremental increases in the number of application producers in this context led to a decrease in innovation incentives, on average, as measured by the rate at which new versions of existing titles were generated. (The particulars of this context and the econometric approach allowed me to interpret results in this manner.) This overall "crowding out" of incentives with intensive competition is consistent with what one might expect in more typical contexts of industrial competition and innovation (see Aghion et al. 2005, Aghion and Griffith 2008). Distinct from more typical industries, however, I also found that the net negative effect was the result of two separate and ambivalent effects. On the one hand, adding producers of different kinds of software tended to increase innovation incentives, consistent with network effects—a more attractive selection of applications creating more consumer demand and a market pull for more investments. On the other hand, adding producers tended to reduce incentives of producers of similar applications within the same software genre (e.g., spreadsheets and spreadsheets, games and games) On average, the second effect overwhelms the first in these data, leading to the observation of an overall crowding-out.

This second set of patterns is consistent with there being a virtuous circle of sorts acting between added producers and innovation, at least in part (Bresnahan and Greenstein 1999, Gawer and Cusumano 2002, Grindley 1995, von Burg 2001, Yoo et al. 2010a). The tension between network effects and crowding, as has been suggested by several papers (e.g., Augereau et al. 2006, Church and Gandal 1992, Economides 1996a, Ellison and Fudenberg 2003, Parker and Van Alstyne 2010, Venkatraman and Lee 2004). Links between these ambivalent strategic interactions and innovation investments have been theorized by Markovich and Moenius (2009).

Given the regular flux of producers in and out of these markets, the "churn" of suppliers at the margin might itself be a source of useful innovation and novelty (see Baldwin and Clark 2000)—one that might potentially offset negative incentive effects (see Boudreau et al. 2011). I examined the sales of individual products as a rough proxy of the quality and "compellingness" of products by subsequent cohorts of entrants. I found that even the very best extreme-right tails of products within later-entering cohorts were just also-ran products. I found no evidence of notable innovations coming from fringe entrants in this case.

Overall, I infer from these findings that platforms with large numbers of producers may come to depend on variation generated by population-level processes, rather than heroic efforts of any one innovator. In the context studied here, it was the diversity of the pool of individual specialized innovators that was paramount. However, at least in this case, the pattern of sorting onto these platforms led later-entering producers to produce increasingly less compelling software. Each of these results is not necessarily general, and many depend on the particulars of the context. Nonetheless, this case is an archetypal example of a hardware–software two-sided platform within the academic literature, and such platforms may be indicative of broader patterns in the economy.

Therefore, a number of economic mechanisms shape innovation on app platforms; not all will necessarily benefit from a growth in numbers of producers. The findings illuminate that there should be much scope to alter these innovation mechanisms—particularly given that platform owners may alter the conditions faced by app producers in terms of platform and tools design, information provision, subsidies and access pricing, and any number of other strategic "levers" (Gawer and Cusumano 2002, Iansiti and Levien 2004).

There are several particularities of this research and context that should be noted. In this there were already a relatively large number of producers on platforms (an average of 860 per platform in each period in the sample) in a period of relatively stable growth. Producers remained small and symmetrical. New development remained rather incremental. The small and symmetrical character of suppliers follows the usual textbook characterization of application producers around a platform; nonetheless, these are simply common, rather than necessarily general, conditions.

Also note that the estimated effects were "at the margin" in the sense of the effect of incremental variation in platforms where there were already many entrants. This is an important point; it is quite possible that inframarginal effects are quite different in magnitude and possibly in sign. For example, a newly launched platform with few application developers might only see application innovation increase as it initially adds application producers; only when it reaches some larger number of producers might the effects eventually turn negative. Also, the analysis focused on variation in numbers of producers on

a platform and thus controlled for interplatform competition and interplatform market structure. Finally, the empirical approach was devised while acknowledging the two-sided interactions with the consumer side of the platform without explicitly modelling these interactions.

Acknowledgments

The author thanks the team at Handango, and particularly Jason Wells, who kindly provided the data used in this study and who devoted considerable time and organizational resources to share insights. The author also thanks participants in seminars and presentations at Carnegie Mellon, Dartmouth, London Business School, Massachusetts Institute of Technology, HEC Paris, IESE, Wharton, the International Industrial Organization Conference (IIOC), IDEI Economics of Software and Internet Industries in Toulouse, the Sorbonne, the Center for Research in Economics and Strategy at the Olin School of the University of Washington in St. Louis, and the User and Open Innovation Workshop. This research was shaped in important ways by comments provided by Henry Chesbrough, Michael Cusumano, Andrei Hagiu, John Henderson, Joachim Henkel, Andrew King, Nicola Lacetera, Joan Enric Ricart, Fernando Suarez, Steven Klepper, Ron Adner, Carliss Baldwin, Rebecca Henderson, Karim Lakhani, Costas Markides, Ramana Nanda, Richard Nelson, Marc Rysman, David Teece, Catherine Tucker, James Utterback, Joel West, David Yoffie, Eric von Hippel, and Marshall Van Alstyne. The editors of this volume and review team deserve special mention for urging the author to more deeply ponder the institutional and technical characteristics of application software innovation—an invaluable suggestion. Last, this paper benefitted from the useful insights of a number of industry leaders and observers: Alessandro Araldi, Peter Bancroft, Larry Berkin, Michael Davies, Richard Doherty, Chris Dunphy, Jerry Epplin, Andrie Devries, Jessica Figueras, David Gannon, Eric Giguere, Randy Giusto, Avner Goren, Todd Gort, George Grey, Dan Hanttula, David Kipping, Evan Koeblentz, David Limp, David Nagel, Oto Hiroyuki, Michael Mace, Steve Mann, Darrin Massena, Robert Quinn, James Ready, Inder Singh, Steven Strassman, and Chris Tilley. Any errors are the author's own.

Endnotes

[1]Examples include credit and payment networks (linking merchants and consumers); Internet portals, magazines, newspapers, and other media (linking advertisers and readers); dating and matching markets (linking men and women, or other groups); electronic marketplaces and trading platforms (linking buyers and sellers), videogame consoles (linking game publishers and gamers), and many others. The Internet is itself a macrocosmic multisided platform that allows content and service producers to interact (Lee and Wu 2009).

[2]See Rysman (2009) for a recent review of the literature on multisided platforms. See Katz and Shapiro (1994) and Economides (1996b) for reviews on closely related research on network effects and systems competition.

[3]I refer to "producers" rather than "developers," because these groups may include both individual developers and larger firms and organizations.

[4]Farrell and Weiser (2003) more correctly quote Chairman Mao Zedong's reference to a hundred, rather than a thousand, flowers (p. 114; see also footnote 138 on p. 115), but I preserve the modern English (mis)translation as it more closely accords with the large numbers—thousands or tens of thousands—of application developers commonly found working on modern computing platforms.

[5]Even when third-party development is not sanctioned or supported, independent developers often begin developing on a platform in any case, as in the case of "jailbreak" application development on the iPhone immediately after its launch or the creation of new games on personal computer game engines (Boudreau and Jeppesen 2011).

[6]Examples include DOS, UNIX, Windows, Symbian, BREW, Mac OS, Blackberry OS, and Android.

[7]For example, database software is based on a long history of relational database theory. Medical prognosis software is based on established science and practice in the field. Modern videogame design increasingly requires deep knowledge of visual production and storytelling. Videogames may also build on rights to use film characters, sports figures, and the like.

[8]These points should be understood as highly salient within the literature, rather than necessarily comprehensive.

[9]Strictly speaking, formal models in this area do not explicitly model the division of labor in models of growing numbers of software titles and growing platform adoption. The link between the numbers of producers and numbers of software products remains casually understood.

[10]It is possible for extreme outcomes to emerge in certain instances. If network effects are strong, preferences for platforms are homogeneous, and switching costs are low, then a "winner-take-all" outcome can emerge, with the market "tipping" to just one or few platforms. Strong network effects can also create a "chicken-and-egg" problem, whereby a platform may have difficulty attracting members to get off the ground (Eisenmann and Hagiu 2008, Evans 2009, Evans and Schmalensee 2010). Would-be joiners might otherwise simply choose to "wait and see" before joining (Bakos 1991). Consequently, many businesses subject to network effects simply fail (Evans 2009, Evans and Schmalensee 2010, Noe and Parker 2005, Srinivasan et al. 2004).

[11]The rhetorical association to "software" is incidental and not exclusive. Stoneman (2010) uses "soft" to refer to innovations of an aesthetic and/or intellectual nature, in general.

[12]This is analogous in many ways to the European diatonic scale and a few common time signatures that have been the basis, throughout the ages, of a continually novel stream of popular music. One might also think of the finite set of rules governing the game of chess and the virtually infinite combination of games that might emerge.

[13]Bob Kahn, engineer and computer scientist, who, along with Vinton G. Cerf, invented the transmission control protocol (TCP) and the Internet protocol (IP), stated that the Defense Advanced Research Projects Agency "'would never have funded a computer network in order to facilitate e-mail' because other goals were more paramount, and person-to-person communication over telephones appeared sufficient" (Greenstein 2010, p. 5).

[14]Yoo et al. (2010a) refer to this property as the "self-referential" nature of digital innovation on a digital platform.

[15]Apple, for example, has recently launched its Mac Store to carry software directly to users. Microsoft has itself expressed an intent to launch an app store of a similar nature.

[16]A more ambitious project might consume many months and involve a range of skills, including programming (e.g., Objective-C, Cocoa), sketching, and user-interface design.

[17]The empirical literature on software platforms has thus far focused on the link between software variety and platform adoption, rather than underlying division of labor and supply structure. See, for example, Clements and Ohashi (2005), Zhu and Iansiti (2007), Nair et al. (2004), and Shankar and Bayus (2003).

[18]In this paper I use "diverse" in relation to producers and "variety" in relation to products. Here, I stress diversity in the sense of productive knowledge, i.e., capabilities and comparative advantages. In principle, diversity in motivations, beliefs, independent experimentation (of otherwise similar producers), or other dimensions sources of heterogeneity may play a role.

[19]To note, although there is work that begins to systematically study selection and sorting of producers of variety efficiency or "quality" onto platforms (e.g., Ambrus and Argenziano 2009, Gick 2010), a systematic and predictive view of the nonrandom processes governing the generation of diversity and the sorting of those with different sorts of skills onto platforms has yet to be developed.

[20]Katz and Shapiro (1994) mention an alternative view of the link between number of producers and variety. They suggest that large numbers of producers will intensify competition, guaranteeing the competitive level of variety. In this characterization, added producers would only lead to greater application variety to the point where the market for applications becomes competitive.

[21]Hagiu (2009) shows a trade-off between variety and quality in a case where quality is given and the challenge is to restrict entry to low-quality "types."

[22]In addition, the authors study cases in which complementary software developers are sufficiently large and influential to produce interactions between inter- and intraplatform competitive dynamics. In the context analyzed here, developers are small and numerous, and they do not have such market power.

[23]In the United States, legislation was enacted in the 1980s to extend copyright law to include machine-readable software. In the 1990s, treaties of the World Intellectual Property Organization set forth legislation in participating countries (including the United States) to explicitly criminalize reverse engineering of encryption devices. Software patents became routine only in the 1990s.

[24]Mann (2005) discusses patents and competition at greater length. For pioneering systematic empirical work, see Cockburn and MacGarvie (2006, 2009) and Hall and MacGarvie (2010).

[25]Bakos and Brynjolfsson (1993) point out still another mechanism for reduced incentives with added producers. Adding producers might reduce individual producers' bargaining power in relation to the platform owner, allowing the platform owner to capture a greater share of profits and potentially reducing marginal returns to investing.

[26]In an analogous argument in relation to matching platforms such as dating platforms, Halaburda and Piskorski (2010) note that notwithstanding the presence of network effects, restricting the number of potential matches (e.g., potential dates) may be beneficial by reducing competition on the same side of the platform.

[27]Anecdotal accounts from a wide range of industry participants suggested that roughly half of all trade was handled by Handango during the sample period. My own comparisons of NPD sales versus independent estimates of the market confirm this approximation.

[28]Anecdotal accounts and survey data gathered from Handango and market researchers suggested that by the early 2000s, a minority of handheld users (roughly a third) downloaded third-party applications (typically fewer than 10 per user). These are consistent with my own estimates of app sales and the size of the installed base.

[29]Replacing these measures of variety with simple raw counts of software titles leads to similar results.

[30]Desktop personal computing, being the source of most developers, is particularly relevant here.

[31]Estimates are unchanged when interacting genres of software and when U.S. computer science graduates are substituted for employment and wage time series as an instrumental variable.

[32]I continue to report results from a simple linear ordinary least squares framework. Results are unchanged in a count framework to reflect the fact the dependent variable is a nonnegative integer.

[33]Varying the particular combination of variables that is included has little impact on the results. I also investigated the use of a dummy for multihoming producers, finding no significant relationship.

[34]The Tobit framework allows for coefficients to be directly interpreted (as the number of months to a new version). Methods of implementing instrumental variables with Tobit are well established (Newey 1987).

[35]Because of the tiny size of individual producers, I disregard possible strategic timing of new versions (Ellison and Fudenberg 2000).

[36]There may be any number of explanations for this result. For example, large producers may tend to build complex products or be less inclined to replace their offerings. Alternatively, these markers of growth and differentiation might be construed to be consistent with inefficiencies, or diseconomies of scope and scale.

[37]Consistent with this interpretation, prices or volumes sold also decline with the addition of close substitute producers and increase with the addition of producers selling different types of software.

References

Aghion, P., R. Griffith. 2008. *Competition and Growth: Reconciling Theory and Evidence*. MIT Press, Cambridge, MA.

Aghion, P., N. Bloom, R. Blundell, R. Griffith, P. Howitt. 2005. Competition and innovation: An inverted-U relationship. *Quart. J. Econom.* **120**(2) 701–728.

Ambrus, A., R. Argenziano. 2009. Asymmetric networks in two-sided markets. *Amer. Econom. J.: Microeconomics* **1**(1) 17–52.

Anderson, C. 2006. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, New York.

Arthur, W. B. 1994. *Increasing Returns and Path Dependence in the Economy.* University of Michigan Press, Ann Arbor.

Augereau, A., S. Greenstein, M. Rysman. 2006. Coordination vs. differentiation in a standards war: 56K modems. *RAND J. Econom.* **37**(4) 887–909.

Bakos, J. Y. 1991. A strategic analysis of electronic marketplaces. *MIS Quart.* **15**(3) 295–310.

Bakos, J. Y., E. Brynjolfsson. 1993. From vendors to partners: The role of information technology and incomplete contracts in buyer-supplier relationships. *J. Organ. Comput. Electronic Commerce* **3**(3) 301–328.

Baldwin, C. Y., K. B. Clark. 2000. *Design Rules: The Power of Modularity*, Vol. 1. MIT Press, Cambridge, MA.

Baldwin, C. Y., C. J. Woodard. 2009. The architecture of platforms: A unified view. A. Gawer, ed. *Platforms, Markets and Innovation*. Edward Elgar, London, 19–44.

Bessen, J., R. M. Hunt. 2007. An empirical look at software patents. *J. Econom. Management Strategy* **16**(1) 157–189.

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

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

Boudreau, K. J., A. Hagiu. 2009. Platform rules: Multi-sided platforms as regulators. A. Gawer, ed. *Platforms, Markets and Innovation*. Edward Elgar, London, 163–191.

Boudreau, K. J., L. B. Jeppesen. 2011. Unpaid complementors and platform network effects? Evidence from on-line multi-player games. Working paper, London Business School, London. http://ssrn.com/abstract=1812084.

Boudreau, K. J., K. R. Lakhani. 2009. How to manager outside innovation. *MIT Sloan Management Rev.* **50**(4) 69–76.

Boudreau, K. J., N. Lacetera, K. R. Lakhani. 2011. Incentives and problem uncertainty in innovation contests: An empirical analysis. *Management Sci.* **57**(5) 843–863.

Bresnahan, T. F., S. Greenstein. 1999. Technological competition and the structure of the computer industry. *J. Indust. Econom.* **47**(1) 1–40.

Brynjolfsson, E., Y. Hu, M. D. Smith. 2006. From niches to riches: Anatomy of the long tail. *Sloan Management Rev.* **47**(4) 67–71.

Burrows, P. 2010. Apple vs. Google: How the battle between Silicon Valley's superstars will shape the future of mobile computing. *BusinessWeek* (Jaunary 14) http://www.businessweek.com/magazine/content/10_04/b4164028483414.htm.

Casadesus-Masanell, R., H. W. Halaburda. 2010. When does a platform create value by limiting choice? HBS Working Paper 11-030, Harvard Business School, Boston.

Caves, R. E. 2000. *Creative Industries: Contracts Between Art and Commerce*. Harvard University Press, Cambridge, MA.

Chesbrough, H. 2006. *Open Business Models: How to Thrive in the New Innovation*. Harvard Business School Press, Boston.

Chou, C., O. Shy. 1990. Network effects without network externalities. *Internat. J. Indust. Organ.* **8**(2) 259–270.

Church, J., N. Gandal. 1992. Network effects, software provision, and standardization. *J. Indust. Econom.* **40**(1) 85–103.

Cifuentes, C., K. J. Gough. 1995. Decompilation of binary programs. *Software: Practice Experience* **25**(7) 811–829.

Clements, M. T., H. Ohashi. 2005. Indirect network effects and the product cycle: Video games in the U.S., 1994–2002. *J. Indust. Econom.* **53**(4) 515–542.

Cockburn, I. M., M. MacGarvie. 2006. Entry, exit, and patenting in the software industry. NBER Working Paper 12563, National Bureau of Economic Research, Cambridge, MA.

Cockburn, I. M., M. MacGarvie. 2009. Patents, thickets and the financing of early-stage firms: Evidence from the software industry. *J. Econom. Management Strategy* **18**(3) 729–773.

Cohen, J. E., M. A. Lemley. 2001. Patent scope and innovation in the software industry. *Calif. Law Rev.* **89**(1) 1–58.

Coughlan, P. 2004. Golden age of home video games: From the reign of Atari to the rise of Nintendo. HBS Case 704487, Harvard Business School, Boston.

Cusumano, M. A. 2004. *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad*. Free Press, New York.

David, P. A. 1985. Clio and the economics of QWERTY. *Amer. Econom. Rev.* **75**(2) 332–337.

Economides, N. 1996a. Network externalities, complementarities, and invitations to enter. *Eur. J. Political Econom.* **12**(2) 211–233.

Economides, N. 1996b. The economics of networks. *Internat. J. Indust. Organ.* **14**(6) 673–699.

Eisenmann, T., A. Hagiu. 2008. Staging two-sided platforms. HBS Note 808-004, Harvard Business School, Boston.

Ellison, G., D. Fudenberg. 2000. The neo-Luddite's lament: Excessive upgrades in the software industry. *RAND J. Econom.* **31**(2) 253–272.

Ellison, G., D. Fudenberg. 2003. Knife-edge or plateau: When to market models tip? *Quart. J. Econom.* **118**(4) 1249–1278.

Evans, D. S. 2009. How catalysts ignite: The economics of platform-based start-ups. A. Gawer, ed. *Platforms, Markets and Innovation*. Edward Elgar, London, 99–130.

Evans, D. S., R. Schmalensee. 2010. "Failure to launch": Critical mass in platform businesses. *Rev. Network Econom.* **9**(4) Article 1.

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

Farrell, J., P. Weiser. 2003. Modularity, vertical integration, and open access policies: Towards a convergence of antitrust and regulation in the Internet age. *Harvard J. Law Tech.* **17**(1) 86–134.

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

GigaOM. 2010. Updated: The Apple app store economy. (January 12), http://gigaom.com/2010/01/12/the-apple-app-store-economy/.

Gick, W. 2010. Platforms or franchisors? Two contractual forms when consumers care about quality, Working paper, Harvard University, Cambridge, MA.

Greenstein, S. 2010. Nurturing the accumulation of innovations: Lessons from the Internet. NBER Working Paper 15905, National Bureau of Economic Research, Cambridge, MA.

Grindley, P. 1995. *Standards, Strategy, and Policy: Cases and Stories*. Oxford University Press, Oxford, UK.

Hagiu, A. 2009. Quantity vs. quality and exclusion by two-sided platforms. HBS Working Paper 09-094, Harvard Business School, Boston.

Halaburda, H., M. J. Piskorski. 2010. When should a platform give people fewer choices and charge more for them? *CPI Antitrust J.* **6**(2) 2010–2011.

Hall, B. H., M. MacGarvie. 2010. The private value of software patents. *Res. Policy* **39**(7) 994–1009.

Hohmann, L. 1997. *Journey of the Software Professional: A Sociology of Software Development*. Prentice-Hall, Upper Saddle River, NJ.

Iansiti, M., K. Lakhani. 2009. SAP AG: Orchestrating the ecosystem. HBS Case 609-069, Harvard Business School, Boston.

Iansiti, M., R. Levien. 2004. Strategy as ecology. *Harvard Bus. Rev.* **82**(3) 68–78, 126.

Jeppesen, L. B., M. Molin. 2003. Consumers as co-developers: Learning and innovation outside the firm. *Tech. Anal. Strategic Management* **15**(3) 363–383.

Katz, M. L., C. Shapiro. 1994. Systems competition and network effects. *J. Econom. Perspect.* **8**(2) 93–115.

Langlois, R. N., P. L. Robertson. 1992. Networks and innovation in a modular system: Lessons for the microcomputer and stereo component industries. *Res. Policy* **21**(4) 297–313.

Lardinois, F. 2010. Apple's app store soars past 300,000 apps. *ReadWriteWeb* (blog), November 4, http://www.readwriteweb.com/archives/apple_app_store_pushes_past_300000_apps_for_real_this_time.php.

Lee, R. S., T. Wu. 2009. Subsidizing creativity through network design: Zero-pricing and net neutrality. *J. Econom. Perspect.* **23**(3) 61–76.

Lemley, M. A. 1995. Convergence in the law of software copyright? *Berkeley Tech. Law J.* **10**(1) http://btlj.org/data/articles/vol10/Lemley.pdf.

Lessig, L. 2002. *The Future of Ideas: The Fate of the Commons in a Connected World*. Vintage Books, New York.

Lessig, L. 2008. *Remix: Making Art and Commerce Thrive in the Hybrid Economy.* Penguin Press, New York.

Lohr, S. 2010. Google's do-it-yourself app creation software. *New York Times* (July 12) http://www.nytimes.com/2010/07/12/technology/12google.html.

MacCormack, A., J. Rusnak, C. Y. Baldwin. 2006. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Sci.* **52**(7) 1015–1030.

Mann, R. J. 2005. Do patents facilitate financing in the software industry? *Texas Law Rev.* **83**(4) 961–1030.

Markovich, S., J. Moenius. 2009. Winning while losing: Competition dynamics in the presence of indirect network effects. *Internat. J. Indust. Organ.* **27**(3) 346–357.

McGahan, A. M., L. L. Vadasz, D. B. Yoffie. 1997. Creating value and setting standards: The lessons of consumer electronics for personal digital assistants. D. B. Yoffie, ed. *Competing in the Age of Digital Convergence*. Harvard Business School Press, Boston, 227–264.

Nagel, D., M. Mace, L. Berkin; Palmsource executive team. 2005. Personal communication. Palmsource, Sunnyvale, CA.

Nair, H., P. Chintagunta J. Dubé. 2004. Empirical analysis of indirect network effects in the market for personal digital assistants. *Quant. Marketing Econom.* **2**(1) 23–58.

Newey, W. K. 1987. Specification tests for distributional assumptions in the Tobit model. *J. Econometrics* **34**(1–2) 125–145.

Noe, T., G. Parker. 2005. Winner take all: Competition, strategy, and the structure of returns in the Internet economy. *J. Econom. Management Strategy* **14**(1) 141–164.

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

Parker, G., M. W. Van Alstyne. 2008. Innovation, openness, and platform control. Working paper, Tulane University, New Orleans. http://ssrn.com/abstract=1079712.

Potts, J., S. Cunningham, J. Hartley, P. Ormerod. 2008. Social network markets: A new definition of the creative industries. *J. Cultural Econom.* **32**(3) 167–185.

Prochnow, D. 2009. How to make an iPhone app: Part one. *Popular Sci.* (February 26) 1–6.

Robertson, D., K. Ulrich. 1998. Planning for product platforms. *Sloan Management Rev.* **39**(July) 19–32.

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

Rohlfs, J. H. 2003. *Bandwagon Effects in High-Technology Industries*. MIT Press, Cambridge, MA.

Rowan, D., T. Cheshire. 2009. The app explosion. *Wired* (December 22) http://www.wired.co.uk/magazine/archive/2010/02/features/the-app-explosion.

Rysman, M. 2009. The economics of two-sided markets. *J. Econom. Perspect.* **23**(3) 125–143.

Sah, R. K., J. E. Stiglitz. 1987. The invariance of market innovation to the number of firms. *RAND J. Econom.* **18**(1) 98–108.

Samuelson, P., S. Scotchmer. 2002. The law and economics of reverse engineering. *Yale Law J.* **111**(7) 1575–1663.

Schilling, M. 1999. Winning the standards race: Building installed base and the availability of complementary goods. *Eur. Management J.* **17**(3) 265–274.

Scotchmer, S. 2004. *Innovation and Incentives*. MIT Press, Cambridge, MA.

Shankar, V., B. L. Bayus. 2003. Network effects and competition: An empirical analysis of the home video game industry. *Strategic Management J.* **24**(4) 375–384.

Simonsohn, U. 2010. eBay's crowded evenings: Competition neglect in market entry decisions. *Management Sci.* **56**(7) 1060–1073.

Simpson, T. W., Z. Siddique, J. Jiao. 2006. *Product Platform and Product Family Design: Methods and Applications*. Springer, Berlin.

Srinivasan, R., G. L. Lilien, A. Rangaswamy. 2004. First in, first out? The effects of network externalities on pioneer survival. *J. Marketing* **68**(1) 41–58.

Stoneman, P. 2010. *Soft Innovation: Economics, Design, and the Creative Industries*. Oxford University Press, Oxford, UK.

Suarez, F. F. 2004. Battles for technological dominance: An integrative framework. *Res. Policy* **33**(2) 271–286.

Tanenbaum, A. S., A. S. Woodhull. 1997. *Operating Systems: Design and Implementation.* Prentice-Hall, Upper Saddle River, NJ.

Taudes, A., M. Feurstein, A. Mild. 2000. Options analysis of software platform decisions: A case study. *MIS Quart.* **24**(2) 227–243.

Tucker, C., J. Zhang. 2010. Growing two-sided networks by advertising the user base: A field experiment. *Marketing Sci.* **29**(5) 805–814.

Varian, H., J. Farrell, C. Shapiro. 2004. *The Economics of Information Technology: An Introduction*. Cambridge University Press, Cambridge, UK.

Venkatraman, N., C.-H. Lee. 2004. Preferential linkage and network evolution: A conceptual model and empirical test in the U.S. video game sector. *Acad. Management J.* **47**(6) 876–892.

von Burg, U. 2001. *The Triumph of Ethernet: Technological Communities and the Battle for the LAN Standard*. Stanford University Press, Stanford, CA.

von Hippel, E. 1990. Task partioning: An innovation process variable. *Res. Policy* **19**(5) 407–418.

von Hippel, E. 2005. *Democratizing Innovation*. MIT Press, Cambridge, MA.

von Neumann, J. 1945. First draft of a report on the EDVAC (June 30). Contract W-760-ORD-4926 between the United States Army Ordinance and the University of Pennsylvania, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia.

Weyl, E. G. 2010. A price theory of multi-sided platforms. *Amer. Econom. Rev.* **100**(4) 1642–1672.

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

Yoo, Y., K. Lyytinen, N. Srinivasan, N. Berente. 2008. Design principles for IT in doubly distributed design networks. *Internat. Conf. Inform. Systems (ICIS) 2008 Proc.*, Paper 178. http://aisel.aisnet.org/icis2008/178/.

Yoo, Y., K. J. Lyytinen, R. J. Boland Jr., N. Berente. 2010b. The next wave of digital innovation: Opportunities and challenges: A report on the research workshop "Digital Challenges in Innovation Research." Working paper, Temple University, Philadelphia. http://ssrn.com/abstract=1622170.

Zhu, F., M. Iansiti. 2007. Dynamics of platform competition: Exploring the role of installed base, platform quality and consumer expectations. HBS Working Paper 08-031, Harvard Business School, Boston.

Zittrain, J. 2006. The generative Internet. *Harvard Law Rev.* **119**(7) 1974–2040.

Kevin J. Boudreau is an assistant professor at the London Business School. He is an applied microeconomist and strategy scholar doing research at the intersection of innovation, organisation, and competition. He is particularly interested in platform competition and distributed innovation.