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

Kevin J. Boudreau

London Business School, Regent’s Park, London NW1 4SA, United Kingdom, kboudreau@london.edu

It is often presumed that bringing more members on board a multi-sided platform will stimulate value creation. Here I study the thousands of software producers building applications (“apps”) on leading handheld computer platforms (1999-2004). Consistent with past theory, I find a lock-step link between numbers of producers and varieties of software titles. The narrow, unchanging scope of producers and a series of other patterns are consistent with a pronounced role of specialization and heterogeneity of producers. I also find that while adding producers making different types of software stimulated investment incentives, consistent with network effects, adding producers making similar software crowded-out innovation incentives. The latter of these two effects dominates in this context. The patterns also indicate non-random generation and sorting of producers onto platforms, with later cohorts generating systematically less compelling software than earlier cohorts of entrants. Overall, added producers led innovation to become more dependent on population-level diversity, variation and experimentation—while drawing less on the heroic efforts and investments of any one individual innovator.

Key words: competition and innovation; multi-sided platforms; distributed and open innovation; network effects; software and digital innovation.

History:

1 Introduction

Multi-sided platforms (hereafter, simply “platforms”) are characterized by two or more sets of actors interacting through an intermediary. Decisions to adopt and use a platform on one side can benefit the other, potentially giving rise to cross-platform network effects (Parker and Van Alstyne 2005, Rochet and Tirole 2003, Weyl 2010). 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 article reconsiders this argument for a particular class of multi-sided platform: computer platforms on which large numbers of independent software producers3 develop applications software. For this class of platform, the presumption has been that a large pool of independent applications software producers will generate a wide variety of

---

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 multi-sided 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 multi-sided platforms. See Katz and Shapiro (1994) and Economides (1996) for reviews on closely-related research on network effects and systems competition.

3 I refer to “producers” rather than “developers”, as these groups may include both individual developers and larger firms and organizations.
applications 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.

The practice of bringing large numbers of applications 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. 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 things 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 App Store launched roughly a year and a half later with about 500 applications. By 2010 this number had already grown to a staggering 300,000 applications (Lardinois 2010), with the number of developers being roughly a quarter of this number (Giga.com 2010). Competing smartphone platforms, Google Android and 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.”

---

4 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” applications development on the iPhone immediately after its launch or the creation of new games on personal computer game engines (Boudreau and Jeppesen 2011).
Notwithstanding the many examples of platforms with large numbers of independent applications 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 video game industry, the Atari platform and indeed the broader home video game 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 video game industry similarly 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 the Apple App Store, recent press coverage begins to suggest potential tradeoffs from large numbers of independent applications producers. Business Week (2010) reports, “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 (2010), a successful developer, Kostas Eleftheriou, is quoted as describing early days of development for 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 the economic mechanisms shaping applications software innovation, particularly as the number of producers grows in number. To help guide the analysis, I being by highlighting the particular institutional details of applications development and hypothesize implications for innovation. The core of the paper is an empirical analysis of software applications built for mobile computers and PDAs, such as Palm, Microsoft Windows CE, Symbian, and Linux platforms. I study data from the period 1999-2004, as at the time, the bulk of all transactions for applications 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 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. In the paper I further discuss other considerations for measuring relevant relationships with the simple panel framework in the context of a two-sided platform and given the particular details of this empirical context. With these data, I am able to assess effects of varying numbers of applications
developers on variety and investment incentives. The data also provide some indication of variation generated by successive cohorts of entrants.

The paper proceeds as follows. In Section 2, I define and describe the generation of applications software. In Section 3, I review key technical and institutional properties of applications 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, Applications 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 applications 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. Applications software can be programmed and re-programmed “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 applications 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

5 Examples include DOS, UNIX, Windows, Symbian, BREW, Mac OS, Blackberry OS, and Android.
“architecting” of an operating systems involves making numerous subjective tradeoffs to reconcile a complex array of technical constraints to utilize limited computer system resources effectively (Tenenbaum 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 is will do less well, and the design approach to achieving these ends. It is therefore a relatively open-ended problem.

The development of applications 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 pre-existing design rules and a logical “grammar.” Further, applications 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, as the particular strengths and limitations of a given system can necessitate to particularities in how applications call upon the operating system. A consequence is that software applications built for one platform are still often not able to run on other platforms (without add-on interfaces, converters or “middleware”, or otherwise significantly rebuilding the application). Applications software can typically be developed with a personal computer, perhaps using a simulator for the actual type of computer platform used, or perhaps the actual hardware and computer, itself (e.g., videogame console, smartphone, etc.).

Apart from the technical aspects, applications software design also often requires sensitivity to aesthetics and man-machine user interface, and a keen understanding of particular user requirements. This latter knowledge often coincides with specialized domain knowledge or assets relevant to the application. 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 Applications Software Innovation Process
All innovation, creation and discovery may to some degree be an uncertain and even mysterious process (Scotchmer 2004). In this section, I review several important properties of applications

---

6 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 story-telling. Videogames may also build on rights to use film characters, sports figures and the like.
development that should at least shape the broad outlines of the innovation process in this context. These properties will serve to inform later hypothesis development.

(I) **Network Effects.** In the case of independent applications software developers on a computer platform, adding producers is understood to expand the selection/variety of software available for a platform, thereby attracting more users to adopt the platform. This, in turn, leads to still more applications developers and variety becoming available (e.g. Chou and Shy. 1990, Church and Gandal 1992, Hagiu 2009). In effect, adding more software applications producers should encourage still more producers, by effectively expanding the market into which applications 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 2010, Evans and Schmalensee 2009, Suarez 2004). “Bandwagon” dynamics may then follow, with escalating network effects (Arthur 1994, David 1985, Rohlfs 2003, Schilling 1999). Important to note here, the presumption that added producers leads to (the innovation of) a superior selection of applications software being innovated and becoming available is fundamental to the notion of network effects in this context. Subsequent discussion relates to properties of this innovation process that takes place within this broader notion of network effects.

(II) **Infinite Product Space.** Arguably a most basic distinguishing aspect of applications 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. While a software compiler might

---

7 These points should be understood as highly salient within the literature, rather than necessarily comprehensive.
8 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 numbers of producers and numbers of software products remains casually understood.
9 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 2009). 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 2009, Noe and Parker 2005, Srinivasen et al. 2004).
10 The rhetorical association to “software” is incidental and not exclusive. Stoneman uses “soft” to refer to innovations of an aesthetic and/or intellectual nature, generally.
develop within a finite set of instructions, the possibilities of generation are virtually boundless. In this sense, Zittrain (2005) describes innovation in digital outputs as possessing “generativity”, a tendency to expansive novelty. We might then interpret this case of applications software development as at least closely analogous to innovation in creative endeavours, more generally, which Caves (2000) describes as possessing an “infinite variety property”, occupying large, many-dimensional product spaces. This possibility of virtually infinite variety is clearly consistent with the network effect characterization, if only because applications software variety has a large space into which it may expand.

(III) Extensive Recombination. 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 as notably “sequential” and “cumulative”. Cohen and Lemley (2001) place this feature at the heart to the “special nature of innovation within the software industry,” similarly stating that software is “characterized by rapid sequential innovation, reuse and recombination.” That “digital artefacts can be flexibly programmed and re-programmed” (Yoo et al. 2010) 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 to a social networking application (Yoo et al. forthcoming). Summarizing this sentiment, Zittrain (2005) emphasizes that digital platforms “enable continual reinterpretations, expansions and refinements of products and services” and that the “design characteristics of digital representations… [and] foster unbounded innovations through incessant recombination and modification of different elements in digital service architecture.” In practice, recombinations may chiefly be limited by practical institutional limitations and the exertion of intellectual property rights. Where rapid recombinations and many ideas may be accumulated in an innovation, we may see “wakes” of innovation emerge (Boland et al. 2007).

---

11 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.
(IV) Uncertainty & Skewed Outcomes. A near infinite technical frontier, by its nature, implies that much of this frontier remains uncharted and unknown. Thus innovation into new applications will often be shrouded by uncertainty. As a consequence the innovation of applications software typically leads to many failed “experiments” and only isolated successes. Zittrain (2005) observes in digital innovations, more generally, a high degree of “equivocality” or serendipity as regards what or who will be successful.\textsuperscript{12} Echoing this sentiment, Lee and Wu (2009) 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) states “[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, we might still expect skewed outcomes if a large number of offerings simply reflects heterogeneous and dispersed niche interests (\textit{i.e.}, Anderson 2006, Brynjolfsson \textit{et al}. 2006), or if expanded numbers of alternatives simply implies that “smaller quantities of each product will be produced on average” (von Hippel 2005).

(V) 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 2005, Simpson \textit{et al}. 2006). This platform approach therefore entails an initial non-trivial fixed investment, but then provides a base for innovation inherently geared to handling large numbers of variants, supporting varied experiments, and providing sufficient flexibility and economies of scope and scale (Bakos 1991, Yoo 2010). In the case of software platforms, the close co-mingling of applications and the underlying platform within the same digital media, creates still more scope for economizing.\textsuperscript{13} For

\textsuperscript{12} 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 DARPA “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).

\textsuperscript{13} Yoo \textit{et al}. (forthcoming) refer to this property as the “self-referential” nature of digital innovation on a digital platform.
example, applications development on a platform is now routinely associated with the provision of an “application programming interface” (API), an abstracted vocabulary that enables applications 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” (IDE). 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., Taudes et al. 2000). Further, to this, the applications are themselves information goods (Varian et al. 2004), allowing platforms to also easily serve as distribution channels or intermediaries to users (i.e. the buyers of applications software). This then alleviates the need to develop complementary assets in distribution.14

(VI) Ease of Access and Entry. Reducing both resource and skill requirements and “partitioning” the development task (von Hippel 1990) between applications and platform has the effect of deepening the prospective pool of would-be applications software producers (cf. MacCormack et al. 2006). Applications platforms are therefore “inclusive” in this sense of allowing 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 weeks of part-time development work (Prochnow 2009).15 “Digital technology,” Lessig (2001) writes, “could enable an extraordinary range of ordinary people to become part of a creative process,” and von Hippel (2005) 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. 2010). Manifestations of this ease-of-joining sometimes reach extreme proportions, as when the do-it-yourself Google App Inventor, announced in 2010 and intended to enable regular users to develop applications on the Android handheld phone platform, was tested mainly on sixth graders and nursing students (Lohr 2010).

14 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.

15 A more ambitious project might consume many months and involve a range of skills including programming (objective-C, Cocoa), sketching, and user-interface design.
4 Let a Thousand Flowers Bloom?

This section develops simple hypotheses to help guide the empirical analysis that follows, and later interpretation of results. Drawing on the properties of applications development (Section 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 applications producers, the link between numbers of applications producers and the selection of software that is generated has yet to be tested in a systematic fashion.16 Further, 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 applications innovation process (Section 3) suggest that, at least under certain conditions, there may be a positive link between numbers of producers and variety, as is generally presumed (3.I). To begin, the virtually infinite expanse of possibilities (3.II and 3.III) should at least allow for ever-expanding variety. Complementary to this, low barriers to feasibly participating in development (3.V) create the possibility of a diverse17 pool of producers. This heterogeneity might plausibly be sustained if there is no compelling tendency towards consolidation (i.e., limited scale and scope economies), or if there is a weak selection environment. Features of the application platform context (Section 3) suggest this is again quite plausible. For example, the use of common development and distribution tools and assets (3.V) 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 (cf., Baldwin and Clark 2000, Chesbrough 2006, Yoo et al. 2008, 2010, von Hippel 2005).18 Therefore, while the link between producers and variety is not a matter of strict or general conditions, the properties of

---

16 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 (2004), Iansiti and Zhu (2007), Nair et al. (2004), and Shankar and Bayus (2002).

17 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.

18 To note, although we have 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), we have yet to develop a systematic and predictive view of the non-random processes governing the generation of diversity and the sorting of those with different sorts of skills onto platforms.
applications platforms (Section 3) should make a positive link more likely. Our first hypothesis is therefore simply an explicit test of the usual presumption:

**Hypothesis 1**: Adding more applications 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. Central role for diversity, as earlier, implies enduring productive differences and comparative advantages across producers (consistent with 3.VI). Equivalently, we should observe diseconomies of scope in cases in which producers venture beyond their areas of expertise. Without any such diseconomies of scope, applications development would tend to consolidate; large numbers of producers simply would not be sustainable. This leads to the following hypotheses.\(^\text{19}\)

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

Provided earlier conditions hold, the presence of large numbers of heterogeneous producers should be crucial to the provision of a wide variety of applications—and experimentation and variation, more generally. But, a remaining question is how the group of applications producers will respond to added numbers, in terms of their own choices, conduct and behaviour in stimulating innovation. Perhaps a simplest issue to focus on is the effect on effort and investment. Several scholars have begun to suggest that that we might begin to understand innovation incentives by likening groups of applications producers on a platform to *markets* of competing innovators, in which case varying intensity of competition and rivalry should shape innovation incentives (Parker and Van Alstyne 2008, Rysman 2009).\(^\text{20}\) Markovich and Moenius (2009) go furthest in explicating a model with both competition and indirect network effects.\(^\text{21}\) To the extent this is true, added competition has the

\(^{19}\) Katz and Shapiro (1994) mention an alternative view of the link between number of producers and variety. They suggest large numbers producers will intensify competition, guaranteeing the competitive level of variety. In this characterization, added producers would only lead to greater applications variety to the point where the market for applications becomes competitive.

\(^{20}\) Hagiu’s (2009) shows a tradeoff between variety and quality in a case where quality is given and the challenge is to restrict entry to low-quality “types”.

\(^{21}\) This very nice paper also goes further to show links between inter and intra-platform competition and innovation and dynamic path implications where software developers are large enough to exert significant monopoly power. In the context analyzed here, developers small and numerous and do not have such market power.
potential to stimulate innovation; however, especially intensive competition risks crowding-out innovation incentives (cf. Aghion et al. 2005, Aghion and Griffith 2008).

While the market analogy has clearly played a useful role in theory development on platforms, it might not wholly capture a long list qualitatively distinct features of the applications innovation context. For example, the virtually infinite expanse of the product space (3.II and 3.III) implies that producers need not necessarily compete head-to-head with near-identical or even closely positioned offerings. Uncertainty and the search for “hits” (3.IV) 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 (3.VI) 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, 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 applications software competition and innovation that might still open the door to especially intensive competition. While copyright22 (cf. Lemley 1995) and, increasingly, patents (cf. Cohen and Lemley 2001, Besen and Hunt 2007) may protect software innovations from outright machine code copying or “de-compilation” (cf. Cifuentes and Gough 1995, Samuelson and Scotchmer 2001),23 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 concepts has been proven in the marketplace, competing producers might then be able to simply vary or recombine these concepts (3.III). Further, they may engage in this copying with the support of the very same development tools, technologies and distribution channels that the were used by the concept originator (3.V). 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

---

22 In the United States, legislation was enacted in 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.

23 Mann (2005) discusses patents and competition at greater length. For pioneering systematic empirical work, see Cockburn and MacGarvie (2006, 2009), Hall and MacGarvie (2006).
of producers and products might also lead to congestion or a “glut” of producers that impairs fully-informed decision making of platform adopters (cf. Simonsohn 2006 and Tucker 2010), creates additional search or matching costs or added costs where, where applicable, network effects around individual applications fail to form (Casadesus-Masanell and Halaburda 2010). 24 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 multi-sidedness of platforms (3.1) may lead to complementarities, in the sense that added producers can stimulate innovation incentives. Added producers and associated expanded variety of software (Hypothesis I) 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, Gaver and Cusumano 2002, Grindley 1995, von Burgh 2001, Yoo et al. forthcoming). A string of theoretical papers goes further to suggest this virtuous circle and network effects could coexist with simultaneously intensifying competition, as more producers are added (e.g., Church and Gandal 1992, Economides 1996, Ellison and Fudenberg 2003, Parker and Van Alstyne 2010). 25 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, consistent with avoiding more intense competition on platform with many producers. Augereau et al. (2006), in the case of a Internet Service Provider using different modem technology platforms, rather than the case of software, finds evidence of both complementary and competitive interactions. Tucker (2010), also in an example outside of software—in electronic markets—finds evidence of both

24 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.

25 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, that restricting the number of potential matches (e.g., potential dates) may be beneficial by reducing competition on the same side of the platform.
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 applications producers to a platform, all else being equal, raises individual producers’ investment incentives.

**Hypothesis 4**: Adding independent applications producers to already-served applications 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 applications producers for handheld computer platforms in the late 1990s and early 2000s, a period dominated by personal digital assistants (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’s iTunes Store or Blackberry’s App World) had not yet emerged, and the bulk of all transactions across leading platforms were carried on third-party retailer Handango.com. Monthly data from its point-of-sale database for the period November 1999 to December 2004, kindly provided by Handango, enabled 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.

<Figure 1>

The building of apps on handheld platforms quite closely followed the textbook conception of a multi-sided platform. Third-party apps development on handhelds has been an important motivating example for theory development on platforms and network effects (cf. Evans et al. 2005, 2006, Rochet and Tirole 2003). McGahan et al. (1997) provide a detailed descriptive account of the

---

26 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.
positive interactions and network effects between increasing variety of apps and growing installed base of platform users, and Nair et al. (2004) report econometric estimates of feedback between variety and consumer demand. But 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) commences well after the jarring changes of the mid-1990s that were precipitate by the entry of the iconic Palm Pilot, and ended prior to industry convergence with music players and handheld phones in the mid-2000s. 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, there were already several tens of millions of users of these devices.\textsuperscript{27} Palm, the early market leader, claimed in 2004 that to that date more than 20,000 titles (cumulatively) had been developed for its platform. Claims of other platform owners, such as Microsoft and Symbian, albeit more modest, still suggested thousands of releases.

\subsection*{5.2 Sample}
The initial database from Handango included 56,760 distinct products. Dropping the non-software products (hardware, accessories, and peripherals), titles for which a platform was not specified, server-side software, and multi-user 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 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 removed.

The Handango database divides titles into distinct software categories. The sample breaks down as follows: games (19.4\% of titles), personal productivity (8.1\%), translation and travel (4.2\%),

\textsuperscript{27} 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 ten per user). These are consistent with my own estimates of apps sales and the size of the installed base.
education and reference (3.8%), utilities and tools (5.5%), medical (4.4%), business and professional (4.5%), databases (1.2%), and unspecified (41.1%). (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, Research in Motion (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, as titles for these platforms were uncategorized. The categories remained unchanged throughout the sample period.

5.3 Data and Variables

Most closely related to Hypotheses I and II, 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.28 Tables 1 and 2 define the variables and provide descriptive statistics.

To infer innovation incentives, as is most closely related to Hypotheses III and IV, I develop measures that distinguish new titles from releases of new versions of existing titles (i.e., versions 1.0, 2.0, 3.0, etc., or different varieties such as “home” and “professional”). I construct a measure of the timing between subsequent versions of releases a given producer 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, as 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 numbers of members on a platform during a given period of time, NumProducers (i.e., those with at least one active title), has the potential to double count those who offer products across multiple platforms, fewer than 1% of producers “multi-homed.” NumProducerSameGenre counts the number of producers on a given platform working in a particular genre (e.g., databases, medical and health, games, etc.), NumProducersOtherGenres the number of producers working on the same platform in other than

28 Replacing these measures of variety with simple raw counts of software titles leads to similar results.
the focal genre. 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 (cf. Section 5.4). These included measures of 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 average wages in real dollars. These data were acquired from the US Bureau of Labour Statistics.

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 measures of new software that was generated. Simple regressions of the dependent variables on measures of 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 possible changing average composition or quality of producers in cases of low or high membership. It is consequently important to isolate variation in 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 multi-sided platform. In effect, we would like our estimated coefficients on measures of number of producers to incorporate any stimulating effect related to network effects. For this reason, we do not want to control for the number of users on the “other side” of the platform. However, any changes in numbers of users not related to the network effect that are not controlled for would create an upward bias, as entry and investment decisions would likely respond in the same direction. The estimation approach must therefore address this estimation challenge, along the range of other potential sources of bias.
Also regarding omitted variable bias, it should also be noted 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 there were updates in development kits or other features that may have impinged on the functioning of competition and innovation in applications, here I rely on time controls. Further, each of the results presented here are robust to the inclusion of fixed effects for individual versions of the operating system (e.g., Palm 1.0, 2.0, 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 with first controlling for cross-sectional heterogeneity and compositional effects with a series of fixed effects: platform and genre fixed effects in the least and individual software producer or product-level fixed effects in the most stringent specifications. Transitory variation is then controlled for, at least in part, with time trends, even platform-specific time trends, and a series of control variables (Section 5.3). To address remaining sources of endogeneity bias, I isolate exogenous variation in numbers of producers using an instrumental variables (IV) approach. The estimation procedure essentially projects the measures of numbers of producers onto Bureau of Labour 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.\(^29\) But 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. Further, given that producer firms were generally lone developers or small groups of developers that tended not to change 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 .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

\(^{29}\) Desktop personal computing, being the source of most developers, is particularly relevant here.
tended to move into handheld computing software development when other IT 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 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.\(^{30}\)

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 lock step with the numbers of producers added to a platform, and that producers’ scope decisions were little affected by changes in 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) re-estimates 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 re-estimates the relationship with the instrumental variables specification, which exploits Employment and Wages interacted with platform dummies as instrumental variables, and 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 factors that lead them to join a platform also lead producers to release more products. Allowing for non-linear specifications (not reported) find a statistically significant, but substantively minute convexity in the relationship between Varieties and NumProducers. But the effect is substantively tiny; the relationship is essentially linear, with variety moving in lockstep with numbers of producers. Figure 2 plots the relationship plainly illustrate that the relationship should be taken to be linear in any substantive sense. These results support Hypothesis 1.

<Table 3>

\(^{30}\) Estimates are unchanged when interacting genres of software, and when US computer science graduates are substituted for employment and wage time series as an instrumental variable.
Results related to *Scope* (essentially, the number of varieties offered by individual producers) are reported in Table 4. 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* and *Experience* and *Age*. Adding platform-specific time trends, as in model (4-2), has little effect on the coefficient estimates, suggesting the model is quite stable. The main variable of interest, *NumProducers*, is introduced in model (4-3). The estimated coefficient on this variable is significant but rather small (.0002). Re-estimating 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 (.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 while allowing 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 0-500 producers and 501-1,000 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 earlier) and unchanging, consistent with Hypothesis 2.

### 6.2 Innovation Incentives

I now turn to how varying 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 impact the amount of time taken to

---

31 I continue to report results from a simple linear OLS framework. Results are unchanged in a count framework to reflect the fact the dependent variable is a non-negative integer.

32 Varying the particular combination of variables that is included has little impact on the results. I also investigated the use of a dummy for multi-homing producers, finding no significant relationship.
complete a new version. However, the analysis here particularly isolates how variation in numbers 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 (non-negative) durations between new versions using an IV Tobit framework.\textsuperscript{33} For an instrumental variable, I interact \textit{Employment} and \textit{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 essentially 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.\textsuperscript{34}

I first study relationships between \textit{Time to NewVersion} 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 \textit{Scale}, \textit{Experience}, and \textit{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 or wait longer to release new versions.\textsuperscript{35} The coefficient on \textit{Version} is not significant. To control for possible non-linear 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 \textit{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 following models, although results do not depend on choosing this specification.

I now turn to the explanatory variables of greatest interest related to numbers of producers. Here, I distinguish the number of producers within the same genre as a the focal producer, \textit{NumProducerSameGenre}, from those in different genres, \textit{NumProducersOtherGenres}. Adding these variables to the model without exploiting the instrumental variables results in positive coefficients, consistent with some unobserved factor that simultaneously encourage entry and investment in a given genre and platform and period of time. I then re-estimate the model with the instrumental variables specification, reported in model (5-3). The -.09 coefficient on

\textsuperscript{33} 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).

\textsuperscript{34} Because of the tiny size of individual producers, I disregard possible strategic timing of new versions (Ellison and Fudenberg 2000).

\textsuperscript{35} 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.
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 NumProducerSameGenre 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 crowding out of incentives among similar offerings (i.e., Hypothesis 4).

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, non-transitory title, producer, platform, and genre heterogeneity. This is a 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 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, in order to meaningfully interpret coefficients. I estimate, in model (5-5), the overall net effect by replacing the earlier measures with the aggregate measure of NumProducers. The .04 coefficient is positive, indicating, on average (weighted across all dampening and stimulating effects of all producers), the addition of producers caused an overall slowdown in the generation of new versions, on average.

6.3 Variation Generated by Subsequent Cohorts of Entrants?

Given marginal joiners had a negative impact on incumbent producers’ innovation efforts, it is natural to then assess whether perhaps the latter-joining (marginal) producers may have themselves contributed usefully to the basket of software generated. Some indication of how compelling was the software of later entrants can be gleaned by comparing dollar sales of titles across successive cohorts. Panel I of Figure 4 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 assure these patterns are not created by general trending or inherent differences across platform or genre, Panel II of Figure 4 plots linear fits of average sales of titles, having

36 Consistent with this interpretation, modelling pricing or volumes sold of individual titles finds analogous results suggesting crowding among similar producers and complementarities among less proximate producers.
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.

<Figure 4>

It remains possible that the “extreme tail” of titles developed by late-entering producers might still have meaningfully added compelling titles. However, Panel III of Figure 4 (showing sales of the 95th percentile) and Panel IV of Figure 4 (showing sales of the 99th percentile), reveals 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 much exceed this. Underlining this point, in the top three platforms, no single late entrant created an application that entered the top 100 sellers.

7 Discussion and Conclusions

It is becoming common for owners of computer platforms of all sorts to try to attract large numbers of independent producers to their platforms in order to generate an attractive selection of applications software. This might be understood as contrasting with usual notions of competition and innovation, where too much competition can quash innovation incentives (cf. Aghion et al. 2005, Aghion and Griffith 2008). However, in the case of applications 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 applications 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 applications software producers does not simply increase or decrease the innovation level, but rather qualitatively transforms the nature and sources of innovation.

Within the analysis, I began by assessing the usual prediction that increasing numbers of developers should cause increase in the variety of software titles. Consistent with the literature, I found not only that an increasing number of producers led to increased variety, it 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 numbers 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, and collective diversity of the pool of application producers in generating such a wide array of applications. Therefore, not only did the variety of applications benefit from adding producers, 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 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 numbers 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, however, it is important to note that a range of other mechanisms we might expect to be associated with heterogeneity—including spillovers, recombination and accumulation of ideas—could not be directly observed in these data. Further, it should be emphasized 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 an area for future research.

I then proceeded to examine whether there is evidence that varying numbers of producers might have also 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 allow me to interpret results in this manner.) This overall “crowding-out” of incentives with intensive competition is consistent with what we might expect in more typical contexts of industrial competition and innovation (cf. Aghion et al. 2005, Aghion and Griffith 2008). However, distinct from more typical industries, we also find 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 investment. On the other hand, adding producers tended to reduce incentives of producers of similar applications, within the same software genre (ex: spreadsheets and spreadsheets, games and games, etc.) The second effect overwhelms the first, on average in these data, leading to our 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 Burgh 2001, Yoo et al. forthcoming). However, the analysis also provided evidence of a tension between network effects and crowding, as has been suggested by several papers (e.g., Augereau et al. 2006, Church and Gandal 1992, Economides 1996, 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). The structure and ambivalence of this tension is perhaps more nuanced (in terms of substitution and complementarities across different genres). Beyond the patterns that could be directly observed in results, the earlier theoretical discussion further suggested that properties of the applications software context—an infinite frontier, uncertainty, scope for tweaking, features of the appropriability regime and so on—could themselves lead to important differences in the nature of competition and crowding than what we understand from more typical contexts of industrial innovation. This possibility remains an area for future research.

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 (cf., Baldwin and Clark 2000)—one that might potentially offset negative incentive effects (cf., Boudreau et al. 2011). I examined the sales of individual products as 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. Therefore, apart from the pronounced role of producer heterogeneity in this context, there is an equally clear non-random process of generation and sorting onto these platforms. These would appear to be first-order issues determining the selection of software and promise to be among the most important areas for future research.
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 upon the particulars of the context; nonetheless this case is an archetypal example of a hardware-software two-sided platforms within the academic literature and therefore these may be indicative of broader patterns in the economy.

Therefore, in bluntest terms, the results illuminate that a number of economic mechanisms shape patterns of innovation on “apps” platforms and not all will necessarily benefit from growth in numbers of producers. While the nature of analysis performed here does not allow a precise quantification of net value created or destroyed by adding to platform scale, the strength of descriptive patterns alone suggests that marginal entrants curtailed overall innovation. From a competitive standpoint, the results also illuminate that competition should depend on platform scale alone; there should be much scope to alter these innovation mechanisms—particularly given that platform owners may alter the conditions faced by apps 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. First, the findings related to an industry context with particular attributes: there were already a relatively large number of producers around platforms (an average of 860 per platform in each period, in the sample, on average), in a period of relatively stable growth. In this case, producers remained small and symmetrical, while new development remained rather incremental. The small and symmetrical character of suppliers follow the usual textbook characterization of applications producers around a platform; nonetheless these are simply common, rather than general conditions. More deeply, it should be noted that whether or not small and symmetrical producers emerge and are sustained over time might in large part be an outcome of platform owner decisions.

Also to note, the effects that were estimated were “at the margin”, in the sense of the effect of rather incremental variation in platforms where there were already roughly free entry conditions. This is an important point; it is quite possible that infra-marginal effects are quite different in magnitude,
and possibly in sign. For example, a newly-launched platform, with few applications developers, might only see applications innovation increase as it initially adds applications producers; and only when it reaches some larger number of producers do the effects eventually turn negative. Also, the analysis focused on variation in numbers of producers on a platform, and thus controlled for inter-platform competition and market structure. Finally, the empirical approach was devised while acknowledging the implications of interactions with the consumer side of the platform, without explicitly modelling these interactions.

Acknowledgments

I would like to thank 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. I would like to also thank participants in seminars and presentations at Carnegie Mellon, Dartmouth, London Business School, MIT, 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 WUSTL, 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 me to more deeply ponder the institutional and technical characteristics of applications 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 my own.

References


Ellison G. and D. Fudenberg. 2000 The neo-Luddite's lament: Excessive upgrades in the software industry.


Gick, W. 2010 Platforms or Franchisors? Two Contractual Forms When Consumers Care About Quality. mimeo.


Parker, G. and M. Van Alstyne. 2008. Innovation, Openness, and Platform Control. Available at SSRN:
Boudreau: Let a Thousand Flowers Bloom?

http://ssrn.com/abstract=1079712
Taudes et al. 2000 Options analysis of software platform decisions: a case study. MIS Quarterly 227-243
Tables

Table 1 Variable Definitions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Unit of Analysis</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>Producer-Month</td>
<td>Number of distinct genres (ex: databases, medical, games, etc.) in which a producer offers applications software</td>
</tr>
<tr>
<td>Varieties</td>
<td>Platform-Month</td>
<td>The sum of scope across all producers on a given platform</td>
</tr>
<tr>
<td>TimetoNewVersion</td>
<td>Title Version</td>
<td>No. of months between subsequent releases developed by a producer within a given system and genre</td>
</tr>
<tr>
<td>NumProducers</td>
<td>Platform-Month</td>
<td>Total number of active producers in a system</td>
</tr>
<tr>
<td>NumProducersSameGenre</td>
<td>Genre-Month</td>
<td>Total number of active producers in a system that are selling titles within a given genre of software (ex: databases, medical, games, etc.)</td>
</tr>
<tr>
<td>NumProducersOtherGenres</td>
<td>Genre-Month</td>
<td>Total number of active producers in a system that are selling titles outside of a given genre of software as the focal observation</td>
</tr>
<tr>
<td>Scale</td>
<td>Producer-Month</td>
<td>Total producer dollar revenues across all products in the time period</td>
</tr>
<tr>
<td>Experience</td>
<td>Producer-Month</td>
<td>Total cumulative past releases by the producer</td>
</tr>
<tr>
<td>Age</td>
<td>Producer-Month</td>
<td>The number of months that have passed since the producer entered the data</td>
</tr>
<tr>
<td>Version</td>
<td>Title-Month</td>
<td>The (integer) number of the current release of title within a given system and genre (i.e. first = 1, second = 2, etc.)</td>
</tr>
<tr>
<td>Time Trend</td>
<td>Month</td>
<td>Count of months since beginning of the data</td>
</tr>
<tr>
<td>Employment</td>
<td>Month</td>
<td>Total software jobs in the US, 1000s</td>
</tr>
<tr>
<td>Wages</td>
<td>Month</td>
<td>Average wage of those employed in software development in the US, $1000s</td>
</tr>
</tbody>
</table>

Table 2 Variable Means, Standard Deviations and Correlations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
<th>(9)</th>
<th>(10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) TimetoNewVersion</td>
<td>6.1</td>
<td>5.7</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) NumProducers</td>
<td>859.6</td>
<td>624.7</td>
<td>.06</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) Scope</td>
<td>1.7</td>
<td>.8</td>
<td>-.02</td>
<td>.05</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) Experience</td>
<td>1.2</td>
<td>7.3</td>
<td>-.11</td>
<td>.04</td>
<td>.15</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5) Age</td>
<td>19.1</td>
<td>13.4</td>
<td>-.02</td>
<td>.39</td>
<td>.29</td>
<td>.20</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(6) Sales</td>
<td>189.7</td>
<td>1640.8</td>
<td>.12</td>
<td>-.01</td>
<td>.18</td>
<td>.17</td>
<td>.32</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(7) Version</td>
<td>1.6</td>
<td>3.2</td>
<td>-.14</td>
<td>-.06</td>
<td>.31</td>
<td>.42</td>
<td>.25</td>
<td>.14</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(8) Time Trend</td>
<td>32.0</td>
<td>12.1</td>
<td>.15</td>
<td>.39</td>
<td>.02</td>
<td>.11</td>
<td>.17</td>
<td>.09</td>
<td>.24</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(9) Employment</td>
<td>451.9</td>
<td>65.1</td>
<td>.2</td>
<td>-.60</td>
<td>-.12</td>
<td>-.05</td>
<td>-.34</td>
<td>-.12</td>
<td>.04</td>
<td>-.62</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>(10) Wages</td>
<td>30.5</td>
<td>2.8</td>
<td>-.1</td>
<td>.24</td>
<td>.06</td>
<td>.03</td>
<td>.22</td>
<td>.18</td>
<td>-.02</td>
<td>.58</td>
<td>.22</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Note: 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, as it is directly constructed from Scope.
Table 3 Results of Variety Fixed Effect (IV) Regressions

<table>
<thead>
<tr>
<th></th>
<th>Dependent Variable = Varieties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(3-1)</td>
</tr>
<tr>
<td></td>
<td>(3-2)</td>
</tr>
<tr>
<td></td>
<td>(3-3)</td>
</tr>
<tr>
<td>Explanatory Variables</td>
<td>Platform F.E.'s</td>
</tr>
<tr>
<td>NumProducers</td>
<td>1.80*** (0.01)</td>
</tr>
<tr>
<td>Platform Fixed Effects</td>
<td>Yes</td>
</tr>
<tr>
<td>Platform Time Trends</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Notes. * *, **, and *** indicate statistical significance at the 10%, 5% and 1% levels, respectively. Robust standard errors in parentheses; explanatory variables are lagged; instrumental variables are Employment and Wages, interacted with platform dummies; number of observations = 393 platform-months.

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

<table>
<thead>
<tr>
<th></th>
<th>Dependent Variable = Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(4-1)</td>
</tr>
<tr>
<td>Explanatory Variables</td>
<td>Platform Covariates &amp; FE's</td>
</tr>
<tr>
<td>NumProducers</td>
<td>.0135*** (0.000)</td>
</tr>
<tr>
<td>Producer Characteristics</td>
<td>Scale</td>
</tr>
<tr>
<td>Scale</td>
<td>.0110*** (0.00)</td>
</tr>
<tr>
<td>Experience</td>
<td>.0122*** (0.00)</td>
</tr>
<tr>
<td>Producer Fixed Effects</td>
<td>Yes</td>
</tr>
<tr>
<td>Other Controls</td>
<td>Platform Fixed Effects</td>
</tr>
<tr>
<td>Platform Time Trends</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Notes. * *, **, and *** indicate statistical significance at the 10%, 5% and 1% levels, respectively. Robust standard errors in parentheses; explanatory variables are lagged; instrumental variables are Software Employment and Software Wages, interacted with platform dummies; number of observations = 121,503; number of cross-sectional units (suppliers) = 5,994.
Table 5 Results of *TimetoNewVersion* Tobit (IV) Regressions

<table>
<thead>
<tr>
<th>Explanatory Variables</th>
<th>(5-1)</th>
<th>(5-2)</th>
<th>(5-3)</th>
<th>(5-4)</th>
<th>(5-5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Covariates</td>
<td>Version F.E.’s</td>
<td>IV Tobit</td>
<td>Individual Title F.E.’s</td>
<td>Overall Net Effect</td>
</tr>
<tr>
<td><em>NumProducers</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.04***</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(.00)</td>
</tr>
<tr>
<td><em>NumProducersOtherGenres</em></td>
<td>-.09***</td>
<td>-.03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(.02)</td>
<td>(.02)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>NumProducersSameGenre</em></td>
<td>1.24***</td>
<td>.68***</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(.21)</td>
<td>(.23)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Producer &amp; Title Characteristics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Scale</em></td>
<td>.12***</td>
<td>.12***</td>
<td>.07</td>
<td>.09</td>
<td>.34***</td>
</tr>
<tr>
<td></td>
<td>(.03)</td>
<td>(.03)</td>
<td>(.08)</td>
<td>(.18)</td>
<td>(.07)</td>
</tr>
<tr>
<td><em>Experience</em></td>
<td>.01**</td>
<td>.02**</td>
<td>.01</td>
<td>.01</td>
<td>.10**</td>
</tr>
<tr>
<td></td>
<td>(.01)</td>
<td>(.01)</td>
<td>(.02)</td>
<td>(.06)</td>
<td>(.02)</td>
</tr>
<tr>
<td><em>Scope</em></td>
<td>.48****</td>
<td>.47***</td>
<td>.51</td>
<td>.16</td>
<td>.07</td>
</tr>
<tr>
<td></td>
<td>(.10)</td>
<td>(.10)</td>
<td>(.27)</td>
<td>(.64)</td>
<td>(.28)</td>
</tr>
<tr>
<td>Version</td>
<td>-.07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(.20)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version F.E.’s</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Individual Title F.E.’s</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform Fixed Effects &amp; Trends</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Genre Fixed Effects &amp; Trends</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Notes. *, **, and *** indicate statistical significance at the 10%, 5% and 1% levels, respectively. Robust standard errors in parentheses; explanatory variables are averaged over the periods during which a new version was developed; Software Employment and Software Wages, interacted with genre dummies; number of observations = 2,263.
Figures

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

Figure 2 Variety (Corrected for Platform and Time Effects) versus NumProducers
Figure 3 The Distribution of Producer Scope for Different Numbers of Producers on a Platform

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

Figure 4 Dollar Sales per Title in Successive Cohorts