PLATFORM BOUNDARY CHOICES & GOVERNANCE: OPENING-UP WHILE STILL COORDINATING AND ORCHESTRATING ☆

Kevin J. Boudreau

ABSTRACT

Rather than organize as traditional firms, many of today's companies organize as platforms that sit at the nexus of multiple exchange and production relationships. This chapter considers a most basic question of organization in platform contexts: the choice of boundaries. Herein, I investigate how classical economic theories of firm boundaries apply to platform-based organization and empirically study how executives made boundary choices in response to changing market and technical challenges in the early mobile computing industry (the predecessor to today's smartphones). Rather than a strict or unavoidable tradeoff between "openness-versus-control," most successful platform owners chose their boundaries in a way to simultaneously open-up to outside developers while maintaining coordination across the entire system.

Keywords: Platforms; firm boundaries; theory of the firm; platform regulation & orchestration; smartphones & mobile computing

JEL classifications: D4; E26; J4; L1; L8; 03

☆This chapter is based on the third chapter of my Ph.D. dissertation.

Entrepreneurship, Innovation, and Platforms
Advances in Strategic Management, Volume 37, 227–297

ISSN: 0742-3322/doi:10.1108/S0742-332220170000037009

INTRODUCTION

Since the turn of the century, "platforming" has become a leading mode of organizing exchange, innovation, development, production, consumption, and other forms of interactions in the economy and society (Choudary, Van Alstyne, & Parker, 2016; Ellison & Ellison, 2005; Evans & Schmalensee, 2016; Gawer, 2011; Hall, 2001; Levin, 2011; Rochet & Tirole, 2006). Rather than organizing along a traditional sequential "value chain" (Porter, 1985) of upstream and downstream firms or otherwise focusing on large-scale internal production capabilities (Chandler, 1962), platforms sit at the nexus of a multiple relationships to orchestrate value-creating interactions (Boudreau & Hagiu, 2009; Gawer & Cusumano, 2002). To varying degrees, we see many of the world's largest and most iconic firms – from Amazon to Uber – now implementing platform modes of organization (Evans & Gawer, 2016). Moreover, trends enabling platforming – such as networking, software, digitalization, automation, AI, and big data – are only accelerating and spreading across the wider economy (e.g., Altman, Nagle, & Tushman, 2015; Bessen, 2016; Branstetter, Drev, & Kwon, 2015; Evans, Hagiu, & Schmalensee, 2008; Sundararajan, 2016). As a result, increasing numbers of entrepreneurial businesses are "born open" as multi-sided platforms. At the same time, many traditional industry champions "born closed" are attempting to open-up to incorporate platform approaches (e.g., Altman & Tripsas, 2014; Chesbrough, 2013; Robertson & Ulrich, 1998; Zhu & Furr, 2016).

This chapter investigates a most basic question of platform-based organization, that of boundaries. Despite the conspicuous emergence of platforms in the economy, there is yet little research studying boundaries from the vantage point of organization and governance. Within economic and management research, most existing models proceed with a simplifying assumption that platform boundaries are "given" and one and the same with the platform owner's own economic scope – and then proceed to investigate other questions of interest on the basis of this simplifying set of assumptions (Church & Gandal, 1992; Katz & Shapiro, 1994; Parker & Van Alstyne, 2005; Rochet & Tirole, 2006; Weyl, 2010). (Exceptions are reviewed herein.)

The bulk of existing work on platform scope and boundaries relates to an internal logic of technology and design, whereby benefits of variety and modularity are traded off against integral design within a standardized and reused platform (e.g., Eppinger & Ulrich, 2015; Matutes & Regibeau, 1988; Robertson & Ulrich, 1998; Sanderson & Uzumeri, 1995; Simpson, Maier, & Mistree, 2001; Suh, De Weck, & Chang, 2007). In highlighting these principles of good design, it remains a question whether governance and organizational factors might play an independent or at least accompanying role in shaping boundary choices.

Most crucial boundary choices in platform-based organization are the choice of platform boundaries (i.e., the scope of a system's components and elements that are standardized and reused in a system as a single variety[1]) and choice of platform owner boundaries (i.e., the economic scope of production of the platform's owner). Jointly, these choices determine much of the overall organization of a system. For example, components that fall outside of platform scope and not exclusively controlled by the platform owner are effectively "opened up" – to be supplied as substitutable mix-and-matchable "modula" components by outside independent suppliers ("complementors," "contributors," "solvers," "independent component suppliers," or even "users"). The past literature then emphasizes possible benefits of outsiders' diversity, distributed productive knowledge, comparative advantages, and best-of-breed specialization and capabilities (e.g., Baldwin & Clark, 2000; Baldwin & von Hippel, 2011; Boudreau, 2012; Boudreau, Lacetera, & Lakhani, 2011; Chesbrough, 2003; Gambardella, Raasch, & von Hippel, 2016; Garud & Kumaraswamy 1995; Katz & Shapiro, 1994; Langlois, 1992; Meyer & Lehnerd, 1997; von Hippel, 2005). The analysis here attempts to add depth to our understanding of trade-offs in setting platform and platform owner boundaries. The starting point is to consider how well-established economic theories of firm boundaries (relating to the classical make-or-buy problem) might somehow apply to platform-based organization. Established economic theories suggest boundary choices are made in relation to several goals:

1. Allocating investment incentives (e.g., Grossman & Hart, 1986; Hart & Moore, 1990);
2. Achieving internal coordination by integrating around coordination problems (e.g., Williamson, 1975); and
3. Achieving external coordination of multiple independent agents or actors by exerting control over critical assets of production (e.g., Holmström, 1999; Holmström & Milgrom, 1991).

These theories are argued to apply (somewhat) differently from what would be expected in traditional firms and make-or-buy problems. In relation to allocating incentives, narrowing platform and platform owner boundaries should do little to convey higher incentives to independent suppliers to participate and make investments – if narrowing boundaries does little to affect the asymmetric power held by the platform owner over trade partners. In relation to internal coordination, established theories should translate more or less directly to platform contexts – with most difficult coordination problems coordinated within the platform, itself. In relation to coordinating independent agents, widening platform boundaries to encapsulate "critical" components (defined herein) should convey to the platform owner greater ability to coordinate ("orchestrate," "regulate," and "strategically sponsor") actors beyond its own boundaries.

In empirically investigating these claims, a first challenge is to observe a set of meaningful comparisons (of boundary choices) from which inferences might be derived. To do so, I document boundary choices in the early history of mobile computing platforms (from the 1980s through to the early 2000s). In this context and period, it was possible to observe multiple comparable systems and to do so over time. Therefore, although boundaries might be chosen differently, each case adhered to the same basic architecture of elements defining the system.

This industry and period are also notable in being the early pioneering beginnings of a platform industry when executives faced a varying array of technical and market conditions. In each case, they needed to design and build new systems, while attracting users and independent component suppliers to come "on board" their platforms. This early growth and innovation period is typically a most critical period in the determining the success of a platform and surrounding system (Bessen & Farrell, 1994; Evans & Schmalensee, 2010; Suarez, 2004).

A second challenge in the empirical analysis is to interpret patterns, discriminating among alternative theoretical interpretations of observed patterns and possible confounding factors. To do so, the approach used here is – for each and every major organizational and boundary choice across systems and over time – to document executives' justifications and rationale for each decision, while also observing the resulting impact. The analysis thus provides "thick" historical description across leading pioneering platforms in the industry (Psion, Palm, Microsoft and leading Mobile Linux supplier, Mobilinux), while referring to systematically collected accompanying descriptive data.

Summary of Empirical Findings

The basic descriptive patterns, even on their own, readily suggest that platform boundaries cannot be taken as "given." Despite observing comparable platforms, in the same industry, with the same architecture, and at the same time, there was considerable cross-sectional variation and regular changes over time in boundary choices. Platform and platform owner boundaries varied independently. There was no single tendency in trending, as changes in some instances were toward wider boundaries and other cases to toward narrower boundaries.

Clearly, technological design considerations did play a role in shaping platform scope and definition, as would be predicted from past theory in systems design and product management. Differences across platforms in large part reflected executives' own subjective design priorities at the time of system founding. For example, designs featuring low-performance component technologies required greater co-specialization to achieve high performance and therefore tended to be organized in a highly integrated way, with a wide

platform – as in Psion and Palm's early histories. By contrast, attempts to produce exceedingly modular designs coincided with narrow boundaries, as in Microsoft's commitment to modular component reuse or Mobilinux's commitment to remaining highly modular and flexible in hopes of working with large partners. However, even these examples readily reveal that technical design choices were hardly independent of organizational and governance considerations.

In relation to organizational and governance-related determinants themselves, the observed patterns are each consistent with each of the three main predictions. Most conspicuous was the importance of widening platform boundaries to directly integrate challenging integration problems. A number of different types of coordination challenges emerge in these histories; however, most fundamental to the success of these systems was the ability to coordinate cross-component design decisions across the system. Thus, these issues emerged in a context of imperfect and evolving technologies and necessarily imperfect modularity (see Staudenmayer, Tripsas, & Tucci, 2005). For example, both early Palm and Psion carried out meticulous design decisions blurring lines between hardware and software teams to overcome these challenges.

Integration and wider boundaries also quite clearly played a role in orchestrating, supporting, and sponsoring activities of outside independent suppliers. Contrary to what "opening-up" might casually connote, opening components such as software application development did not come so much from disintegrating or narrowing platform boundaries, but rather from integrating, investing, and widening boundaries into critical components and interfaces meant to enable opening-up. For example, Microsoft would find greater success once it more directly integrated into the creation of reference designs for its hardware partners. There are many other such examples.

Finally, narrowing platform boundaries was not an effective means of boosting investment and participation by independent suppliers and complementors. This was attempted in multiple instances, as in later Palm and Psion strategies and in Mobilinux without achieving desired ends. What was found to be productive in this regard was to take added means, such as contracting or transfer of ownership to create inducements. For example, Psion devolved control and equity to hardware designers; Palm engaged with a series of ad hoc contracting with hardware developers to entice them to work on its platform and to deliver new innovations. These results are consistent with prior research stressing the use of various managerial practices (other than boundaries) for platform owners or protections by complementors to encourage investments (e.g., Huang, Ceccagnoli, Forman, & Wu, 2013; Gawer & Henderson, 2007; Perrons, 2009).

The arguments herein clarify how the usual logic of shifting boundaries in a make-or-buy problem (e.g., Grossman & Hart, 1986), or classical ideas of effects of "open" standards (e.g., Cusumano, Mylonadis, & Rosenbloom, 1992; Katz & Shapiro, 1994; Shapiro & Varian, 1998; Simcoe, 2006) do not directly translate to platform contexts. This is because narrowing platform boundaries

of a platform does not necessarily change the fact that the platform remains a "bottleneck," the pinch-point in a series of one-to-many relationships with trade partners. An overall implication of these findings is that although each of these systems was organized around open platforms that drew on external suppliers' diversity and comparative advantages, seeking to have narrowest boundaries and to be "most open" was not a path to greatest adoption, investment or innovation in the system. Platform owners here faced the challenge of harnessing contributions of outsiders while at the same time assuring coordination across the system. Thus those open platforms for which platform owners retained considerable control were most successful. This was not simply a "lever" to tradeoff, but rather a challenge of carefully devising an effective organizational approach to platform-based organization.

LITERATURE AND BACKGROUND: PLATFORMS AND PLATFORM-BASED ORGANIZATION

This section provides background and terms of reference from the existing literature.

Platforms as a Technical Design and Product Management Approach

Platform boundaries necessarily meld questions of technology and design, with those of economics and organization (e.g., Gawer, 2014; Porch, Timbrell, & Rosemann, 2015). Following a technological design perspective, platforming follows principles of design "decomposition" (Alexander, 1964; Parnas, 1972; Simon, 1962). this means subdividing a system in a way that groups together tightly interacting core components (i.e., design problems) at the nexus of a system's architecture (e.g., Baldwin & Woodard, 2009; Eppinger & Ulrich, 2015; Robertson & Ulrich, 1998; Sanderson & Uzumeri, 1995; Simpson et al., 2001). Platforms are then, as a result, foundational to a system in the sense of embodying reconfigurable, general technological capabilities (Bresnahan & Greenstein, 2014). Therefore, technological design and product management research already imply there should be an internal logic of design that determines an optimal definition of at least the scope, definition and boundaries of the platform, itself.

Therefore, from the technological design perspective, the platform should consist of core design elements that can be used in common, standardized, and "reused" across the system, inasmuch as economies of standardization and reuse outweigh benefits of instead realizing multiple mix-and-matchable

versions, varieties, and implementations (Adler, Mandelbaum, Nguyen, & Schwerer, 1995; Jose & Tollenaere, 2005; Krishnan & Gupta, 2001; Krishnan & Ulrich, 2001; Matutes & Regibeau, 1988).

For example, an operating system sits at the heart of a computer system, taking instructions from a wide range of application software programs and translating and directing instructions to system components and analog subsystems. The frame, suspension, and drive-train of an automobile's design can be reused across multiple automobile models. A battery and power system, electric motor and control system can be reused across a family of power tools. Standardized "interfaces," protocols, gateways and rules can themselves be platforms in the sense of serving as key technological objects that are reused and which ensure coordinated use across hardware and software components, as in say media and media players, and web-based platforms that interact with other machines or humans. Thus, a practical working definition of the platform boundaries or scope is simply as those components or system elements that are used in common, available in a single standard variety, or equivalently reused across the system design. Platform boundaries may vary independently of the economic scope of production of the platform owner.

Platforms as a Mode of Economic Organization

An economic and organizational perspective on platforms refers to a distinct mode of organizing production and exchange (e.g., Boudreau, 2010; Church & Gandal, 1992; Hagiu, 2007; Katz & Shapiro, 1994; Langlois, 1992; Parker & Van Alstyne, 2005; Rochet & Tirole, 2006; Rysman, 2009). Platform-based organization puts the platform owner at the nexus of a series of one-to-many interactions among different sorts of surrounding independent suppliers of complementary, mix-and-matchable components and users. For example, Google's Android smartphone operating system platform sits between and governs interactions between developers of apps, phone designers, makers of peripheral devices, and users. Much of the existing literature stresses how there can be interactions and network effects acting within and across different sorts of actors around a platform (e.g., Boudreau & Jeppesen, 2015; Eisenmann, Parker, & Van Alstyne, 2006).

The position at the nexus of exchange and interactions creates the opportunity for a platform owner to promote the overall success, competitiveness, and value creation in the system, while also creating an opportunity to capture a share of this value (Adner & Kapoor, 2010; Shapiro & Varian, 1998; West, 2003). Most straightforward ways a platform owner might use to help implement productive outcomes is, for example, to carefully set prices across different types of platform participants (e.g., Hagiu, 2006; Jin & Rysman, 2015;

Seamans & Zhu, 2013), to provide subsidies, and to engage in "strategic sponsorship" (i.e., investment in one form or another) to encourage entry and investment (e.g., Katz & Shapiro, 1986; Shapiro & Varian, 1998). Other instruments include information provision (e.g., Tucker & Zhang, 2011), selective granting of platform access and licensing (e.g., Boudreau, 2012) or certification (Tarnacha & Maitland, 2010), provision of enabling tool-kits (e.g., Jeppesen, 2005), the creation of trust and reputation mechanisms (Perrons, 2009), and – of course – contracting with surrounding parties. This considerable scope for coordination has lead platform owners sometimes to be described as "leaders" (Gawer & Cusumano, 2002), "orchestrators" (Iansiti & Levien, 2004), and "regulators" (Boudreau & Hagiu, 2009) in their wider ecosystems. Most profoundly, platform owners often have the power to entirely (re)define the institutional framework on different sides of the platform. For example, component suppliers, complementors, and contributors working on and around a platform might be arranged to work in a market, a rank-order tournaments or crowdsourcing setup, or a collaborative community (Boudreau & Lakhani, 2009, 2013; Felin & Zenger, 2014).

PLATFORM AND PLATFORM OWNER BOUNDARIES?

Questions of organizational design, governance choices, and boundaries remain a hotly contested and actively researched in a number of literatures (e.g., Gibbons, 2005; Gulati, Puranam, & Tushman, 2012; Santos & Eisenhardt, 2005; Whinston, 2001). In exploring the question of boundaries in platform contexts, the approach taken is to consider how predictions of three established economic theories[2] of boundaries – as usually applied to make-or-buy problems – might apply instead to platform-based organization. Thus, hypotheses here consider how platform and platform owner boundaries might achieve following goals: (i) allocating incentives; (ii) internalizing coordination problems; and (iii) consolidating control over critical assets to orchestrate surrounding component suppliers. In a nutshell, arguments here suggest that narrowing platform and platform owner boundaries do not necessarily produce the accustomed effect of conveying higher incentives to outside independent suppliers. Moreover, widening boundaries can have the effect of improving coordination across the system – both inside and beyond the platform owner's boundaries.

Using Boundary Choices to Allocate Investment Incentives

Property Rights theory (Grossman & Hart, 1986; Hart, 1995; Hart & Moore, 1990) emphasizes the need to make relationship-specific investments

and hazards thereof.[3] This theory emphasizes the decision to integrate or disintegrate a given step of production bargaining will shape bargaining positions of appropriable rents. The party who retains control and ownership over key assets of production will retain residual income and control rights and therefore have higher incentives to make investments. And so, following the Property Rights theory, firm boundaries around assets of production should set to give control to the party whose (uncontractable, relationship-specific) investments are most important to value creation. In the standard formulation, such an allocation will make all parties better off, so long as it is possible to contract over any reallocation of assets and to make associated payments and transfers to realize the value-maximizing approach.

We might expect questions of power[4] and bargaining position (and associated residual income and control rights) implied by Property Rights theory to somehow play an important role too in platform-based organization. However, a starting point is to recognize that the platform owner will typically have a highly asymmetric level of power and bargaining position over its trade partners.[5] As the owner of a central enabling bottleneck, the platform owner's power and bargaining position often towers above that of independent suppliers of mix-and-matchable complementary components. Platform owners will often have both "bouncer's rights" (Strahilovetz, 2006) to grant access to the system and stipulations associated with access.

Extreme asymmetry of power might not necessarily harm incentives or impair trade relationships if terms of trade can be contracted under agreeable terms. For example, a crowdsourcing platform can stipulate a prize for solving a given problem and convey the rules of a contest. For example, Apple stipulates that 30% of all app developers' revenues will go to Apple as a variable platform "tax" of sorts. However, outsized platform power will most often be associated with many *uncontractable* levers and avenues for *ex-post* value capture through its bargaining position (e.g., Bakos & Brynjolfsson, 1993a, 1993b, 1999; Jacobides, Knudsen, & Augier, 2006; Kim & Kogut, 1996) and "profit-squeezing" (Farrell & Katz, 2000). "Lock-in" and hold-up (Farrell & Klemperer, 2007) of trade partners by a platform owner might take many forms. In the earlier crowdsourcing example, for example, it may be difficult to specify and contract on fully objective evaluation criteria. In the example of Apple, the (uncontractable) market structure and competition among app developers might lead app developers to face extremely high levels of competition from zero price apps, effectively "commoditizing the complementor" (Shapiro & Varian, 1998).[6]

Most relevant to the boundaries question, theorists have particularly highlighted threats of platform owners integrating (entering) into markets for complementary goods and thereby disadvantaging independent suppliers (e.g., Becchetti & Paganetto, 2001; Choi, Lee, & Stefanadis, 2003; Farrell & Katz, 2000; Heeb, 2003; Martin & Orlando, 2007; Niedermayer, 2013;

Nocke, Peitz, & Stahl, 2007). It is also possible that platform owner might expand platform scope by dragging the functions offered by independent suppliers into the platform itself ("enveloping" those functions), making independent suppliers unnecessary (Eisenmann, Parker, & Van Alstyne, 2011).

Analogous to lessons from Property Rights theory, the problem from this asymmetric power and expropriation, is that it may quash independent suppliers' incentives to participate and to make platform-specific investments. This is especially important where the platform owner depends on the heterogeneity, diversity and comparative advantages of independent component suppliers (Baldwin & Clark, 2000; Boudreau, 2012; Boudreau, et al., 2011; Cennamo & Santalo, 2015, p. 15; Farrell & Katz, 2000; Katz & Shapiro, 1994).[7]

For simplest and narrowest platforms – as in say a set of interoperability standards or a set of instructions within a software platform – it may be possible to resolve this problem of the threat of expropriation by placing the platform in the public domain as when publishing a standard to a standards organization or a software platform is open sourced. However, under many conditions, absolute devolution of control might not be practicable or desirable. Further, where any measure of bottleneck control is retained, the owner retains asymmetric power. Thus, it does not follow that disintegrating from a particular market for complementary components removes the threat of expropriation.[8]

The existing related empirical research, for example, does not emphasize narrowed platform boundaries *per se* as a means of promoting investments, but rather sheds light on a range of management practices to attempt to credibly signal forbearance by the platform owner and an enduring and trusting relationship with independent suppliers (see Gawer & Henderson, 2007; Perrons, 2009). Further, Huang, Ceccagnoli, Forman, and Wu (2013) also find that some degree of residual control and income rights might be effectively retained by a platform owners' trade partners where they possess intellectual property rights.

Therefore, whereas classical Property Rights theory predicts transformative effects of reassigning boundaries on the allocation of incentives, we should expect little such effect in the case of incremental variation in platform-based organization where shifts in platform or platform owner boundaries do not alter the bottleneck status of the platform (owner). Implications are summarized in the following hypothesis:[9]

> **Hypothesis I:** Narrowing platform or platform owner boundaries is not an effective means of encouraging outside trade partners to make platform-specific investments, inasmuch as it leaves the platform owner as a bottleneck.

Using Boundary Choices to Solve Coordination Problems through Direct Integration

Transaction Cost Economics (Williamson, 1975) originated with the idea that for whatever inefficiencies integrated firms may or may not suffer in relation to market coordination, integrated firms will have privileged access to instruments of coordination.[10] The simple prescription is then that activities involving pronounced coordination problems should be placed inside the same firm boundaries. In particular, Transaction Costs Economics often emphasizes costs of "hold-up" and "haggling" and inefficient rent-seeking activities in cases of market exchange that could arise where relationship-specific investments create appropriable rents (Klein Crawford et al., 1978; Williamson, 1975). However, interpreted more generally, wider scope and wider integration might solve any number of coordination problems emanating from asymmetric information, economic spill-overs, investment externalities, problems of the left-hand-knowing-what-the-right-hand-is-doing kind.

Integrating decisions within the same firm may, for example, enable the use of authority and decisions by fiat in resolving conflicts. The interests of parties belonging to the same firm might also be more closely aligned for that simple fact. Cooperating might also be better sustained by the threat of alienation from the firm or any number of other of forms of organizational control (Williamson, 1971). Numerous other theories of the firm offer complementary perspectives on why firms may have advantages in solving coordination problems, including common experience, language, organizational knowledge and perhaps collocation and common experience of workers (e.g., Kogut & Zander, 1992; Sosa, Eppinger, & Rowles, 2000, 2003).

Research on multi-component systems and platforms has similarly recognized the benefits of integrating multiple components and elements of a system under a given supplier (Becchetti & Paganetto, 2001; Katz & Shapiro, 1986). A single integrated supplier is argued to have advantages in implementing better coordination than decentralized component suppliers in achieving coordinated and "coherent" technological advance of a system across multiple components (Bresnahan & Greenstein, 1999; Gawer & Cusumano, 2002). An integrated supplier might also have advantages in developing and implementing technical standards necessary to ensure interoperability of components (Bessen & Farrell, 1994; David & Greenstein, 1990; Farrell & Saloner, 1985), and making globally optimized pricing decisions across components (Davis & Murphy, 2000; Davis, MacCrisken, & Murphy, 2002), along with bundling decisions (Bakos & Brynjolfsson, 1999; Carlton & Waldman, 2002) or to internalize other forms of externalities (Farrell & Weiser, 2003).

Theories of technological determinants of boundaries (see Introduction) and "modularity" of design can themselves be recast as examples of setting boundaries to minimize coordination problems. The idea here is that subdividing the

architecture into separate design problems (modules), linked by rules (interfaces) typically makers the design problem more tractable (Baldwin & Woodard, 2009; Simon, 1962), as design teams can closely coordinate and organize around tackling one problem at a time. In this instance, however, boundaries are set to improve coordination by reducing explicit coordination among groups. However, here too, there might still be benefits of integrating across multiple components to achieve more globally optimized trade-offs in cases where interfaces and modularity are imperfect, and there remains scope for coordination (Staudenmayer et al., 2005).

Following preceding points, the core ideas of Transaction Cost Economics should, therefore, translate somewhat directly from the classical make-or-buy scenario to multi-component systems and platforms. However, the topology of platform-based organization (a platform bottleneck, reused in a single variety, surrounded by mix-and-matchable components) might relate to degrees or levels of coordination. For example, greatest mutual dependencies defining the core of the system might most naturally be placed within the platform, itself (Baldwin & Woodard, 2009). (Where the platform owner's boundaries fail to entirely encompass most important coordination in the platform, we might expect costliest problems.) The platform owner might also integrate beyond the scope of the platform, from this perspective of integrating over coordination problems, if to integrate imperfectly modularized mixed-and-matched components. However, to emphasize the point, the direct translation of existing principles of Transaction Cost Economics implies too that even independent developers might integrate across multiple components or elements in the wider system, where there are opportunities to integrate coordination problems not inclusive of the platform, itself. Most central implications are summarized in the following hypothesis:

> **Hypothesis II:** Widening platform and platform owner boundaries is an effective way to resolve coordination problems, through direct integration; this is especially so where coordination is carried out within the platform and by the platform owner.

Using Boundary Choices in Gain the Power to Orchestrate the "Ecosystem"

Whereas the Transaction Cost Economics perspective emphasizes coordination and productive alignment *within* the boundaries of a given firm, platform owners frequently engage in "leadership" and coordination *across* the wider ecosystem of multiple firms and actors. The platform owner might therefore be understood to act as the central planner, regulator or "orchestrator" of the wider ecosystem (Boudreau & Hagiu, 2009; Gawer & Cusumano, 2002; Iansiti & Levien, 2004). This subsection considers how Agency Theoretic perspectives of

organization might shed light on this role of external coordination, and how this could relate to choice of platform and platform owner scope.

Agency Theoretic perspectives emphasize inefficiencies arising when one party (a "principal") gets its work done through other parties (the "agents") (Holmström, 1979). To minimize agency costs, a principal will devise a system of incentives or even rules and constraints for workers, to regulate agents' choices and behavior in ways that are productive (e.g., Alchian & Demsetz, 1972; Holmström, 1999; Holmström & Milgrom, 1991) and by regulating agents' behavior in ways that contemplate possible interactions and externalities among multiple agents (Segal, 1999).[11] Control over critical assets also, of course, allows the firm owner to directly decide who can join the firm in the first place (Rajan & Zingales, 1998).

An important basis for this "power to regulate" is that firms control the critical assets of production (Hart & Moore, 1990[12]; Holmström, 1999; Holmström & Milgrom, 1991). Agents or workers in the firm are willing to be subjected to the firm's system of incentives and rules, only so long as gaining access to the firm and its assets of production will raise workers' returns higher than would otherwise be the case. Therefore, this Agency Theoretic view raises the question of scope in the sense of which assets should be held exclusively by the firm owner to maintain this productive power to define incentives, rules, and constraints within a given sub-economy.[13]

In the case of platforms and external suppliers and actors in the surrounding ecosystem, several of the earlier issues apply in a somewhat isomorphic fashion. If access to the platform raises the productivity of surrounding independent suppliers – then the platform owner can stipulate conditions of access to which surrounding actors are willing to accede. This implies that components within the scope of the platform should be highly complementary to outside actors, while allowing the platform to maintain "bouncer's rights" to them (Boudreau & Hagiu, 2009; Strahilovetz, 2006). This ability of platform owners to contract with outsiders upon granting access has been readily noted in the literature, with platform owners being analogized as "licensing authorities" (Rochet & Tirole, 2006), "public regulators" (Farrell & Katz, 2000), or "regulatory commissions" (Boudreau & Hagiu, 2009; Rochet & Tirole, 2006). Thus, it may be crucial that the platform owner retains the ability to grant – or withhold – access to the platform by controlling critical assets (Boudreau, 2010)

Apart from formal contracts stipulating actions and conditions, the literature has also noted a role played by rules, restrictions, and pathways to taking certain actions that are embedded into the technology platform, itself. For example, the architecture and application programming interfaces are themselves rules that limit and guide what can be designed within a system. In this regard, authors have also emphasized the importance of retaining control over those components that convey "architectural control." Architectural control has been explicitly linked to scope, as past research points to specific

architectural control "points" in the system architecture that are important to maintain control over (Lessig, 1999; Morris & Ferguson, 1992; Woodard, 2008).

A separate but closely related set of arguments in the literature reflects on the platform owner's ability and incentives to take privately costly actions in order to enable greater value creation overall in the ecosystem, or to act as a "system sponsor" (David & Greenstein, 1990; Katz & Shapiro, 1994). Beyond just investing in coordination of actions and activities, this might too include making common investments, and strategic subsidies (Katz & Shapiro, 1986).

Following these points, this Agency Theoretic perspective indicates that the boundaries and scope of the platform that the platform owner controls should extend to "critical" assets or components within the system. Earlier arguments suggest that "critical" here should be understood to include those components that (i) are highly complementary (i.e., enhancing) to a wide variety and number of external actors; (ii) serve as a basis for feasibly bottlenecking the system and excluding unauthorized access to the platform; (iii) embody rule-making and architectural control; and (iv) serve as a source of appropriability over the value created in the system.

The following hypothesis summarizes implications:

> **Hypothesis III:** Widening the technical platform (over which the platform owner exerts control) to include "critical" elements of a system allows the platform owner to coordinate or "orchestrate" external actors in the ecosystem.

The foregoing arguments highlight that many of the taken-for-granted attributes or even *definitions* of platforms (that they be components that are generalizable and value-creating to many different actors, that they be the one in a one-to-many system, that they be surrounded by architectural control points) are perhaps better understood as functional *consequences* of boundary choices.

EMPIRICAL APPROACH AND DATA

The remainder of the chapter is devoted to empirically investigating the hypotheses developed in the preceding section. A first research design challenge is to observe variation in boundaries, making meaningful comparisons across systems, components, and over time – in response to varying conditions. A second key challenge is to derive inferences when platform choices are shaped by a number of factors at the same time. This likely includes not only those factors hypothesized earlier, but also technological design considerations, and other possibly confounding factors.

The research design approach to deal with these challenges involves studying the early history of the mobile computer industry, a predecessor to today's

smartphone industry. This context provides an opportunity to document boundary choices in multiple, comparable, and relatively simple systems in the same industry and period. Beyond just documenting covariation of boundaries and changing conditions, the analysis involves and primarily relies on directly documenting executives' motivations for making boundary choices and observing consequent outcomes.

Empirical Context: Early Mobile Computing Platforms

Experiments in the miniaturization computers, mobility, and related pen-computing interfaces began to accelerate in the 1980s led by large computer and technology firms and a handful of startups across Europe, United States, and Japan. Isolated commercial successes began to be registered by the 1980s, achieving tens of thousands of shipments in sales (Fig. 1). In the early 1990s, shipments expanded to hundreds of thousands of units, and increasing numbers of large and smaller venture-funded entrants attempted to launch new platforms and new technology, in hopes of growing and dominating the mobile computing industry, just as IBM had done in mainframes and Microsoft and Intel had done in personal computing. Products in this period were still dominated by "tablets," "palmtops," and "handhelds" and other variants and types of small mobile computers; however, a number of devices in this period also enabled networking and even wireless networking, foretelling the smartphone industry that would arrive in earnest a decade and more later (Fig. 1). By the end of the 1990s, industry-leading platforms began to shift development and production toward smartphones, as telecommunications technical standards bodies began to develop plans for data-enabled future generations of public

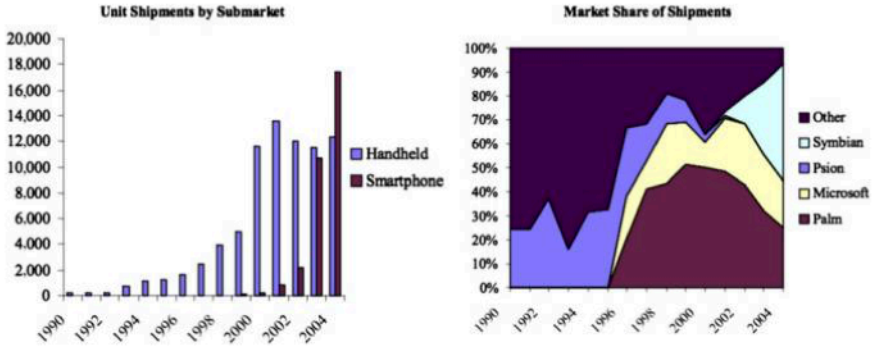

Fig. 1. Market Size by Shipments, and Shares by Platform. *Source*: Taken from Boudreau (2010). *Notes*: Data sources are IDC and Gartner. Unit shipments are in thousands.

commercial mobile radio (cellular) networks in North America and Europe. By the early 2000s, Japans's Docomo service already demonstrated an early version mobile phones carrying data services. This roughly quarter-century of industry history took place before the popularization of RIM's Blackberry, the Apple iPhone, and Google's Android would transform the industry to the smartphone industry we know today.

The analysis here focuses on this early history of the industry and includes the three industry leaders at this early history of the mobile computing industry: Psion (later to transform to Symbian), Palm, and Microsoft (Figs.1 and 2). Studying these early industry leaders also provide opportunities to study relatively long histories, including both periods of success and of failure. To complement lessons from these comparisons, the analysis also includes the leading Linux-based mobile computing platform owner of this period, Mobilinux. The analysis proceeds from the founding of these platforms until 2004. Proceeding to 2004 allows the transition from (non-networked) mobile computing to smartphones to be observed, during a period in which the platforms under study attempted to succeed in this transition. By the middle of the 2000s, new suppliers (particularly RIM, and then Apple and Android) would rise in prominence (Fig. 1).

Notwithstanding differences in the organization of these systems (as will be documented here) and differences in design choices, the essential architecture of these systems was similar and comparable in important ways during this early history of the industry. At the highest level, these platforms would be referred

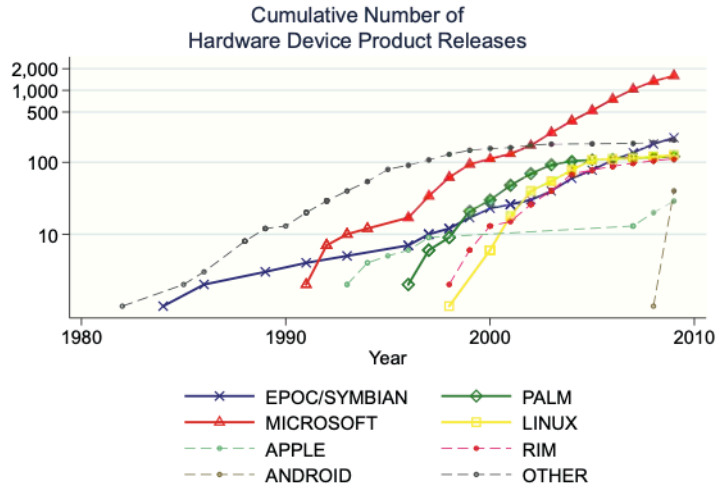

Fig. 2. Product Releases Across the Industry and Over Time. *Source*: Data are sourced from hand-collected data documented in Boudreau (2010) and market research firm PDA.DB.

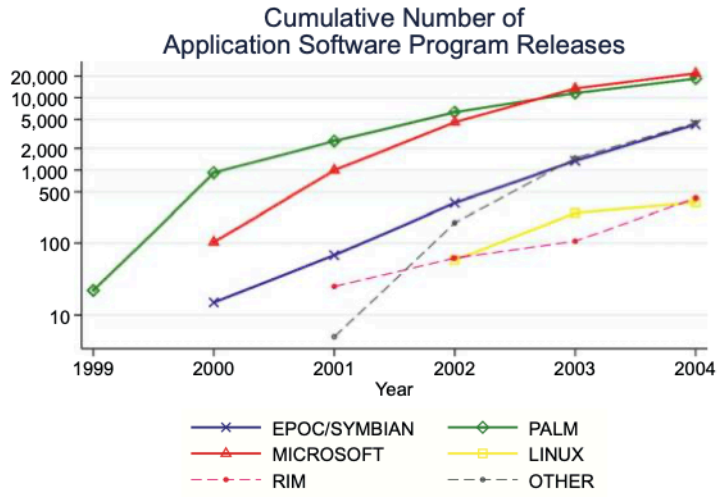

Fig. 3. Product Releases across the Industry and over Time. *Source*: Data are sourced from Handango.com, the leading online marketplace for mobile computing and mobile phone apps in this period, documented in Boudreau (2010).

to as multi-sided platforms in the modern literature, both hardware developers and software developers building "on top" of platforms (Figs. 2 and 3).

Data and Measurement

System components: The essential structure and outlines of system definition were comparable across each of the studied systems and remained stable over time. Thus, the organization and boundaries in these platform-based systems can be recorded in relation to a stable and defined set of elements or components:

- *Operating system*: The analysis distinguishes between the low-level kernel of the operating system versus individual utilities to drive particular analog components of the system, versus graphical user interface with which users interacted.
- *Mobile computing hardware (device and board-level design and integration)*: This includes all system-level electrical engineering and industrial design of hardware components and their integration into a working device. The analysis distinguishes between the broad definition of hardware and system requirements versus more detailed component or "board-level" configuration and design, versus the creation and supply of complete working designs.

- *Central processing unit (CPU)*: By this time, all main processing was carried out by a single, integrated microprocessing unit. The analysis also records whether the platform and platform owner integrated into any drivers and creation and specialization of low-level software controlling the central processing unit.
- *Device applications software*: This includes all specialized applications software that calls upon the operating system. The analysis also records the provision of supporting frameworks and tools that allow the development of applications software (as these components needed to be built and were not always supplied by the platform owner).
- *Peripheral hardware*: These include all plug-in devices, from printers to cameras and modems, and so further.
- *Network services*: These are all facilities and services beyond the mobile computing device, itself. This includes network connections and connectivity, any sort of hosting framework and definitions upon which content and services can be defined, and any content and services, themselves.

Platform boundaries and platform owner boundaries: *Platform scope and boundaries* are defined as those components that are (re)used in common across the system, and therefore available in but a single variety. *Platform owner scope and boundaries* are defined as those components that fall under the economic scope of production of the platform owner – those components the platform owner develops and produces.

Main data set construction: A data set was constructed to indicate which components fell under the scope of the platform, and which components fell under the scope of the platform owner in each system and over time. This was done by recording month-by-month changes in each element of the architecture in each system by recording all press releases and all articles on each platform owner and each of their main trade partners in the Lexis-Nexus database from the time of founding of the platform until 2004. Main patterns and changes are presented graphically and in text description in the following case studies. These data are complemented market data and data on details of the technology, with multiple sources cited in the pages to follow.

Interpretation of Patterns

The case studies that follow attempt not just to summarize a description of patterns but also to provide a narrative of the reasons for executives' decision-making at the time. This is first done on the basis of the very same press releases and articles in the Lexis-Nexus database used to produce the patterns of platform and platform owner scope, as described above. The case studies presented here are intended first to essentially summarize the narratives

provided in articles and press releases, while cohering them in an overall narrative. rationale underlying changes, as conveyed in press releases and articles.

Beyond the articles and press releases that served as the core basis, several added books and articles outside of the Lexis related to the industry, listed in footnotes, served as important references and further validation of narratives. Finally, as added validation and clarification of details, interviews were conducted with the senior teams of each of the platforms, apart from Microsoft (who declined to be interviewed and for whom we therefore relied on secondary materials and independent data acquisition). A total of 42 interviews were conducted via telephone, Internet, and in-person.

PSION AND SYMBIAN

Before proceeding with more detailed discussion, Fig. 1 summarizes key choices in the establishment and evolution of platform and platform owner boundaries in Psion's (and later Symbian's) system. The single most important choice of Psion was to implement "wide" platform and platform owner boundaries to allow Psion to apply its considerable skills in making cross-system-coordinated design and development choices (Hypothesis II). Also, Psion went on the integrate further into activities that enabled wider stimulation and orchestration of outside developers, notably in software applications, peripherals and network content and services (Hypothesis III). Psion's opened hardware development in the mid and late 1990s, enabling outsiders to build devices. However, it was not until Psion devolved equity stakes and made deeper commitments to hardware designers that the platform attracted significant investment and commitment from this group (Hypothesis I). Toward the end of the history recorded here, narrow boundaries and loss of control impaired future innovation and its ability to act as a central leader of the system into the 2000s (Hypotheses II and III).

Broad outlines of boundary choices described here are captured in Fig. 4 and associated changes in devices released on Psion's (and later, Symbian's) platforms are captured in Fig. 5.

Mid 1980s: Wide boundaries and innovating a new system: Working closely together – even side-by-side in a small 3,500 square foot London office in 1984 – Psion's small cadre of engineers and scientists, led by former academic Ph.D. Physicist David Potter, designed one of the earliest mobile computers: the Psion Organiser.[14] This first product was an 8-bit, calculator-like, 2k RAM device with a standard single-line LED display. The device ran on an 8-bit Hitachi microprocessor running at 0.9 MHz, a standard part in many of the 8-bit microcomputers running the popular CP/M operating system of the day. Psion's team had previously gained experience with this microprocessor and architecture in developing especially lean and performant software – games, a

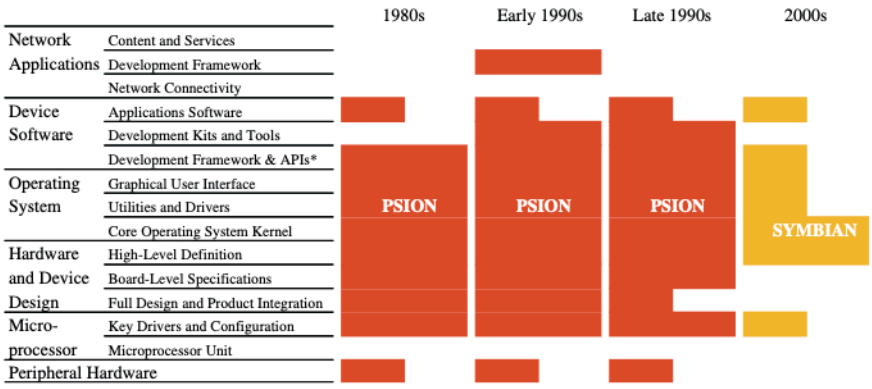

Fig. 4. Organization of the Psion and Symbian Systems. *Notes*: Whole-bar width indicates component supplied only by focal firm; half-bar indicates component supplied by focal firm, but not exclusively.

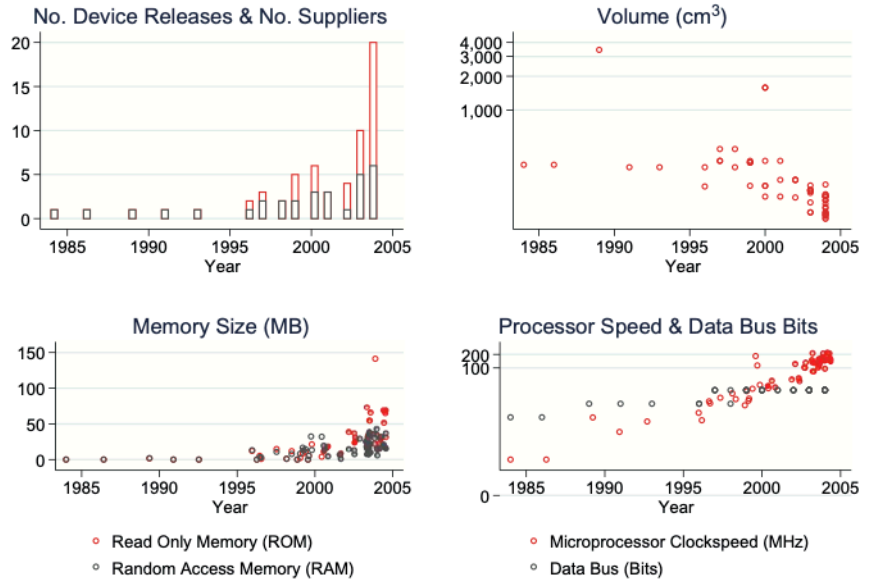

Fig. 5. Device Releases in the Psion and Symbian Systems. *Source*: Data are sourced from hand-collected data documented in Boudreau (2007). *Note*: Data points related to memory size and processor speed are "jittered" to avoid perfect overlap.

flight simulator, a business suite of applications – for Sinclair microcomputers.[15]

Despite modest specifications, the cigarette-pack-sized Organiser was expandable with modular memory cartridges,[16] and was designed with the ability to build and manage databases. If these things were not remarkable enough, the device could last six months on a single 9-volt battery. This success in design was able to be achieved through extraordinary parsimony in software design, with shared code across operating system and applications preinstalled in system memory,[17] and analog component drivers enmeshed in the operating system that were designed hand-in-glove with the hardware, itself. The ability to reuse code across applications, graphical user interface and operating system was a major tenet of Psion design throughout its history. Longtime Psion technologist and executive Colly Myers years later stated in a Psion press release: "Software re-use is a science for us, so the announcement of another two products utilizing EPOC further illustrates our platforms' enduring and stable architecture." Founder and Chairman, David Potter, described this "software and systems approach" as Psion's key advantage.[18] Extraordinary performance was thus wrung from modest component technologies with tightly interwoven decisions taken by an equally closely interacting development organization. Psion proceeded to sell a surprising 30,000 units with almost no marketing.[19]

The choices of platform boundaries – with most of the system available in just one standard variety – are most starkly consistent with solving the primary coordination problem (Hypothesis II) of the day, which was to overcome exceedingly challenging cross-system design challenges to eke out a viable and functioning product. While a key part of the platform – the microprocessor – was not developed by Psion, here Psion benefitted from using a standard part that was available from the market.

Continued improvements and independent experimentation (by users): In April 1986, Psion released a second version of its product, the Organiser II. With this product, Psion brought about quantum improvements to the design with a follow-up evolution of the product, using much the same design and organizational approach. The Organiser II was designed with many more times the original system resources (especially screen size and memory), and a considerably more elaborated operating system.[20] Apart from modular memory, this new device was able to attach peripheral hardware such as printers, scanners, and other devices. The larger screen and greater memory resources also opened the possibility of running spreadsheets, word processors and diaries, and a range of enterprise applications.

Even since the first version of the Organiser, the Psion had designed the Organiser to be programmable, with a built-in BASIC-like language that became known as Psion's Organiser Programming Language (OPL). Thus, apart from Psion's own software, users could develop their own programs to run on the device. With the Organiser II, however, independent development,

innovation, and experimentation by users went beyond just simple programs developed by individual users and hobbyists. The ability to at once connect mix-and-matchable peripheral hardware extensions at once, for example, led large British retailers such as Marks and Spencer to exploit the programming and databasing framework and hardware ports to develop a barcode scanning hardware for use in retail and logistics applications.[21]

The Organiser II grew to be more popular than the earlier version, quickly selling hundreds of thousands of units and won over both individuals in addition to large corporate buyers. The Organizer II and a series of incremental versions would sell half a million units over the following decade.[22] So robust and successful was the design it remained in production for over a decade, finally retiring in the late 1990s.[23]

Important to observe, Psion's "opening-up" system components such as software development and hardware peripherals were not a matter of retracting platform or platform owner boundaries, but rather integrating into the provision of enabling facilities (Hypothesis III). Considerable development effort and added system resources were required to allow outside components to access the operating system and wider system. While these efforts led to important developments by outside suppliers and users, Psion nonetheless remained the most important developer of mix-and-matchable components as it integrated into development alongside outsiders. The company possessed greater depth of skill and experience, and better knowledge of and ability to coordinate changes across its system (Hypothesis II).

Mid 1980s: Wide boundaries – and again innovating a new system: In the midst of growing success of it 8-bit Organiser, in 1987[24] Psion executives deliberately decided to avoid "incrementalism"[25] and set about working on a then-radical, quantum leap to 16-bit design.[26] To help fund this ambitious advance, the company held a public stock offering on the London Stock Exchange in 1988.

Psion executives' vision of a fully miniaturized palmtop bit device would take several iterations and several years to achieve. The first SIBO devices, the "MC" series (MC200, MC400, MC600), were released by the late 1989 and were essentially small sub-notebooks (considerably smaller than the laptops of the day) that would run for a week on six AA cells. The devices were impressive in their functions, even relative to desktop PC's at the time. The MC series already had a touchpad controller, added slots for proprietary solid state (SSD) memory, docking bays for peripheral hardware, and even had voice interface and voice compression technologies.

Platform and platform owner boundaries essentially mirrored those of the earlier Organiser project, with most components and elements of system design part of the platform and available in just the one variety, with just software and hardware peripherals as independent mix-and-matchable components. Efficient software design and component reuse remained the cornerstone of the Psion approach, allowing for these advances ahead of the wider PC industry.[27]

Despite astounding raw advances in system capabilities for the time, the MC series sold just 2,000 units.[28] The goal of implementing a revolution in its platform with the SIBO platform would nearly bankrupt the company. With little demand for this platform, there was also no real market for third-party components or broad-based user development.

Large advances through incremental improvements: It was only with the radical shrinking of SIBO devices to become "palmtops" with a new Series 3 in September 1991, that commercial success would finally arrive to the new generation system. These were pocketable palmtop devices – considerably smaller than the MC series and for a fraction of the price – with full QWERTY keyboards, and a graphical user interface drawdown menu interface (years before even Microsoft Windows would be available). These devices came pre-loaded with software application suites that were comparable to desktop functionality, including a fully functioning word processor and spreadsheet. The devices lasted for a month on two AA cells. Several improved SIBO models (3a, 3c, and several slight variants of these products) would be released by Psion in five years following, with updated industrial designs, more memory, software updates and new applications, longer battery life and infrared connectivity. Thus, Psion's ability to tightly coordinate altogether new systems (from scratch) with its expert software reuse and tight trade-offs finally led to a device that was clearly more performant in many dimensions relative to even personal computers, while being elegantly designed into a palmtop.

Thus, while many of the key inventive steps of SIBO occurred with the MC Series, it was the Series 3 that led Psion to rise to market leadership in mobile computing. Shipments grew an order of magnitude to hundreds of thousands of units, one-third of worldwide mobile computer shipments.[29] Psion entered the FTSE 100 of very largest publicly traded British companies. The creation of these palmtop devices would also begin a long string of design awards.[30]

Opening-up and attempts at orchestration external suppliers: Psion's system had been user-programmable from its first Organiser in the 1980s and continuing on with the SIBO platform, with the BASIC-like OPL and this basic programming framework built-in.[31] A basic developer kit, including documentation and desktop tools, had even been made available for the MC series. Although these had arrived late and still did not grant third-party programmers much freedom to exploit system capabilities.[32] Now in the early 1990s, noting cues from the growing PC industry, executives developed a growing interest in more actively pursuing a larger population of complementary software application developers in hopes it would boost overall demand for the Psion system. George Grey, former President of Psion USA explained: "This was the time when everyone was talking about compatibility and the lessons of the PC model versus Apple were becoming apparent. Really, If you were going to be successful you were going to have to openly license the platform. And, it struck a chord with Psion."[33]

A first step to accelerate development was to integrate further into the creation of development tools, first by increasing the power of the OPL in drawing on system capabilities, and granting access to key programming frameworks, such as the data database framework that underpinned Psion's own database and spreadsheet applications. The first official OPL Software Development Kit (SDK) giving many utilities and macros for nearly full access to the SIBO operating system services, was released shortly after the release of the first palmtop Series 3, in 1991. When Series 3a was released in 1993, OPL was upgraded again. The development framework would then remain unchanged until 1996 when OVAL (Object-based Visual Application Language) and associated development tools were created developed to be compatible with Microsoft Visual Basic. Development in more mainstream C++ would eventually come with a new generation EPOC32 32-bit platform, by 1997.[34] Notably, none of these steps to "open" to outsiders involved disintegration or devolving control; rather, they instead involved the opposite: integrating and investing further into facilities to grant outsiders structured and rule-based access to the platform (Hypothesis III).

Attempts to stimulate outside development also involved more than just technical steps. Support and encouragement included holding coding contests for cash prizes in attempts to numbers and quality of software titles from independent suppliers. Psion also arranged to co-market and facilitate distribution with a number of developers. These efforts led to dozens of new titles. The disclosure of database frameworks, in particular, led to a large proportion of applications developed with these initiatives to involve such things as in wine lists, travel schedules, baseball scores and other structured data-related applications. Here again the level of regulation and sponsorship was more forceful than in earlier generations of supporting applications development and other independent developers. Of note, apart from seeing integration into and investment in programming frameworks – there was outright restructuring of the institutional structures available to developers – including markets and rank-order contests (Hypothesis III). Unfortunate for Psion, the high costs and difficulty of distributing simple apps on physical cartridges[35] and still early maturity of development tools led to titles that were expensive, underwhelming in quality and little demanded.

In attempts to help resolve the limits of the media on which apps were held, in 1990s Psion worked closely with Intel and SanDisk on what would eventually become the industry-standard format of compact flash solid state memory. Thus, again attempt to promote third-party development meant a great deal more investment, definition, and integration into new enabling steps. In still further efforts to promote third-party software running on its devices, Psion's development team even began work to integrate a Java runtime engine (to run small Java-based programs) as it completed development of a next release of its platform in 1996 – well before standard Sun-sponsored standard Java configurations were even ready for commercial use.[36]

It will not be until some years later, once development tools and programming more powerful programming frameworks had become available, and online dial-up bulletin board services became more common and readily accessible means of distribution that thousands of titles would eventually become available (Fig. 3).[37] Amidst these efforts to promote third-party software, Psion nonetheless remained the most important applications software developer for its devices, continuing to exploit its own extraordinary strengths, knowledge and access to and reuse of low-level programming code to produce personal information management (PIM), word processor, database, and spreadsheet software that came preinstalled on the machines (consistent with Hypothesis II).

In 1994, amidst its growing success and its technological lead, Psion would once again repeat its prior patterns, setting about to advance the existing palmtop concepts, but with a multi-year project to build a new system around a 32-bit processor, with a multi-tasking, object-oriented operating system, and support for a pen interface. This new platform would be known as EPOC32.[38] In anticipation of this next leap, Psion's even attempted to acquire microprocessor designer Acorn – thus going still farther into its approach of integrating around all key platform components. Acorn was the designer of the then revolutionary new reduced instruction set computing (RISC) architecture, resulting in dramatic advances in power efficiency and cost. (This company would eventually provide designs for the Apple Newton and evolve into ARM, today's provider of standard architecture in most of today's smartphones.) The deal, however, fell through as the parties were not able to agree to a price.[39] Nonetheless, this interest of executives again goes to showing the importance of Psion's emphasis on internal integration and coordination (Hypothesis II).

While engaging in this new generation project, Psion also simultaneously expanded into networking and communications in attempts to bring more value to its system through direct networking and networked services and content for its devices. For example, in an alliance with Motorola in 1994, Psion produced a network messaging and wireless data terminal service called RWAN. The RWAN device combined (then advanced) packet-data radio communications technology with Psion device, along with newly developed e-mail gateways and host databases at the wireless carrier central office. Demand for the application never attained a sustainable scale. Also in networking, Psion developed 3Fax, a peripheral plug-in modem allowing palmtop owners to dial-up form a phone connection. This effort was coordinated with Compuserve, to enable content and services and e-mail by dialing-up over the public telephone network.[40] Psion also enabled networking to the PC desktop by developing its PsiWin synchronization software. Once again these examples illustrate a tendency toward internal control and integration (Hypothesis II), while also being consistent with Psion's expanding its platform boundaries not narrowing them to promote external development (Hypothesis III).

Late 1990s: Narrowing the platform, while keeping platform owner boundaries: As both the technology and competition in the marketplace advanced through

the mid-1990s, Psion decided it would begin publicly re-position itself as an "open standard." Apart from beginning to frame and describe itself as an open standard, company's announcements related to its new EPOC32 platform, beginning 1996, emphasized continuing attempts to promote third-party software development, the use of an industry-standard microprocessor architecture (ARM), and the integration and use of industry-standard communications standards (TCP/IP, Java, compact flash memory slot, etc.).

Psion executives were more ambivalent about how to organize hardware and device design. Whereas software had been mixed-and-matched and available to develop by outsiders from the beginning, hardware design and development had effectively been part of the platform – integral – up to this point. Whereas independent software applications development had been possible from Psion's beginning and had benefitted from an accumulation of tools and gateways developed over the years (and still had limited success) – hardware had always been integrated with and integral to the platform to this point. "Platforming" hardware would require a turn to separating hardware design with the remainder of the system, while also beginning to create at least some basic design framework for partners to follow. Executives worried too that licensing the platform would bring necessarily bring far fewer revenues per unit than would simply selling the devices. Device sales had been the company's primary source of income and there was an interest in preserving that income. The hope and key driver to the decisions, however, were that opening hardware and device development to independent suppliers might bring much-needed marketing know-how and resources of outside device suppliers. This was thought to be especially helpful in North America, where from the early 1990s, Psion had developed retail channels, but had never achieved significant sales. The release of the Palm Pilot in 1996 – then the most rapidly adopted consumer electronics product at the time and encroaching – provided greater impetus and eventually it was decided Psion would support multiple hardware developers.[41]

It was announced in 1996 that the new fourth generation 32-bit platform would be designed to allow external licensees to be built upon it, departing from earlier designs.[42] In close association, it was also announced that the company itself would be divided into a hardware design division and a platform development division. Executives believed that separating into separate divisions, after having created EPOC32, would better support the platform approach by creating fewer conflicts of interest between promoting Psion's own hardware business and supporting independent hardware developer licensees.[43] And thus, Psion's "opening-up" of hardware implied no change in firm boundaries; rather, the platform scope became narrower, with accompanying changes in design approach, internal organization and provision of support to independent hardware developers.

The opening to outside suppliers led to four platform licensees to join to develop new devices (apart from Psion's own Series 5 device). Philips and Ericsson each released EPOC products with networking and

telecommunications capabilities. Geofox released a tiny sub-notebook with the then novel touchpad. Oregon Scientific released a simpler and cheaper EPOC-based Palmtop with fewer applications, built at lower-cost in China. It retailed for about half the price of the Psion Series 5.

Apart from "platforming" hardware development, Psion executives also attempted to extend marketing and commercialization with a resale agreement signed with then leading maker of MP3 music players, Diamond. The deal involved fully-built Psion Series 5MX devices, re-branded as "Diamond Mako" to be marketed in North America through Diamond's established channels.[44] This was a purely commercial and marketing relationship in which Diamond bought fixed volumes of devices in exchange for exclusive rights to consumer retail distribution in North America. Unable to sell the product in agreed volumes, this relationship quickly ended in threats of counter-suits, and accusations that Diamond was attempting to dump devices at low prices in European markets.[45] Despite the separation of divisions, Psion had considerable interest in not seeing its core European markets erode or be underpriced.

Thus, opening-up external hardware development had little positive impact. Apart from the conflicts with Diamond, the cumulative sales of *all* of Psion's five device supplier partners (Geofox, Philips, Diamond, Ericsson, and Oregon Scientific) only amounted to 60,000 units – an order of magnitude smaller than Psion's own device sales. Projects by Philips and Ericsson also failed to produce compelling new generation of networked devices. Meanwhile, talks with other large hardware designers, Sharp and Hewlett Packard, failed to entice more licensing.[46]

While Psion's initial efforts to open-up hardware generated only tentative efforts and some reluctance to continue investing, it is difficult to point out any one single reason for this. Once the platform was in place, outsiders contended with immature technical design frameworks, rules, and support. At the same time, they competed with new products in a new market, while facing growing competition. Psion's own worries about its hardware partners' price mark-downs and growing share also suggest that the narrowing of the platform and internal reorganization into divisions also limited the company's incentives and resources to further sponsor partners' success. That the new platform and Psion's own hardware were themselves the most advanced and performant in Psion's history might, in large part, be explained by the company adhering to its usual highly integrated strategy in developing the new system (before opening-up to external licensing).

Late 1990s: Narrowing platform and fundamental reorganization – Devolving control to hardware makers: Despite the launch of the next-generation EPOC32 as an open platform, and amidst the limited inroads toward opening-up hardware, worldwide market share for all EPOC devices began to collapse by 1997 and 1998. A continuing loss of market share to Palm Computing and Microsoft Windows CE platforms led Psion, to offer more compelling incentives for external hardware developers to join the platform (Hypothesis I).

Apart from reacting to its lack of success with independent hardware developers to this point, company executives believed that smartphones would be the next opportunity for EPOC32 – a convergence between mobile handheld computers and mobile phones – and a new opportunity for licensing the platform. At the time, telecommunications standards bodies were planning incorporate wireless packet data protocols into upcoming generations of public cellular networks.

Psion chose to fundamentally re-organize in this round of attempting to open-up, beginning with affording greater independence to its software platform division as a separate company, transforming the platform division into an alliance with leading mobile phone makers, Nokia and Ericsson, and Motorola to form the joint venture, named Symbian, with each party including Psion, holding a minority stake. (Later, other companies would join.) By this time, narrowing platform scope and separating the operating system into a distinct business unit were not deemed sufficient (Hypothesis I); the sale of minority stakes for all parties was a deliberate measure intended to assure no one party could unilaterally overrule the others in how the platform would be managed and developed.[47] A requirement of European competition authorities in the original formation of Symbian – which licenses be made available on non-discriminatory terms[48] and there not be a concentration in ownership in Symbian[49] – may have, in fact, also helped in the establishment of this cooperation.

The concept and challenge of Symbian were to create a common platform and organization around which the powerful hardware developers and Psion could coordinate investments, while differentiating products from each other. A number of factors spurred the agreement. Apart from economizing on common investments and anticipating changes in public cellular networks' data capabilities, there was also a growing concern in the mobile phone industry that Microsoft might "commoditize" the smartphone business, as it had done with personal computer hardware.

The adoption of Symbian by leading hardware builders and associated capital infusions led the staff and development team working on EPOC32 to quickly double from 150 employees to over 400 by 2001.[50] With "a half-billion pounds and several years work"[51] the EPOC32 palmtop platform would be transformed from being an advanced mobile computing platform, still benefitting from Psion's cross-system design trade-offs to the Symbian smartphone platform, characterized by more modularity and flexibility of design.

Apart from narrowing the platform from hardware and reducing the concentration of ownership in the platform, a number of other steps were taken to reduce the power and scope of the platform, such as disintegrating the platform from network services or hardware peripherals development. Further, even the graphical user interface – a component typically considered integral to a platform – was made a mix-and-matchable choice for mobile developers to help support greater differentiation. It was also possible for phone makers to

specialize their own software application development frameworks, to allow divergent designs. Independent software development environments (the testing and development tools that are loaded with the libraries associated with a given platform) cold also be used by app developers. Less emphasis was given to applications development by Symbian itself, in hopes that outside developers would become more active.

Thus, apart shared ownership, platform boundaries were narrowed, allowing mobile phone makers to customize and differentiate more of the system. Rather than altogether unconstrained design, differentiation was attempted to be implemented without all coordination breaking down by creating a series of different sorts of reference designs, referred to as Crystal, Quartz, and Pearl. These were offered to support different classes of devices – from relatively basic mobile phones with limited features, to more general mobile computing phone platforms – that could be further customized by the phone maker.

The first-order result of this changed approach from Psion to Symbian was a quantum growth of new shipments (Fig. 1, right panel) to millions of units. This spike in volumes, however, was the result of a small fraction of phone-building partners' volumes being "cut over" to using the Symbian platform (instead of proprietary platforms). Most of this production volume was the simplest "feature phone" reference design.[52] These patterns are consistent with the decision to begin investing in Symbian once gaining ownership and control over the platform.

Despite doubling staff and having leading phone makers building on the platform and systematically cutting-over volumes to the platform, Symbian failed to deliver a single product that could be described as a "hit" in the coming years and decade. While there are many possible explanations, most of the key aspects of development that had previously been carried out by a single integrated firm (operating system kernel, graphical user interface, development frameworks, hardware specifications and board-level design, were now stretched across a multitude of phone-makers, graphical user interface developers, and development frameworks. Meanwhile, Symbian was no longer a platform "leader" but rather the junior partner to multiple hardware developers.[53] The narrow scope and devolved controlled foreclosed the ability to carry out integrated coordination of decisions (Hypothesis II). While, in principle, Symbian remained central in terms of its position within the network of suppliers, its position was weak (by design and intent), and its ability to orchestrate a coherent system was therefore compromised (Hypothesis III).

GRID, ZOOMER, AND PALM[54]

This section summarizes the establishment and evolution of boundaries for Palm Computing's systems, and closely related predecessors,[55] GRiD and

Zoomer. The patterns and inferences documented in Palm's history exhibit striking parallels to those in the earlier Psion case. Palm's technical and organizational strategies were also highly inter-related and in the same way of leading to a highly integrated organization producing a highly integral design, as in Fig. 2. Just as Psion, Palm benefitted from having a "wide" platform and even wider platform owner boundaries, to exact cross-component design decisions, in the early design and development of its system (Hypothesis II). From early periods, Psion also extended its platform boundaries to stimulate and coordinate activity beyond its own boundaries (Hypothesis III). In contrast to these clear benefits of increased coordination with wide platform and platform owner boundaries, later efforts to boost investments with narrower boundaries in the 2000s did not produce desired results (Hypothesis I). Earlier projects involving GRiD and Zoomer provide useful contrasting examples supporting similar points.

Broad outlines of boundary choices described here are captured in Fig. 6 and associated changes in devices released are captured in Fig. 7.

Late 1990s: Consolidated hardware and software control "on top" of the pc platform: In the 1980s, GRiD Computing was a maker of highly specialized computers, having produced for example a line of "rugged" portable computers for industrial and enterprise customers and field-use. The company had also produced some of the industry's earliest commercial portable computers and laptops. By the mid and late 1980s, its portable computers were mainly built with on top of the Microsoft-DOS operating system and Intel microprocessor platform. Thus, as with most every PC hardware development at the time, hardware design and development at the time could largely take place at arm's length from platform developers, on the basis of a mature development

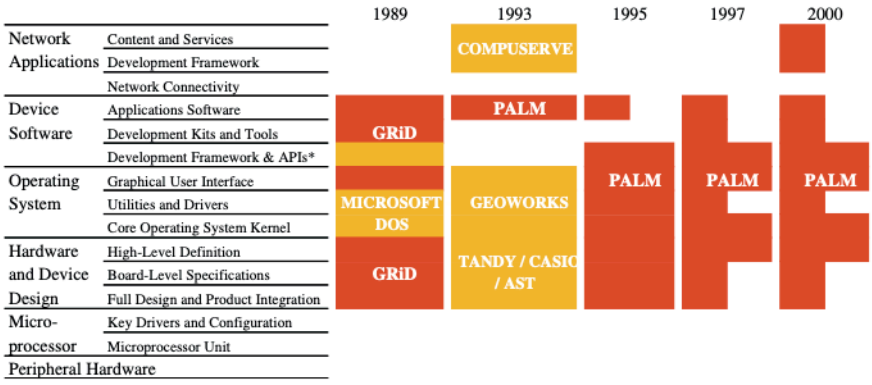

Fig. 6. Organization of the Palm System. *Notes*: Whole-bar width indicates component supplied only by focal firm; half-bar indicates component supplied by focal firm, but not exclusively.

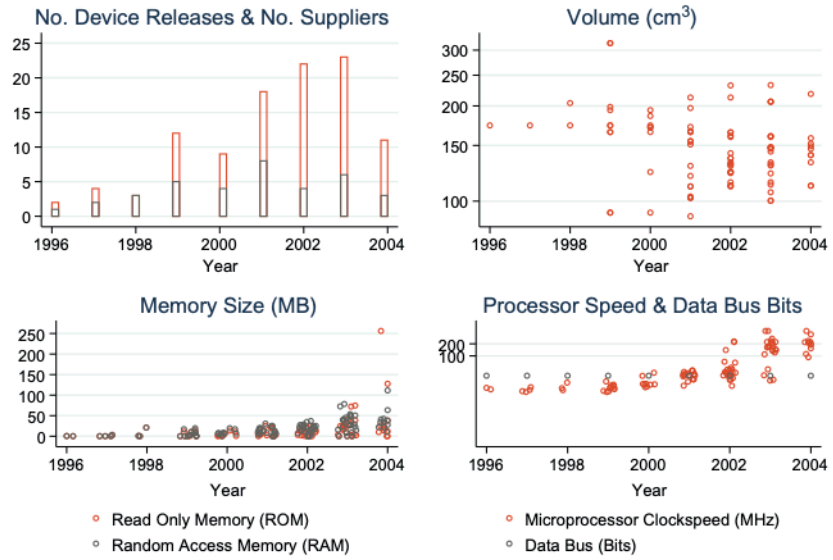

Fig. 7. Device Releases in the Palm System. *Source*: Data are sourced from hand-collected data documented in Boudreau (2007). *Note*: Data points related to memory size and processor speed are "jittered" to avoid perfect overlap.

framework. (See Gawer & Cusumano, 2002; Casadesus-Masanell & Yoffie, 2007, for discussions of the organization of the PC platform of the day.)

GRiD's foray into mobile handheld computers began with the Vice President of Research at the time, Jeff Hawkins – later to be Palm's CEO – conceiving of a handwriting recognition involving character-by-character recognition named PalmPrint. This eventually led to the company working on a touchscreen tablet device with pen interface in this early period. It again would use the same DOS/Intel-based architecture and organizational approach.[56] In this instance, the standard platform was used, but it was GRiD – the complementary hardware developers – that engaged in the integrated design of hardware and software to realize this challenging design at this early stage in the industry. It certainly helped that at the time the DOS platform was a still relatively lean platform, consuming limited memory and other system resources.

This process would lead the GRiDPad to be developed and introduced by 1989, as among the first successfully launched computers using a pen interface. Despite only being the company's first tablet computer, following the standard PC computer organization, GRiD was able to sell to many leading large corporate customers and the United States government, eventually selling tens of millions of dollars in total sales.[57] The device would be adopted for industrial uses such as those in the insurance and the oil industry for inventory management

and bookkeeping tasks.[58] The use of the system for repetitive inputting tasks also meant relatively straightforward use cases for the fledgling handwriting system to contend with.

Apart from providing context to Palm's later projects, the GRiDPad is suggestive of at least two points. First, it is notable in this case the integration to achieve coordination under a single firm's boundaries is achieved in his case by the complementor, rather than the platform. Whereas Palm and Psion, in earlier examples, integrated into hardware and all manner of software: this sort of cross-component integration was carried out by GRiD. This is consistent with the earlier review of integration for purposes of internal coordination, where it was hypothesized that existing theories of vertical integration to achieve internal coordination could more or less directly translate to contexts of platform-based organization. This then relates to a second point of what the difference is in this case from earlier Palm and Psion cases that could allow the complementor to play this role. The simplest explanation is that MS-DOS at the time was a relatively lean and mature and configurable operating system platform with established interfaces – and a well-established and even routinized ability to manage and organize external developers (Gawer & Cusumano, 2002).

Early 1990s: Wide platform but divided platform control – and failure: Jeff Hawkins would be at the center of another early project in mobile computing. During the period of the development and commercialization of the GRiDpad, Hawkins began to advocate that his PalmPrint handwriting recognition and the concept of mobile pen-computing should be brought to the mainstream, as a consumer product. This was a radical idea within the context of a GRiD, a company making premium specialized products for industrial customers. However, by 1988, GRiD was sold to a leading American consumer electronic maker, Tandy.[59] New owners and management agreed that a consumer project would go ahead. The system would be called Zoomer[60] (as in consumer or "con-zoomer"). Given the project would significantly depart from GRiD's usual projects, the initiative would be led by Hawkins from a newly created independent company, Palm Computing.

Palm's Zoomer project would be supported by an alliance of multiple industry-leading firms. Palm would develop PIM applications, handwriting recognition and PC-synchronization software to work with the handwriting recognition system. Intuit and Compuserve would contribute applications and content meant to appeal to consumers. Casio and Tandy would then collaborate to design the hardware. Rather than use MS-DOS, Palm would collaborate with Geoworks to adapt its GEOS 16-bit operating system to this pen-based mobile system. GEOS ran on existing PC industry-standard microprocessor and compatible hardware, just as MS-DOS, but offered a graphical user interface with drawdown menus and cursor control.

As a pocketable, powerful mobile computer directed to consumers, the Zoomer project began with analogies to Psion SIBO and Series 3 system at the

time – and even used a similar microprocessor (a 7-MHz processor with Intel 8088 core, where Psion at the time used an 8-MHz 8088 core processor). Despite these analogies and impressive backing of the Zoomer project, it would fail to produce a commercially viable design. The repurposing and attempted reduction in size of an existing operating system, while adding handwriting recognition and numerous applications – all within a first iteration of the product – slowed system performance. This was made worse by clumsy arm's-length coordination of technical development (with hardware, operating systems and applications designs continually attempting to adjust to one-another) and tedious six-way negotiations[61] among Casio, Tandy, GeoWorks, Palm, Tandy, Intuit and Compuserve. Slow progress, technical compromises and an expanding set of system requirements led to disputes over priorities, pricing, joint selling, and specifications. Tensions grew too from Tandy and Casio and also Geoworks and Palm anticipating the possibility they would eventually compete in hardware and software sales, respectively.[62] These issues ultimately led to a product characterized by "sluggishness, inconvenient size and weight, poor handwriting recognition software, and a steep price ($700)."[63] Palm managers also felt the larger hardware partners, Tandy and Casio, failed to invest sufficiently in marketing.[64] After Palm began work on improvements for a second version of Zoomer Casio and Tandy ceased support altogether for the project in 1994.

The Zoomer sold, 20,000 units immediately after its launch late in 1993, but sales quickly petered out.[65] Founder Jeff Hawkins later reflected: "What I learned through this process is that these computers are not little PCs. They cannot be designed by committee… We didn't know this when we built the Zoomer, we had a lot of problems in its design…."[66] The observation of especially severe coordination problems in the Zoomer project are emerging failing to internalize platform development under a single firm is most clearly consistent with Hypothesis II. Hawkins noted in this that "You have to go against conventional wisdom in many aspects." The prevailing wisdom of the time that forming an alliance of best-in-class partners would be optimal. However, a combination of unwieldy design goals and unwieldy organization were clearly less effective than were Psion's or GRiD's more focused and integrated approach at the time.

The pitfalls of having multiple partners work together on tightly intermingled technical and economic relationships in complex early system development did not appear to be isolated to the Zoomer project. The link between high integration and unified control and system development success is consistent in the early development of the history. Early market leaders and successful development projects – Grid, Psion, Palm, Sharp (Wizard), Hewlett Packard, Casio (Boss) – were each highly integrated projects. More disintegrated system development projects – such as the Zoomer and those associated with GEOS (beyond Zoomer), GO Corporation and Magic Cap – were stymied by diverging interests of partners and challenges of coordinating complex development activity.

Mid 1990s: Wide platform boundaries, consolidated control: As Zoomer floundered, Palm managed to stay solvent by marketing its PC-synchronization kits to some of the more successful mobile systems of the day, including Hewlett Packard's 200LX and Omnigo products and the Apple Newton. Early versions of its handwriting recognition software, Graffiti, were also made to work with Apple's Newton.[67]

Now entering into the mid-1990s, the industry backdrop consisted of numerous high profile failed systems and platforms (e.g., Zoomer, Newton, GO, BellSouth Simon, Magic Cap) and where industry volumes still only measured in the hundreds of thousands. Industry investment and product releases decelerated. In this context, Palm's executive team plunged into a new system development project. This time, the executive team proceeded with the firm conviction that developing a pen-based tablet for the consumer market would only be possible if the project were developed in a highly controlled and integrated fashion – the very opposite of what had taken place in the Zoomer project.[68] Thus, in the Spring of 1994, Palm decided to build an integrated system – on its own. This stress on integration and control also was somewhat consistent with Hawkin's experience with GRiD and directly analogous to Psion's organizational approach at the time.[69]

Apart from a strong view of organizational approach, Palm executives became equally emphatic about design criteria. The company would create a much smaller system than Zoomer. This would be a pocketable notepad-sized device with a pen interface. It would respond with negligible latency (delay), and without any boot-up sequence ("instant on"). Rather than a highly generalized computing platform, the device was to feature PIM programs (calendar, notes, contacts) with desktop programs that mirrored the handheld applications. Data would be synched on the user's personal computer. On the mobile device, these personal information programs would be virtually instantly accessible with the touch of a button built into the hardware. The device would also need to retail for under $300 – less than half the price of Zoomer.

To succeed with ambitious goals as this, and in an integrated fashion, Palm and its venture capital investors hoped they would gain a strategic investor to support with complementary skills and resources. But talks with Motorola, Compaq, and other potential strategic partners fell through because of a combination of differing product visions and desired specifications, concerns about Palm's concentrated control over the platform. Palm CEO Donna Dubinsky emphasized the company's position: "We can't do a deal if we lose control of the product. We've gone through this with Casio [in the context of the Zoomer project]."[70] Thus, the company proceeded with its design and organizational approach in an integrated fashion despite not having a single hardware engineer on its roughly two-dozen person staff (including nine engineers), as the time. Neither had the company previously developed an operating system.[71]

Again in a similar fashion to Psion, Palm chose to build its system around an off-the-shelf microprocessor[72] and to integrate this with other hardware and

software components, largely on the basis of tight cross-component design and parsimoniously designed software.[73] The process was one of optimizing, configuring and co-specializing components. A most obvious example of this was the creation of handwriting recognition software that was designed to be read on a specific area of the screen. Perhaps less obvious, the use of Palm's specific Graffitti pen interface reduced both size and power requirements[74] of the device. Trade-offs in memory, power and processor use also pervaded the design. Software applications were installed in ROM memory alongside and intermingled with components of the operating system. Given strict design criteria, the system was tested to excruciating detail and redesigned in trial-and-error for latency. Individual pixel counts were weighed for interface usability on a coarse monochrome display – all while determining where the active touch screen should be placed, along with button layout, and physical "industrial design."[75] Rather than respond to externally imposed specifications, as were typical with the Zoomer and even with the GRiDPad, compromises and trade-offs across design elements were part of the regular development process, down to choosing power source (batteries) on power and physical size. The company carried out this integrated innovation on its own, in a small teams of collocated hardware and software designers.[76] The operating system and preinstalled applications took up just a 320k footprint upon completion.

This approach of a tightly knit design and tightly knit, highly integrated organization paid-off. After launching its first product in 1996, the Pilot, sales rocketed past one million units in the 18 months, fomenting some of the fastest industry growth in the history of consumer electronics. Within a year of sales, Palm overtook Psion's sales share with roughly half of all mobile computer sales and would itself begin to receive awards for its novel design. This success and boost of the overall industry by a tiny integrated system developer was doubly notable given that dozens of much larger, established companies from around the world were failing in their own mobile computing projects.

Ongoing integrated control and substantive incremental advances: After the original Pilot release in 1996, a new operating system and utilities were introduced a year later to support networking, with sufficient memory to load a communications protocol (TCP/IP) stack, and to enabling synching over LAN networks. This new version coincided with the 1997 release of the Palm Professional, with a more streamlined form, greater memory and updated applications.[77] Roughly annual new incremental advances in platform, coinciding with new devices, would proceed in this manner over the next eight years, each using the same basis Palm OS and Motorola-based microprocessor architecture.

Although the outlines of the architecture were more or less set and unchanging in this period, the integrated control over the system allowed for important and ongoing incremental innovations over the platform and the wider system. For example, Palm achieved new levels of market acceptance with its Palm V released in 1999. This design radically slimmed the product's physical footprint,

while moving from plastic to an anodized aluminum case. In the same year, the Palm VII introduced wireless communications, with connectivity provided by an early always-on packet data protocol (Mobitex) over the public cellular network, allowing push e-mail and access to information services.[78] Palm's platform and its corporate boundaries also expanded as the company sought to support network services with its Palm.net network service. Web Clipping applications, also known as Palm Query Applications (PQAs) made use of the network to request and post web data. The devices also provided PQA developers with the user's position, in the form of a ZIP code, making the Palm VII the first web-enabled Location-Based Services mobile platform.

Mid 1990s: Opening-up software applications development: While building the original Palm Pilot system in a highly integrated fashion in 1995, Palm anticipated it would eventually allow independent developers to make software. This would by no means diminish the company's own role in software development, nor its highly integrated approach to designing software applications and having them preinstalled in system ROM memory. Therefore, in originally designing the system, the Palm team created an architecture that would be able to support easy loading of third-party software by synchronizing with applications first loaded to a personal computer. Further, the system was designed with a working library of application programming interfaces (rule-based means of calling upon the functions of the operating system), and "conduits" had been defined so that an third-party application developer could itself use Palm's PC-synchronization for its own desktop applications. Palm also built its platform to enable programming in multiple languages (C, C++, visual basic, Java).[79] The company also published documentation and examples as added support.

Executives had made these investments with the view that apart from PIM software, the creation of a wide range of software applications could further spur adoption of the platform. This, in turn, could draw greater entry and investments by software developers, and ultimately produce network effects around the platform.[80] To encourage their use, Palm offered reduced prices on the combined sale of devices and development kits to programmers. Further, a licensing regime was set up to also formally transfer tools and intellectual property, such as the source code, to the use of programmers. Palm also attempted to orchestrate added distribution and development "on top of" its basic programming framework, API library, conduit instructions and documentation – making these things available to independent development environment builders such as Metrowerks.[81]

The strategy was abundantly successful in rapidly leading to the creation of hundreds of applications and "hacks" (add-ons to improve the functioning of the basic system without changing the underlying system) in just the first year of operation, and this number would balloon to over, 20,000 applications by 2004 (Fig. 3). From this point, although Palm would continue to dominate other suppliers on the platform, the majority of products would be developed by developers other than Palm – with many milestones and breakthroughs

achieved by independents suppliers, such as Handspring's releasing the first color screen device and first smartphone and Sony's work on media players.

Therefore, opening-up the platform to software developers did not imply disintegrating from software development or narrowing either the platform or platform owner boundaries. Rather, if anything, this meant integrating and investing in a wider and more carefully defined platform to properly orchestrate and organize external development (Hypothesis III).

Late 1990s: Opening-up hardware: Palm would begin opening-up hardware to independent suppliers shortly after it did so with software applications. In doing so, it faced added concerns. For example, it did not have nearly the ready-made rule-based framework within which outsider developers could act at arm's length. Much like Psion in the earlier case, Palm was also concerned that liberally opening-up to outside hardware developers could potentially risk its own income, derived mostly from hardware device sales. Thus, in initially opening-up to outside hardware vendors in 1997 (Fig. 7), it did so solely on the basis of branded resale agreements. It did so with IBM and Franklin Covey (a leading supplier of paper-based PIM systems, and related courses). This provided access to corporate buyers and existing PIM markets in ways that Palm figured would only increase its sales without cannabalizing its own sales. Palm later inched farther into value-added resale agreements, including agreements with suppliers who would pre-install software applications software and, in cases, specialized snap-on peripheral hardware, such as arrangements with Crescenda (wireless connectivity), Epocrates (medical applications), and Supra (real estate applications). These sorts of deals again involved appealing to distinct market niches that Palm might not have otherwise addressed, while side-stepping the question of providing an analogous rule-based development framework, as was then provided for software developers. These arrangements also deviated from standard form contracts and downloadable licenses of software applications (Hillman & Rachlinski, 2002). The more particular concerns of hardware licensing led to ad hoc contracting and agreements.

When outside development by outside independent suppliers finally began in 1998, Palm remained interested in preserving its own mainstream market for its own devices while working only with specialized outsiders. Without simple rule-based contexts, the next wave of licensees included highly capable and experienced developers who could take the baseline operating system and existing hardware specifications to build devices. This also implied contracting for disclosure and access to Palm's proprietary specifications and contracting the ability to manipulate and alter the platform, as well. Palm would also support these external development teams with their own engineers, as needed. Symbol technologies, experienced maker of industrial grade computing devices, thus produced version of the Palm's device with functions such as barcode readers and networking capabilities. Qualcomm, leading telecommunications maker, developed a fully functional smartphone.

In 1998, executives also finally resolved to license more liberally to more mainstream device developers who might compete more directly with Palm's own devices (Fig. 7). This was part of an overall shift of toward a platform strategy focused on supporting a "Palm Economy." The key argument in favor was the belief that demand for personal mobile computing devices would accelerate rapidly through and beyond the late 1990s and it would be critical for Palm – as a platform – to capture a large share of that growth.[82] These agreements would take a similar form of signing bilateral agreements with capable and experienced device developers would develop new devices by gaining access to Palm's own platform and hardware designs, while contracting for rights to alter these things as part of their working designs.

Several of these relationships would be especially close and enabled basic features of the platform, itself, to be manipulated. For example, TRG needed to alter the system to accommodate the first implementation of compact flash card adapters in the TRG Pro product. Sony required numerous operating system changes to accommodate the growing multimedia functions of its devices. Because Sony's alterations of the operating system would be more profound, ad hoc contracts were drawn up to explicitly detail how these improvements would be eventually brought into the platform used by all licensees, where Sony then took at 6% share of Palm. Perhaps a closest hand-in-glove relationship of all was between Palm and licensees was Handspring. Handspring was formed when Hawkin's and other Palm executives chose to leave Palm in the face of mounting disagreements and power struggles with company owners. Just as Sony, Handspring, would license the Palm operating system and designs. Handspring's initial design added an expansion card slot, with the interest of enabling a wider library of hardware peripherals snap-onto its device (camera, phone, etc.). The company also used its experience and ability to build on and to manipulate the platform to itself create an early smartphone. (Palm would eventually acquire Handspring.)

Thus, from the late 1990s through the early new century, the majority of devices released on the Palm platform came from independent suppliers. Palm remained the leading supplier, while a number of important advances and experiments and access to niche markets came from partners. Thus, in many regards this program was successful. Much of the successful licensing approach was the result of Palm actively orchestrating the licensing terms and actively coordinating development. However, the choice to carry this out with *ad hoc* contracting and engineering support required mounting attention and investment by Palm's engineering team in a support role.[83]

2000: Disintegration of the platform owner from hardware devices: in 1999, shortly after the opening of hardware licensing, the company announced it would operationally separate its Platform group within Palm to work on the platform, while its group would continue to design and market hardware. Although Palm had to that point received growing interest in its platform from potential external partners, executives had the mindset that forming an intra-organizational "Chinese Wall" would allow the company to improve its

ability to be perceived as a credible and fair broker and supporter of the ecosystem, and thus to entire still more interest and investment by outsiders. This again was analogous to the history of Psion.

Two years later, in 2001, executives took this thinking further, formally spinning-off the Platform group as Palmsource, and increasing the independent of the company with an initial public offering of stock. With Palm's history of record-setting growth, and in these heady days before the 2001 technology stock market crash, the initial public offering market capitalization climb higher – at moments – than even that of General Motors or MacDonald's.

Therefore, here again we see a prevailing logic of a platform owner that narrowing its boundaries might more credibly signal to independent hardware developers that the platform owner would keep their interests in mind, with the hope this would promote outsiders' incentives to make platform-specific investments (Hypothesis I). Executives believed the firm could better support the "Palm Economy" more effectively and with fewer conflicts of interest (and especially in relation to other hardware suppliers) if it itself was were not in the hardware business.[84] Again, the logic was remarkably similar to that of Psion at the time and reflected the popular view at the time that "opening-up" and giving up control of standard technologies could promote their adoption.

The advancing of component technologies, such as industry-standard ARM microprocessors,[85] designed for low-power mobile use, facilitated further separation of control of design tasks.[86] In future years, the patterns of integration of Palm appear to have continued to reflect a tension between a logic of opening the platform to take advantage of contributions by outside suppliers (even porting the platform to Linux kernel, announced in 2005), while further integrating to areas of development in mobile and network services (through Palm's acquisition by mobile computer network applications supplier, Japan's Access Inc.).

These patterns and trends to outside licensees to develop hardware and to do so in a manner requiring ad hoc support and winnowed control would only continue into the new century. The approach would eventually lead 124 of the 163 devices released to year end 2004 to come from outside firms (Boudreau, 2007). Nonetheless, as smartphones became increasingly important and handheld mobile computing (without integrated phone and data) waned in popularity, Palm failed to keep pace with advances in the industry. The inability to create a compelling new smartphone, despite early experiments by its partners Qualcomm and Handspring, might be explained by any number of reasons.

MICROSOFT PALM FOR WINDOWS, WINPAD, PULSAR, AND WINDOWS CE

This section summarizes the establishment and evolution of boundaries for Microsoft's mobile computing systems. Again, here we see close links and

simultaneous decisions taken in organizational strategy and technical design approaches. The Microsoft is largely instructive in this regard a number of failed approaches. Where Microsoft eventually encountered some degree of success, by the late 1990s, this too came from wide boundaries and a wide platform, which finally enabled an ability to coordinate both within its own boundaries (Hypothesis II) and to orchestrate or coordinate beyond its own boundaries (Hypothesis III). Broad outlines of boundary choices described here are captured in Fig. 8 and associated changes in devices released are captured in Fig. 9.

Early 1990s: A mature platform, organized with narrow boundaries: Microsoft's commercial beginnings in mobile computing originated from unintended and unexpected experimentation by independent suppliers building on its Microsoft-DOS PC operating system platform – including such devices as GRiD's GridPad. In effect, this made Microsoft the second mobile computing platform only to Psion, by unit sales in the early 1990s. Devices built on MS-DOS (and Intel's ×86 microprocessors) rapidly multiplied in the late 1980s and early 1990s, produced by a wide range of developers: Atari, Poqet, Bicom, Compaq, Data General, Fujitsu, GRiD, and others.

For example, Hewlett Packard's 95LX (1991) was among the most popular palmtop keyboard-based devices at the time and included DOS-compatible desktop software embedded in ROM while retaining significant battery life, and featured small-scale keyboards, built-in programming language, and several scaled-down desktop programs. This was a time when Microsoft's operating system platform still occupied a relatively small memory and system resource-use footprint, while working with standard parts, and having abundant available compatible applications software. These DOS systems also

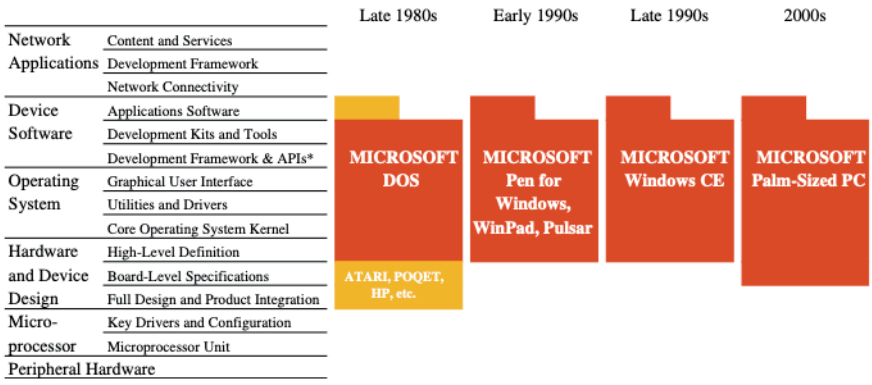

Fig. 8. Organization of Microsoft's Mobile Computing Systems. *Notes*: Whole-bar width indicates component supplied only by focal firm; half-bar indicates component supplied by focal firm, but not exclusively.

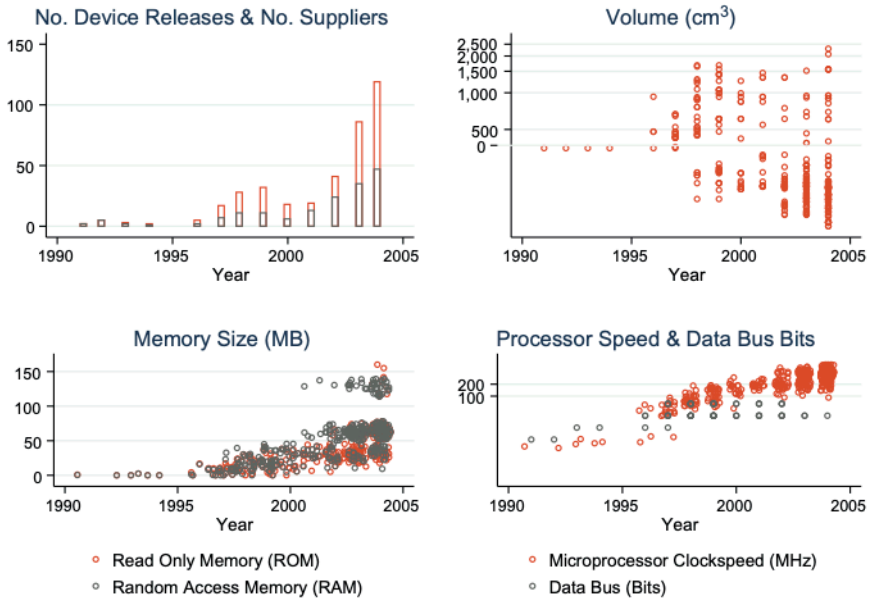

Fig. 9. Device Releases in the Microsoft Systems. *Source*: Data are sourced from hand-collected data documented in Boudreau (2007). *Notes*: Data points related to memory size and processor speed are "jittered" to avoid perfect overlap.

tended not to become burdened with pen interface or wireless capabilities, which may have further burdened the off-the-shelf DOS/×86 chip architecture. MS-DOS mobile devices to continue to be released through the decade to follow – even after EPOC, Palm OS and other Microsoft mobile platforms become available.

Remarkably, there were dozens of DOS-based devices that were developed on the MS-DOS system from the late 1980s through to the mid-1990s. This is especially notable given the approach to organizing the MS-DOS platform, and Microsoft's boundaries were a good deal different from those seen in cases of Psion or Palm in early years, where the platform owner in both cases defined the platform to include the hardware itself and also integrated into software applications and other areas. The MS-DOS platform, essentially that of the personal computing business, essentially included an operating system and a software application development framework; the platform was compatible with Intel's industry-standard microprocessors. Therefore, in these cases, it was, in fact, the device designers who took it up themselves to modify the software and hardware together to make working systems. Therefore, while there was an internalization of coordination problems across software and hardware (Hypothesis II), this was not carried out by the platform or platform owner.

This might be explained by the maturity and ready accessibility of the MS-DOS platform through application programming interfaces, so it could be readily modified at arm's length – a very different situation from those of Psion and Palm's early design efforts. MS-DOS at the time also remained relatively light-weight regarding its consumption of memory and other system resources, while also providing clear and stable design requirements for hardware. Thus, the problem of organizing systems around the MS-DOS platform may have been completely different, as it was a completely different design problem.

Relative to the theories of boundary choices presented earlier in this chapter, it is clear from research outside of this study that the operating system organization internalized tremendous design problems and coordination problems (Gawer & Cusumano, 2002), while also establishing a clear basis of power and control on which it could regulate and lead activity externally (Hypothesis III) (Gawer & Cusumano, 2002). Consistent with earlier observations, related to Psion and Palm, platform boundaries were not so clearly chosen to manage incentives (Hypothesis I); given the immutably towering bargaining power of platform owners in this instance (Microsoft and Intel), the evidence points to instead implementing a series of management practices to attempt to credibly convey to that independent suppliers could make platform-specific investments without a threat of expropriation by the platform.

Early 1990s: Organizing as a thin platform "on top of" the PC Microsoft-Intel PC platform: Microsoft began its development of pen computing in 1988.[87] Much of this effort become focused onto mobile computing by the early 1990s, with growing interest and investment in that market. Around that period, for example, apart from its own success with MS-DOS, GO Corporation developed its own PenPoint operating system platform for Intel's ×86 microprocessor architecture, and had become one of the best-funded startups of all time.

The organizational approach and boundary choices and eventual outcomes, can only be understood in the context of technical design choices at the time. Microsoft executives' technical vision of what could be achieved with mobile computing efforts was in a number of ways, more grandiose and visionary that what either Psion or Palm had in mind. These other companies began with ideas of how to innovate successful, performant product systems – that could transition to acting as platforms. Microsoft's approach was platform-centric from the start.

The vision was to create not just a platform for mobile computing, but to do so using families of modular, scalable operating systems and common code-bases.[88] A modular and reconfigurable platform code-base could, in principle, accommodate different levels of functionality and thus be applied to problems ranging from handheld computers, mobile telephones, automobiles, office machines, household appliances, home entertainment devices, to embedded systems and networked micro-controllers. Following the success (and business and technical logic) in platforming the personal computer industry, the idea here

was to create enormous scale economies on a consolidated Microsoft platform for a wide range of specialized systems beyond just the personal computer.

The practical attempts at first steps toward this vision appeared in 1992 with "Windows for Pen Computing." Rather than a new-to-the-world code-base, this was essentially a set of pen extensions (software drivers), built to run on top of a pared-down version of the personal computer operating system – the then Windows 16-bit operating system kernel. This created a layers-upon-layers system, where the pen interface was specialized to mobile, the Windows layer provided a graphical user interface, and the Windows layer was itself build on the top of MS-DOS, as the core operating system controlling system resources.[89]

Given the relative success of Microsoft and its MS-DOS-based mobile systems when this effort was led by independent experimentation, under the technical and organizational approach of the personal computer platform, one might expect that deliberate efforts by Microsoft to advance mobile computing might have accelerated mobile computing. This was not the case. In the end, several independent hardware device developers built products on Pen for Windows – including GRiD, NCE, Compaq, and Samsung. None succeeded to build technically performant, much less commercially successful products. The ponderous code base overwhelmed any advantages the platform might have had over the simpler MS-DOS operating system. In principle the handwriting recognition was quite advanced, and even designed to recognize a user's specific handwriting. However, the final working performance was disappointing and executives even explicitly instructed developers to avoid designers that required intensive use of handwriting recognition by users.[90]

The organizational design approach and platform boundaries in the Pen for Windows system were similar – and if anything wider – than those of MS-DOS projects. Whereas the coordination of innovative changes – integration of hardware, pen-drivers, graphical user interface with underlying core operating system kernel had previously been carried out by hardware device designers (as with the GRiDPad), here this integration was carried out by Microsoft. On its own, we might infer this was simply a bad design. However, this counter example would appear to also suggest that the task of coordinating this sort of challenging innovation was not inherently a *platform* coordination task, but a coordination task that *could* and was carried out by hardware developers in the case of MS-DOS.

However, it is also useful to point out that the integration with hardware it is more telling to observe that as the platform shifted from a lean, mature DOS platform to dealing with more innovative changes that hardware developers proved themselves capable to integrate around pen-drivers, graphical user interface and necessary software drivers – as in the direct comparison of say GRiD's pen-based systems based on MS-DOS. This contrast again emphasizes the importance of tight coordination in firm boundaries. At the same time and at least given the maturity of the earlier MS-DOS platform, some of these coordination tasks were not inherent *platform* coordination tasks.

Failed attempts to collaborate with large independent developers: Seeing that Apple launched its Newton pen-based device in 1993 to much press coverage and fanfare,[91] Microsoft developed a special interest in developing a competing offering. The system to be developed was the "WinPad." This was to be a 16-bit mobile device (begun five years after the launch of Psion's SIBO devices). The devices intended to access and synchronize data with the PC desktop, while providing mobility of use and an ability to communicate wirelessly with other WinPads.

Recognizing the technical problems and lack of commercial success with earlier Pen with Windows systems, Microsoft formed an alliance with Intel and several OEM manufacturers to build a new device around the operating system. Therefore, rather than develop a platform and leave remaining activities at arm's length, as in the MS-DOS approach, the vision was to more actively attempt to work with other firms to develop a working product. Here, the approach was still unlike the development of say Psion or the Palm Pilot in a highly integrated manner. The approach here was closer to Zoomer, of multiple highly capable firms working within an alliance. Microsoft retained control over the architecture, applications, and software, although it acted in consultation with Intel (which contributed its 386 chip) and OEM partners for the hardware. Neither did continued development, nor attempts to coordinate across firm boundaries overcome the fundamental limitations in the technical design.

Proceeding with the same basic organizational approach and boundaries, Microsoft then sought to ameliorate the design by shifting to a leaner operating system, named Microsoft at Work. This project had been developed to produce a modular, reconfigurable embedded operating system, but for use for simpler operating tasks such as the control of office machines like copiers and fax machines.[92] Apart from making large component-level changes in the operating system, the microprocessor plan was to swap-out the standard 386 chip and to build a specialized microprocessor ("Polar") in a partnership between Intel and VLSI. This goal was to better meet the processing, power and price requirements of the mobile environment and the advancing power of personal computer components were increasingly mismatched to the design needs for mobile computing.

Anecdotal evidence from the project is consistent with lingering problems of effectively coordinating design (Hypothesis II), given the project became delayed by two years, system component-level performance requirements ballooned and estimated product costs more than doubled.[93] Given the design and organizational histories of Psion and Palm around the same time, with extraordinary lengths in cross-component coordinated design needing to be taken to achieve viable designs on the basis of component technologies then available, this outcome might not have been surprising. While the project became delayed, device manufacturers began to leave the project. As prospects of success appeared to dwindle, Intel became more focused on the booming PC market. The Intel and VLSI project to develop a new low-power microprocessor were shuttered by 1994.

A similar organizational and technical approach – and more failure: In the meantime, Microsoft also began design work on still another mobile computer project. The idea in this case was to build a mobile computing device to communicate with the paging system, called "Pulsar." Also essential to this concept was to move ahead to develop the system of on the basis of a next generation of the personal computing platform, designing on the basis of the 32-bit Windows 95 operating system, while also depending on the low-power Polar chip under development.[94]

In many ways, this was a similar organizational and technical approach – once again reusing an existing code as a family of reconfigurable platforms. Whereas the WinPad project had attempting to make progress by backing away from personal computing component technologies to "lighter weight" and lower capability components to attempt to adhere to design requirements for mobile, this project was initiated to carry out a similar concept, but moving in the very opposite direction, with "heavier duty" component technologies, increasingly specialized and advanced for desktop computing.

This project immediately encountered problems in adapting the desktop operating system kernel or its object-oriented programming approach to the smaller-scale Pulsar system.[95] The development team also struggled to pare-down the system while retaining the integrity of the programming interfaces (APIs) that would allow Pulsar would remain familiar to programmers with experience in desktop computing.[96] Exacerbating these challenges, Pulsar's operating system was also be used in interactive television set-top boxes, in keeping with the concept of a reusable family of operating system platforms. Progress in the project slowed as the goals of the mobile computer and set-top boxes diverged,[97] and the project became much delayed.

Thus a series of attempts to deliberately develop a mobile platform and working system post-DOS each pursued a similar technical design approach while more or less adhering to the MS-DOS organizational approach and boundaries, but each failed. Attempting to make quantum innovation steps requiring nuanced trade-offs proved to be not possible when organizing a usual (narrow) operating system platform and repurposing components from other uses. Thus both technology design and organizational design appear to have conspired against these years of effort and investments.

Mid 1990s: Reduced modularity, reduced compatibility, greater control, greater integration – to create a working design: in 1995 – now 6 years after the original pursuit of a mobile computing platform, executives decided to bring personnel from both the WinPad and Pulsar projects together, to pursue the low-power 32-bit RISC architecture of Pulsar. This "Project Pegasus" would be the beginnings of the Windows CE platform and finally a meaningful entry into handheld computing.

In one sense – internal to the platform, itself – the design remained highly modular and compatible. The design approach would be consistent with Microsoft's longtime vision of a highly modular object-oriented operating

system to serve a fragmented set of opportunities in embedded computing such as industrial controllers, set-top boxes, video game players, on-board automobile controllers, and other possible custom applications – anything with containing a processor, memory, and internal clock.[98]

In another sense – external to the platform – the design became fundamentally less modular and compatible. The key and central difference in the design of Windows CE is that development was no longer constrained to reuse of or compatibility with the Windows personal computing operating system code-base. This was ultimately the design decision taken in hopes of simultaneously accommodating severe system constraints (memory use, in particular[99]), supporting multi-tasking, supporting the familiar Windows programming libraries (APIs), while developing a code-base that could be generalized to a range of applications. The resulting code base was indeed exceptionally modular – even more so than Windows – and lived up to the goal of being configurable to a wide range of applications. In this design, not only did the platform owner's boundaries not include the microprocessor, neither did the platform itself: the flexibility and reconfigurability of the operating system design anticipated the platform being used with multiple possible microprocessors.

Juxtaposed with the flexible and internally reconfigurable technical design was tighter consolidated control by Microsoft. The internal design was not a coordinated effort with multiple alliance partners. This was particularly so in the use of Windows CE for the creation of mobile computers. Hardware maker Casio was initially involved on a consultative basis,[100] and only well after a year into the design and after beta releasing the project were other OEM developers signed up. These were Compaq, Hewlett Packard, LG Electronics, Hitachi, NEC, and Philips (Fig. 9).

The Microsoft approach this time was also a good deal more controlled and integrated into hardware design, in the sense that in developing the specifications for the mobile computing or "handheld PC" (HHPC) version of the operating system, the Pegasus team simultaneously produced a hardware reference design that became the strict specification to which OEM developers needed to comply to be granted rights to use the operating system. This reference design specification stipulated power sources (two AA batteries), physical layout (physical size, screen layout size and grayscale specification, and precise keyboard keys that would be allowed), memory, ports (infrared transceiver and serial port PC card slot), audio speaker, and the microprocessors on which the design could run (Hitachi SH3 architecture, or NEC MIPS 3000 or MIPS 4000 architecture). This reference design featured a touchscreen and pen; but handwriting recognition was taken out of the specification. The Pegasus desktop developed for the HHPC bore a striking resemblance to the Windows 95 and Windows NT 4.0 desktop, including the available applications (word processor, spreadsheet, and browser, along with PIM software) despite being created with a different underlying code-base. Apart from significantly degraded performance that deviation from these prescriptions entailed, the Pegasus team

decided it simply was not feasible for engineers to test and validate each and every possible system configuration that a more flexible approach to hardware and software development.[101]

Thus, despite what might seem at first superficially appear to be narrow platform boundaries and considerable flexibility and modularity – at least from a technical perspective; the most salient features of the HHPC specification of the platform clarify the centralized control and integration of boundaries into key control points, for Microsoft to exert coordination and control both within its own boundaries (Hypothesis II) and beyond its own boundaries (Hypothesis III), in hardware development, application software development and the use of microprocessors. Thus, apart from simply executing a minimally viable technical design for the first time since MS-DOS, Microsoft re-established an ability to coordinate both within its boundaries and outside its boundaries for the first time since MS-DOS.

The Handheld PC 1.0 platform would commercially launch in the autumn of 1996, shortly after the commercial launch of the Palm Pilot. By design, the OEM manufacturers releasing quite similar devices – palmtop keyboard-based devices, akin to those of Psion's Series 5 devices, but slightly larger. With considerable technical progress in relation to Winpad, Pulsar, and a strict and optimized HHPC design to attain maximal performance from the system, several hundred thousand units managed to be solve in the first year of commercialization, placing Microsoft's HHPC platform alongside Palm and Psion in the top three mobile computing platforms in the next year. Nonetheless, critics in 1996 gave stinging reviews for power consumption, slow responsiveness and complexity of the PC-like interface in a mobile device. Reviewers might have been particularly critical of these features of the design, as the Palm Pilot, released the same year, had dedicated most of its design trade-offs toward serving these very criteria of power consumption, latency, and simplicity. Customer interest quickly waned. The disappointment in technical performance, relative to the other market leading platforms, led unit demand in 1997 to fall even below 1996 levels.

Stable organization and orchestrating more variation: In this fashion, subsequent releases of the operating system came with a slowly expanding set of pre-configured platforms. (The second platform release in 1998 came in five configurations, such as Auto-PC, an embedded system for automobiles, and Palm-sized PC, a penpad PDA, similar to the Palm Pilot.). This configuration of several platforms was analogous to Symbian's creation of multiple platforms to achieve greater product variety.

And this remained the case with the second-generation platform released in 1998 with even more microprocessors supported and many more device builders creating new mobile computers – with slightly more variation in products.[102] Even though these products now included those in a penpad form factor, similar to Palm's successful products, the products were judged to suffer from the very same fundamental problems of the first generation: slow

performance, poor battery life. The device was also criticized for continuing to appear too much like the desktop PC:[103] poor interface and usability. The handwriting recognition component that had been added to the system also generally disappointed. Mobile computers based on Windows CE remained far behind Palm in the market, capturing roughly a 10th of all sales while Palm captured more than half in the late 1990s.

Early 2000s: Further reductions in external compatibility, greater standardization, greater integration and control — and greater success: With its third version of Windows CE, Microsoft adjusted its design approach.[104] This included a ground-up redesign of the user interface as the requirement of conforming to the PC desktop was dropped.[105] The interface was rebuilt to more closely emulate features of market-leader Palm, removing a number of features to the Windows desktop. The platform was no longer ported to a variety of architectures, and ported to just one de facto standard architecture (ARM) supplied by multiple vendors. Perhaps most crucially, the reach of both hardware and software specification was further extended and tightly defined so as to further optimize the Windows CE 3.0 code to what Microsoft expected to be the commonly encountered uses of Pocket PC products and to better assure compatibility with applications software, development environments and middleware.[106,107] Additionally, a greater number of core applications software titles came preinstalled on the devices, such as the media player and e-mail client and utilities and drivers for supporting numerous components from cameras to networking became increasingly integrated into the platform. These changes would lead to less flexibility in design[108] and less variation in products and even fewer handheld computer releases as large computer makers (such as Hewlett Packard and Toshiba) began to dominate the hardware business on the basis of brand, distribution and economies of scale and small firms from the late 1990s (such as Everex and Novatel) were less able to successfully compete with less differentiation possible.

While it remained possible for small firms to experiment with smartphone models for some time, adapting the Pocket PC to these applications (such as Cesscomm and Cyberbank), eventually Microsoft would enter into the provision of standard hardware reference designs in this area and similarly eliminate the scope for small-scale experimentation. Cases where new firms were able to enter tended to be isolated to low-cost Asian manufacturers with an ability to economically manufacturer models based on the reference designs. With this approach, the quality of Microsoft products rose markedly and Microsoft devices would steadily expand its market share with radically improved product designs and would come to parity with Palm devices by 2004. Despite the reduced number and variety of both hardware makers and chip architectures resulting from its increased integration into core software and hardware design, the systems remained open to third-party software developers (the reduction of variation in hardware and microprocessors, along with deep integration into development tools, in fact facilitated software development[109]). Thus, the

tension benefits of the integration with opening to stimulate innovation of complementary components was negotiated by integrating further into some components while keeping others open.[110]

MONTAVISTA MOBILINUX

This section summarizes the establishment and evolution of boundaries and organization for MontaVista's Mobilinux platform and surrounding technical system. Montavista released about two-thirds of all mobile computing smartphones in this window of interest in this study up to 2004.[111] However, this was still a small minority of all product leases in the industry (Fig. 2). This platform shared with Microsoft's platforms a priority in offering a highly modular platform. This case is also useful in illustrating effects of having a platform owner with especially narrow boundaries, even narrower than the platform, itself. The narrow scope observed here somewhat overlaps with that observed in Symbian, but is much narrower still, and illustrate challenges of maintaining coordination either within the platform or across the wider system (Hypotheses II and III). Broad outlines of boundary choices described here are captured in Fig. 10 and associated changes in devices released are captured in Fig. 11.

Late 1990s: Serving as a small component in others' embedded platforms: In contrast to traditional leading platforms in mobile computing, the scope of MontaVista's "Hard Hat" Linux operating system in launch in 1999 was exceedingly narrow. The company was focused on miniaturizing and adapting

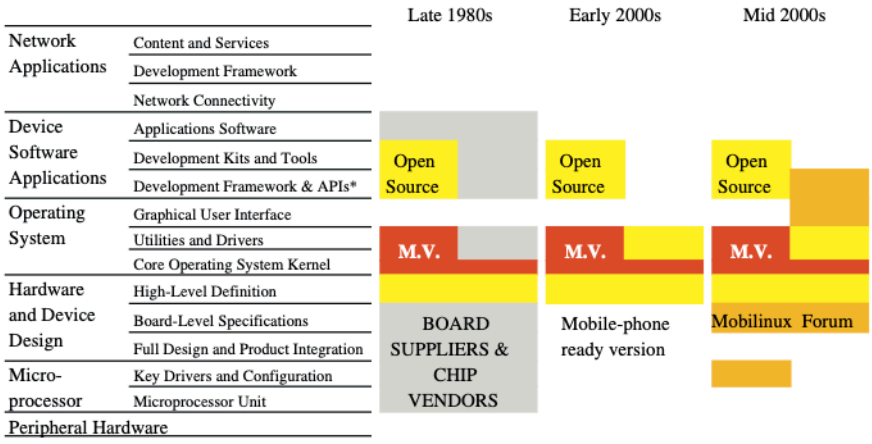

Fig. 10. Organization of the Montavista Linux System. *Notes*: Whole-bar width indicates component supplied only by supplier; half-bar indicates component supplied by supplier, but not exclusively.

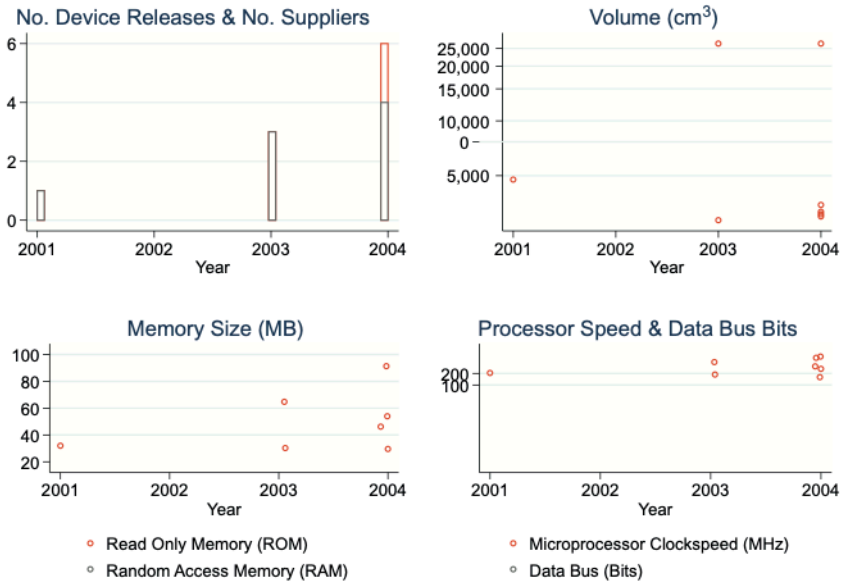

Fig. 11. Device Releases in Montavista's Mobilinux System. *Source*: Data are sourced from hand-collected data documented in Boudreau (2007). *Notes*. Data points related to memory size and processor speed are "jittered" to avoid perfect overlap.

the Linux kernel to work with embedded computer systems, including industrial machine controllers and sensors, automobile electronic systems, consumer electronics and communications systems (applications that Microsoft had addressed as part of its larger Windows CE strategy with modular code base). Linux was not as light, agile, robust, and amenable to embedded applications as traditional embedded operating system, such as VxWorks, itron, or a great many proprietary operating systems that have traditionally come pre-loaded on micro-controllers. But as these systems increasingly required greater computation and communications capabilities to interoperate with other systems and segments in consumer electronics, media players and mobile computers expanded, there was an opportunity for a low-cost, scalable and powerful operating system with richer utilities. Further, the proliferation of proprietary systems in the industry left potential economies of standardization unexploited. The modularity, expandability, unrestricted access to standard source code, standard APIs and lack of royalties of Linux positioned the operating system to serve this market. MontaVista founder Jim Ready, a 25-year veteran of the embedded computer industry,[112] sought to exploit this opportunity in forming MontaVista.

In its initial development, MontaVista's development objective was to transform the Linux operating system to meet the requirements of "embedded" computing, a distinct class of typically small-scale computing systems requiring critical control loops and mission-critical real-time responses. This implied transforming the existing Linux kernel, which required roughly 150 MB of disk space, and tailoring it to be able to run with resources as meager as a simple input-output board with limited on-chip memory (and be able to boot without access to keyboard, monitor, or disk access) – while bringing quantum leaps in latency (response time) and power management. MontaVista's team, including both Linux and traditional embedded computing designers, went through the Linux code and removed components that were not relevant or strictly necessary, while attempting to preserve as many of the essential features of Linux as possible so as to benefit from compatibility, and continuing development related to wider development related to Linux.

The initial product, launched in 1999, "Hardhat Linux" was 11.8 MB,[113] barely larger than MS-DOS and could even fit on a single floppy disk. The kernel was originally built as a layer "on top" of an underlying traditional real-time kernel so as to meet low latency and interrupt management requirements. Although this solution did not offer the same level a latency as in simpler, proprietary systems, The solution provided a 30-fold improvement in responsiveness on the standard Linux operating system.[114] Shortly thereafter, in 2000, MontaVista would be the first firm to do away with the dual-kernel approach, preserving the Linux programming model and leading to an additional 3-fold improvement in operating system response time.[115] "The charm here is that it's standard Linux. It doesn't change the APIs. It's not a separate kernel. We're following our company's philosophy, which is to improve Linux," Jim Ready was quoted as saying.[116] In following years, the kernel and supporting tools would continue to be improved. The system itself was improved with decreasing size, lower latency, and more functions. The basic code base would eventually be expanded to several broad classes of applications – industrial embedded systems, consumer electronics products, mobile phones, and carrier grade applications for telecommunications products – with a couple dozen different configurations to begin design with by 2006.[117]

In the case of highly customized and varied embedded systems development, narrow scope was made possible by selling to embedded systems "board developers" (such as Force Computers, Ziatech, WinSystems, and SBS Technologies) who themselves created basic hardware reference designs, including very basic board components with microprocessors integrated. In this sense, Hard Hat Linux was a component within a larger platform, taking the place of traditionally simpler operating systems that had come bundled in these board-platforms. In this, the extraordinary configurability and modularity of the Linux (MacCormack et al., 2006) system made it economical to port the operating system to many different hardware platforms.

Despite the numerous platforms of which Hard Hat Linux was a part, the tailoring of Linux tools by MontaVista and tools available for native coding on the processors created numerous options for embedded systems developers.[118] In a few years, this approach of providing a modular, configurable, high-performance embedded system would bring the company hundreds of large customers, using 32 different microprocessor manufacturers by 2006[119] (many more than even Microsoft).

The applications of the highly configurable operating system would expand to set-top boxes for televisions, toy robots and even carrier grade telecommunications applications. In addition to the continual augmentation of Hard Hat capabilities, the porting to new platforms and processors led to new releases on a roughly monthly basis.[120] Thus, despite the MontaVista's narrow scope, it was able to serve a highly disparate set of applications by maintaining a highly compact, modular code base (akin to Microsoft's original vision) and to then serve as a component within larger platforms where additional specialization to the application took place.

Cobbling together a smartphone platform with partners: While the strategy of supporting other firm's platforms led MontaVista to move toward supporting mobile computers through its support of other platforms[121], MontaVista took a special interest in developing a platform in which its Linux implementation would be the central and defining element[122] in support of mobile computers – and smartphones, in particular. CEO Jim Ready stated: "This 'platform' idea that could develop around smartphones could be the battle royale. From an overall business standpoint and semiconductor standpoint, you gotta win it. It's the kind of battle that tends to bring in a lot of resources."[123]

In this, the primary development activities of MontaVista remained the continuing specialization and innovation of the kernel,[124] itself, along with supporting utilities and tools. Montavista then simultaneously maneuvered to coordinate and entice outside suppliers to supply other components that would then be used as an integral platform by outside device developers. Given tight competition between numerous embedded Linux suppliers and tighter competition with incumbent mobile computer platform owners (as those discussed), this strategy would accommodate MontaVista's need to rapidly extend the scope of its platform with world-class components (which it, itself, with its deep but narrow expertise did not have the experience or resources[125] to develop) while avoiding a concentration of control over the platform which might discourage the platform's adoption. Thus, the development of external relationships with those firms able to provide the other essential elements of a platform to enable rapid development of smartphones became a key and defining element of MontaVista's strategy.[126] This included developers of essential applications software, graphical user interface, applications development environments, microprocessors, and hardware architecture reference designs.

Incrementally expanded platform owner scope: The earliest extensions of the MontaVista platform, in relation to readying the platform for mobile computer

use and smartphones in particular, related to applications development. These represented substantial extensions from an original product that did not even necessitate the presence of a display screen. As early as 2000, basic applications were ported to the MontaVista platform. For example, in 2000, the viewML browser (an open-source project) was ported to the MontaVista kernel by Century Software.

More importantly, entire applications development, run-time environments and graphical user interface systems were soon added to run "on top" of the kernel, thus serving as both components within the system and also as middleware that allowed software development to proceed somewhat independent of the specifics of Montavista's design. Java support in the form of runtime engines integrated in the operating system and in development tools and support came from Insignia (2000), IBM (2000), and Esmertec (2005). Graphics tools, graphical user interfaces (and associated application development environments) were also ported from Trolltech (2001) and Openwave (2004), which each also brought world-class user interfaces, development environments and libraries of existing software applications. MontaVista, itself, also supplied its own "Hardhat Graphics" environment (2001), offering graphics development across a range of applications. Thus, the strategy was somewhat analogous to Symbian's opening the graphical user interface – although perhaps MontaVista benefitted even more from the differential experience of outside suppliers.[127]

Especially critical to MontaVista's promoting outside suppliers' adoption of the platform was in enticing chipmakers to support the platform and integrate it within their increasingly integrated system-on-chip designs.[128,129] While in many regards, system design on the basis of Linux and modern component technologies had successfully become more modular, support from and compatibility with chipmakers had become particularly crucial – and more so in the development of mobile phones in relation to other embedded applications. In addition to tightly configuring board-level and system design to microprocessors, the microprocessors themselves now increasingly incorporate both central processing functions, radio signal processing and telecommunications utilities, as well as many component and input-output drivers, leading these products to increasingly be referred to as "system-on-chip" microprocessors. Further, by the early and mid-2000s these system-on-chips tended to increasingly accompany reference designs, or direction for building working smartphones which dramatically widened the range of firms able to successfully develop mobile computers. Embedded Linux from MontaVista was therefore made available from a range of leading microprocessor manufacturers, embedded in the chips themselves, for mobile computing applications, including Toshiba, Texas Instruments, Intel and IBM, and based on industry-standard (ARM) architecture. Montavista's early ability to persuade Intel to support Montavista's implementation of Linux and even take an equity stake in the company in 2000 may have been a critical turning point in creating this momentum.[130]

Creating an open coalition (without Power or Control): While diffuse control and power in the MontaVista system may have been necessary to enlist outside firm's support and to promote adoption and to bring together world-class components, the thin scope of control of MontaVista and co-specializing investments ran counter to the lessons of Psion, Palm and later Microsoft who chose to integrate precisely so as to coordinate across multiple such interdependent components.[131] Higher modularity of design and growing component power by this time may have mitigated the severity of design interactions. And yet, MontaVista has been able to take an early lead among Linux developers and present a plausible threat to market leader.

This may be in part explained by the extraordinary modularity of the system, which reduced interactions among designs and the scope for coordination problems. The multi-sourcing of component suppliers (and investments of chipmakers in Montavista) would have also likely reduced the ability and interest of partners to expropriate one-another. The need to avoid any sort of bad reputation in this respect while attempting to promote the adoption of the system would have also reduced the temptation to do so. In the case of the component in which there was not multi-sourcing – Montavista's operating system – the somewhat narrow intellectual property rights over the operating system (related to special functions within the kernel and in development tools) and the associated ability to substitute MontaVista's Linux with a different implementation reinforced this situation.

Nonetheless coordination problems were inherent in the project, as explained by Jim Ready, CEO of Montavista: "In all the other cases, the owner of the platform has complete control. The game that Linux changes so completely is in the control of the platform – the licensing, how the technology evolves, the whole bit – and *nobody controls it and everybody controls it*."[132] In this regard, MontaVista as developer of the operating system was able to act in a special, focal coordinating role to gain coordination of parties in the development of standards. Initially, this was in the form of bilateral negotiations with each of the trade partners mentioned earlier. However, as the set of partners grew, the "Mobilinux Open Framework"(named after the variant of Montavista's Linux specialized to mobile handset design) was formed in 2005 as an open institution intended to serve as a forum for developing and disseminating standards and handset reference architectures, in particular, using Montavista's operating system.

DISCUSSION

It immediately becomes clear in reviewing these cases that platform boundaries could not simply be taken as "given." Even before considering possible organizational and governance determinants, the case studies document considerable

divergence in executives' subjective technological design priorities. These differences were most notable in the initial architecting and founding of systems and were then more or less fixed and hard-to-change, except in isolated bet-the-company rebuilding of systems. Thus, not even principles of good design and product management could be taken for granted.

Most related to the thrust of this chapter, boundaries here were simultaneously determined by organizational and governance factors, alongside technological design factors. These factors were even closely intermingled and codetermined in the early founding of systems. For example, features of technological design, such as nature of component technologies used, degrees of modularity planned, ambitiousness of design goals, commitment to interoperability, and other features of design most clearly impacted the sorts of firms involved in development and how they would organize and coordinate. The role of organizational and governance determinants in shaping boundaries was clearer still in changes over time, as technological design decisions were somewhat difficult to change over time, once established.

Platform boundary choices and investment incentives: The main thrust of this chapter was to investigate whether and how boundary choices in platform-based organization related to several goals: (i) allocating incentives; (ii) internalizing coordination problems; and (iii) consolidating control over critical assets to gain the power to regulate independent agents.

In relation to allocating investment incentives, the analysis repeatedly found a lack of clear evidence that narrowing (platform or platform owner) boundaries would stimulate outside supplier's willingness and incentives to participate and invest in platforms – or, conversely, that widening boundaries would depress investments. For example, separation of the hardware business from the platform had no readily observable effect on other hardware suppliers, integrating to control a greater degree of hardware did not dissuade investment.

On the face of it, this (non-)result might initially appear to run counter to established theory (e.g., Grossman & Hart, 1986; Hart & Moore, 1990). Nonetheless, what is consistent with the established theory is the concern of platform owners that their asymmetric power over independent developers could dissuade investment. This reasoning is wholly consistent too with the threat of "profit-squeezing" by the platform owner (Farrell & Katz, 2000) or "lock-in" by the owner of a proprietary technology standard (Shapiro & Varian, 1998) and the dampening effect these can have on adoption and investment, as theorized in other parts of the literature.

I interpret the lack of effect of narrowing boundaries as related to the one-to-many nature of platform-based organization. Whereas in the established economic theory, an upstream and downstream firm in a bilateral monopoly can quite radically alter incentives by shifting monopoly control over the assets of production (and associated control and income rights) from one firm to the

other. (Likewise, transforming a proprietary standard into an open or public one will surely confer greater protections and assurances to independent suppliers using that standard.) By contrast, incrementally shifting boundaries of a platform bottleneck will nonetheless leave the platform as a bottleneck. Thus, the finding stands to reason and is consistent with Hypothesis I. Affirming this interpretation, and consistent with past results in the literature (e.g., Gawer & Henderson, 2007; Huang, et al., 2013; Perrons, 2009), incentives and willingness to participate were quite visibly increased only once added measures were taken to protect and clarify the rights and interests of outside suppliers.

Platform boundary choices and internal coordination: Among clearest ways in which organizational and governance determinants shaped both platform and platform owner boundaries related to an interest in directly integrating around most difficult coordination problems. This was also an area where the logic of technological design and that of organization and governance were most closely codetermined and intermingled; most important coordination problems were solved by devising cross-component design choices that intermingled system software, preinstalled applications and hardware design by the platform owner.

These decisions related both to platform and platform owner boundaries. Most challenging coordination problems were, for example, integrated under both platform and platform owner boundaries, where platform owner boundaries encapsulated platform scope. Conversely, cases of narrow platform owner boundaries, sometimes even narrower than the platform itself lead to clear coordination costs.

I interpret these patterns as consistent with the hypothesis that the logic of integrating to internalize coordination decisions from the established theory (e.g., Williamson, 1975) translates more or less directly to platform-based organization, as in Hypothesis II. It stands to reason that integrating within the single, reused components of the platform should yield greatest integrated coordination. However, consistent with integrating coordination problems being a general idea beyond just platform-based organization, the analysis documented other scenarios in which executives were motivated by a need to coordinate decisions (and did, in fact, achieve these ends). These included cases of platform owner integration into complementary components, such as hardware device design and applications software, and complementary component suppliers integrating into and modifying parts of the platform.

Platform boundary choices and external orchestration: Consistent with past research on platforms and systems organization, the primary motivation for executives opening-up certain components to outside suppliers was to harness diversity, specialized capabilities, and comparative advantages of independent suppliers (users, complementors, contributors) in supplying mix-and-matchable components (e.g., Baldwin & Clark, 2000; Boudreau, 2012; Boudreau et al., 2011; Chesbrough, 2003; Garud & Kumaraswamy, 1995; Katz & Shapiro, 1994; Langlois, 1992; Meyer & Lehnerd, 1997). However, opening-up platforms in these cases by means of narrowing the platform did not produce an

all-or-nothing tradeoff between "openness-versus-control," as might be suggested by research on say open technology standards (e.g., Shapiro & Varian, 1998; Simcoe, 2006; West, 2003). Rather, consistent with past research, platform bottlenecks retained the power to orchestrate independent suppliers through contracting, price-setting, and setting rules in technical frameworks.

Analogous to established research on firm organization, the empirical analysis finds that platform owners' power and ability to influence and coordinate or orchestrate activity within the system sub-economy is related to chosen platform boundaries in terms of control over critical assets, as in Hypothesis III. In a simplest example, while the language of "opening-up" – by the nature of the word choice itself – might naturally connote disintegration and narrowing platforms and opening in the sense of giving-up control, in fact opening-up was often associated with efforts to integrate farther into critical rule-making control points in the system. This included integrating into application programming interfaces, facilitating development frameworks and the creation of reference designs or simply the development of specifications to which outsiders were made to adhere through contracts. It was also plainly relevant that in those instances in which platform owners lost control over their platforms, this ability to orchestrate outsiders was markedly diminished.

Arguments in earlier hypothesis development suggested that it would be important to integrate into "critical" assets, in particular, to gain the ability orchestrate external actors. "Critical assets" are: (i) a basis for bottlenecking the system and exerting rights of exclusion; (ii) a facility to increase productivity of external trade partners; (iii) an embodiment and basis for rule-making; and (iv) a source of capturing value.

Such nuanced hypotheses are certainly not possible to unequivocally test here. However, there is abundant evidence that could be interpreted in this light. For example, failing to exert consolidated control over the platform was a basis for losing the ability to orchestrate, consistent with (i) and (iv). And, as in the examples above, there were many instances of integrating into rule-making components (iii). Both points (ii) and (iv) relate to maintaining exclusive control over a platform with high demand in the marketplace. Waning prospects and divided control both hindered orchestration in these cases. Also consistent with point (iv), disintegration from valuable hardware devices appears to have diminished platforms' abilities to lead their systems.[133]

CONCLUSIONS

The analysis here showed that platform and platform owner boundaries varied a great deal from system to system, over time and in relation to one another. Organizational and governance concerns played a first-order role in determining boundary choices in platform-based organization, interacting with technical

design and product management choices. Three established economic theories of firm boundaries were found to apply (somewhat differently) to the platform context.

Arguments and evidence are consistent with narrowing platform and platform owner boundaries doing little to boost independent suppliers' investment incentives while diminishing the ability of the platform owner to either coordinate activity inside the platform owner's boundaries or else to coordinate ("orchestrate") activity across the system beyond their boundaries. This did not create a situation of sharp and inevitable trade-offs between openness-versus-control and coordination. Rather, most successful open platforms were those that carefully chose boundaries to reconcile the interest of opening-up to harness contributions of outside suppliers with the interest of simultaneously maintaining coordination (orchestration, sponsorship, leadership, regulation) of activity in the ecosystems or sub-economies built around the platform.

Thus, the case studies and theoretical arguments featured here emphasize the possibility of implementing boundaries (and other management and organizational practices) to achieve both openness-and-control, rather than face openness-versus-control. Rather than a stark and discrete decision, this appears to involve nuanced simultaneous choices of technical and organizational design.

NOTES

1. The analysis here focuses on scope in relation to design activities and intellectual property (without regard to whether or not the designer chooses to manufacture components internally or through external contracting).

2. For a more detailed examination of established ideas in the economics literature on firm boundaries, see Gibbon's (2005) review of four theories of firm boundaries. Relative to his four theories, the treatment here groups two of Gibbons's theories ("rent seeking" and "adaptation") under the first subsection here of "solving coordination problems." His reference to "incentive systems" with references to agency theoretic perspectives and the regulation of a sub-economy is covered under the third subsection here regarding "orchestration" around a platform. The discussion here refers to Property Rights theory just as does Gibbons, in the first subsection.

3. Numerous variants of original idea of using firm boundaries and the allocation of residual rights of control to mediate investment incentives (and recently, an emphasis on the ability to efficiently adapt to unforeseen contingencies) have been elaborated since the original property rights view was developed. This chapter simply wishes to highlight the use of shifting boundaries to mediate investment incentives in broad brush strokes.

4. These concepts here are also related to those of Pfeffer and Salancik (2003) within organizational sociology literature.

5. This is a good deal different from the classical setup of firms setting boundaries in traditional upstream and downstream relationships, where relationship-specific investments made by upstream and downstream firms necessarily then also introduces a degree of mutual dependence. This is because once productivity-enhancing relationship-specific investments are made, neither supplier is necessarily easily substituted without then reducing productivity. By contrast, in a one-to-many production relationship – as in

platform-based organization, where multiple trade partners make platform-specific relationships – where many substitutible suppliers each make platform-specific investments, there is not the same fundamental transformation to bilateral monopoly power – the one-sided power continues to reside with the platform owner.

6. These sorts of *ex-post* hold-up strategies are, on the one hand, sometimes touted as deliberate profit-maximizing strategies (e.g., Shapiro & Varian, 1998) and there is empirical evidence of platform owners, in cases, systematically squeezing highest-value complementors (e.g., Zhu & Liu, 2014).

7. This is another regard in which platform organization may diverge from characterizations in the classical Property Rights theory. Property Rights theory emphasizes investments *per se* – without necessary regard for *which particular actor* delivers these investments. The actors in the theory may be somewhat substitutable in "capabilities" or production functions, and the focus in the theory is on the role of firm boundaries per se on incentives.

8. Neither does disintegrating remove the threat of competition in the market for complementary components that typically has large numbers of suppliers. This again is an important difference with the classical make-or-buy scenario studied in the Property Rights theory, where upstream and downstream suppliers are set within a bilateral monopoly relationship (once parties make relationship-specific investments).

9. An alternative channel through which opening-up a component might increase investments is when opening-up a component in a way that stimulates powerful network effects, sufficient to create a demand-pull for greater investment and innovation (e.g., Bresnahan & Greenstein, 1999).

10. Later developments in this literature came to criticize this seemingly one-sided view of the benefits of firms (Grossman & Hart, 1986), but for our purposes in this chapter, it will be useful to preserve this original interpretation of this literature.

11. A principal at the center of a large number of trade relations, as in workers, can also have advantages from say holding a reputation, possessing "focalness" will coordinating and leading others, and also serving as a central conduit for information flow.

12. A liberal interpretation of the original text.

13. NB. Typical principal–agent models do not in themselves directly suggest a theory of asset allocation and firm boundaries.

14. Langdon and Manners (2001).

15. Litchfield (1998).

16. Add-in software cartridges were created and sold separately, including those with mathematical and financial functions.

17. The ability to re-use code across applications, graphical user interface and operating system was a major tenet of Psion design throughout its history. Longtime Psion technologist and executive Colly Myers years later stated in a Psion press release: "Software re-use is a science for us, so the announcement of another two products utilizing EPOC further illustrates our platforms' enduring and stable architecture."

18. Potter (1998).

19. Langdon and Manners (2001).

20. The operating system itself began to take on much greater sophistication; David Potter referred to it as the "first real EPOC system" (Langdon & Manners, 2001).

21. Old-Computers.com. On-line computer museum. Downloaded December, 2004.

22. Litchfield (1998).

23. Tasker (2000).

24. Langdon and Manners (2001).

25. Langdon and Manners (2001).

26. In relation, graphical video game consoles that were typically connected to full-size televisions sets were still 8-bit machines at the time.

27. MC Word, the word processor, added only 13 kilobytes to memory requirements. Tasker (2000).
28. *The Economist* (1994).
29. *The Economist* quoted annual sales in 1995 as 350,000, representing 1/3 of the global market. Anonymous (1996). Forrester Research study from November 1995. September 6, 1996: "Psion Computers became the world leader in palmtop computers through the early 90s with products based on the EPOC architecture. The company now commands the largest world-wide market share with 32.7%."
30. Psion has won major awards throughout its history, including the high honor of winning several Queen's Awards for export achievement (1995) and Technological Achievement (1993) for the Series 3a and several British Design awards. Other awards included "Best Consumer Product" from the Design Business Association.
31. Users could program simple scripts that would be saved on the devices, themselves, using a simple Psion-developed BASIC-like language, called OPL.
32. Litchfield (1998).
33. Personal interview, 2005.
34. Edwards (1997).
35. While elegant and a harbinger of future technologies in the industry, a simple cartridge without software was priced at $100 in the early 1990s. Cartridges with pre-loaded software sometimes exceeded $200.
36. Tasker (2000).
37. Steve Litchfield asserts that eventually 1,500 third-party titles would be written for the more popular Series 3A, in the five years following its, 1993 release. Litchfield (1998).
38. Tasker (2000).
39. Anonymous (1996) and Langdon and Manners (2001).
40. Psion's modem division, Dacom, simultaneously became a leading vendor of PC card modems in this period.
41. Telephone interviews with George Grey, President of US Division of Psion in late 1990s (2005), Andrie Devries, Psion Manager in late, 1990s (2005). Martin Tasker also makes conspicuous mention of the competitive pressure from Palm in recounting the development of EPOC32. Tasker (2000).
42. Taylor (1996). Psion to license operating software. Financial Times.
43. Wilson (1996).
44. Smith (2000).
45. Smith (2001).
46. Price (1998).
47. This became especially clear when the possibility of Nokia acquiring Psion's shares and with it control of Symbian brought on controversy before an alternative approach involving the acquisition of shares by the minority shareholders was devised. Sunday Business (2003).
48. Suarez and Eisenmann (2004).
49. European Commission (1998).
50. Northam (2006).
51. In-person conversation with Peter Bancroft, Vice President of Market Communications at Symbian (2004).
52. Company reports.
53. Several interviewees complained about the limited ability of Symbian to steer and coordinate more powerful trade partners and owners.
54. The developer of the Palm System began as venture-funded Palm Computing, was then acquired by US Robotics, which was acquired by 3COM, then spun-out as an independent firm, Palm Inc., which then eventually separated to an independent platform owner, PalmSource, Inc. I will refer to the supplier of the Palm OS platform throughout as "Palm."

55. Palm began as venture-funded Palm Computing and was then acquired by US Robotics. The firm was then acquired by 3COM. This was followed by a spin-out to become an independent firm, Palm Inc. Palm Inc. was then separated from the hardware device business to become an independent platform owner, PalmSource, Inc. I will refer to the supplier of the Palm OS operating system platform throughout simply as "Palm."
56. Dillon (1998, p. 97).
57. Atkinson (2008).
58. Barnett, *Pen Magazine*.
59. Tandy was the house brand of leading North American consumer electronics retailer Radioshack.
60. Also marketed as the "AST GRiDPad 2390," Tandy "Z-PDA," Casio "Z-7000," and "Casio XL-7000."
61. Dodson and Hart (1998).
62. Butters and Pogue (2002) and Dodson and Hart (1998).
63. This quote and lucid description of these challenges appear in Chapters 3 and 4 in Butters and Pogue (2002).
64. Dodson and Hart (1998).
65. Butters and Pogue (2002). A smaller estimate of 10,000 units shipped between October, 1993 and January, 1994 comes from Barnett, *Pen Magazine*.
66. Barnett, *Pen Magazine*.
67. In this period, Palm also attempted to adapt its software for a mobile computer project by Sharp that would eventually be canceled.
68. Venture Capital backers of Palm also had direct experience of having invested in multiple mobile computer ventures to serve as a benchmark in this decision.
69. Kosnick, Atluru, and Wasserstein (1998).
70. Butters and Pogue (2002).
71. Butters and Pogue (2002).
72. The design was based in large part on a low-power Motorola 16-bit 68000 chip running at 16 MHz (slightly slower than contemporaneous processors in Psion devices). The processor came with ready-made integrated drivers for the screen and other components.
73. At a presentation at MIT in 2005, Jeff Hawkins stated: "There was nothing special about the pieces of the PalmPilot, it was how they were put together that was important."
74. Kosnick et al. (1998).
75. Butters and Pogue (2002) describe these aspects of Palm development, centered around founder Jeff Hawkins, in some detail.
76. Butters and Pogue (2002).
77. Winton (2001).
78. Mobitex was also used by the earliest Blackberry devices from RIM.
79. Rhodes and McKeehan (1999).
80. In-person interview Larry Berkin Senior Director of Palm (Software Developer Support) (2005).
81. Metrowerk's "Codewarrior for Palm" was available just one month after the release of the product, allowing developers to code Palm programs in C, on the Macintosh desktop. A year later, a PC-based IDE would be made available, but by that time, pioneering Palm developers had already developed their own tools and development kit and made them available on the Internet. Perhaps most important among these was the Alternative Software Development Kit (ASDK) by Darrin Massena, which bundled together the tools, headers, support files, samples and documentation necessary to develop Pilot applications on a PC running Windows 95 or NT.
82. In-person interview with David Nagel, former CEO of Palmsource (2005).
83. Numerous interviews with Palm staff.

84. In-person interview with David Nagel, former CEO of Palmsource (2005).
85. The reduced instruction set architecture and specialized designs of microprocessors designed with an ARM core allowed devices to advance to chip speeds beyond, 200 MHz from the maximum 68MHz available with Motorola's dragonball architecture.
86. Numerous industry analysts and managers, however, speculated that the separation of operating system and hardware development still came with design costs. PalmOne, the hardware manufacturer, considered re-purchasing the operating system supplier, PalmSource, in 2005. Wildstrom (2005).
87. Microsoft (1992).
88. Murray (1998).
89. Microsoft (1992).
90. Microsoft (1992).
91. Morrison (2001).
92. This was the second scalable, modular OS built by Microsoft. in 1991, "Modular Windows" was an earlier attempt to design for real-time embedded consumer electronics applications, such as interactive multimedia and home entertainment players, before being discontinued in 1994.
93. Computable, Byte Magazine, Aug., 1995
94. Fitzgerald (1994).
95. Anonymous (2006).
96. There were even tensions over which API set to use, the well-known Win32 library or those based on the new objected-oriented Windows 95 kernel. The development team successfully argued for the Win32 interfaces.
97. The television application was to be a simpler, closed system dedicated to television use and with high data transmission and networking requirements, whereas the Pulsar was to support open development by third-party software suppliers. Murray (1998).
98. "I hope to have thousands of OEMs building Windows CE devices over the next few years." Microsoft manager, Frank Fite, quoted in Murray (1998). Inside Windows CE. Redmond, WA, Microsoft Press.
99. "We started with a pretty advanced object-based operating system, but when we started looking at what it took to build applications and system components, we realized it was an uphill battle. [W]e… started porting components from Win32. We ended up with probably 80–90% of Windows, but it was big and didn't' give us the flexibility we needed. Approach number three was to use the API model but not the code." Harel Kodesh of Microsoft quoted in Murray (1998). Inside Windows CE. Redmond, WA, Microsoft Press.
100. http://www.hpcfactor.com/support/windowsce/wce1.asp
101. A Brief History of Windows CE, 2005. HPCFactor.com website.
102. The most novel of these products were those that developed custom code base using the Microsoft's platform builder tools.
103. Morrison (2001).
104. "Although … Pocket PC is Windows CE 3.0 under the hood, the differences between Pocket PC and prior versions of Windows CE are dramatic. Windows CE was put through the wringer by many users and deservedly received a great deal of criticism… For these reasons and more, Microsoft set out to develop a completely new handheld operating system when it went back to the drawing board for Pocket PC." Morrison (2001).
105. Anonymous (2006).
106. Microsoft. Comparison of Windows CE. NET 4.2, Pocket PC, 2002, and Windows Mobile, 2003 Software for Pocket PCs.
107. Firms opting out of the standardized design could still use Microsoft's "Platform Builder" software to configure Windows CE software to their needs.

108. Psion Teklogix Inc., "Windows CE, Pocket PC And Software Development Considerations." Internal Report.
109. Microsoft (2004).
110. At the, 2005 Microsoft Mobile and Developer conference, Bill Gates reinforced this point: "If you look at the history of Microsoft's successes, going all way back to the original BASIC or MS thing they all had in common is that we succeeded because we reached out to developers," Gates said during a keynote address.
111. In the broader embedded computing market, Montavista had roughly half of the entire market for embedded Linux distributions by 2004. Other native embedded Linux implementations included Amirix, Coollinux, K Linux, Lineo Embedix, Lynuxworks BlueCat, RedBlue Linux and Red Hat Embedded Linux. Important to note, incorporation of open-source components within a broader system became increasingly common over the decade. For example, Palm began to draw on open-source development by porting the higher levels of its platform to a Linux kernel, as announced in early, 2005. Symbian later converted its longtime development language, OPL, into an open-source product.
112. Hunter and Ready, Inc. co-founded by ready developed the VRTX embedded kernel, which runs systems of the Hubble Space Telescope.
113. Matsumoto (1999).
114. In July 2000, Montavista introduced a real-time scheduler for Linux that eliminated the need for a second kernel operating beneath Linux. Making use of standard features that are already a part of Linux created greater compatibility with Linux APIs. Keller (2000).
115. Anonymous (2000).
116. Childe (2000).
117. Company website.
118. Montavista (2000).
119. Company website.
120. Montavista (2000).
121. For example, in 2000 Montavista's Linux kernel was ported to Instrinsyc Software's "Cerfboard" (hardware and software reference design) platform, intended to support handheld computers, mobile and Internet devices.
122. The earlier use of Hard Hat Linux was as a platform in the sense of a component used in common across multiple implementations, but future versions would begin to define the systems, defining architecture and defining key interfaces.
123. Anonymous (2004).
124. In addition to the general advance of MontaVista's platforms, variants of the platform proliferated. With more variations, each variant tended to become more specialized to a particular application and require less customization off-the-shelf. For example, mobile computing applications once supported by Hard Hat, were later supported by the "Consumer Electronics" version, and later by "Mobilinux."
125. Even by, 2004, Montavista staff included 60 development engineers devoted to kernel and tools development and to supporting customers.
126. Telephone interview, Jim Ready, CEO of Montavista (2005).
127. Symbian already had experience in developing user interfaces.
128. Telephone interview with Jim Ready, CEO of MontaVista (2005).
129. Logic processing and radio communications processing, along with a growing number of key component drivers tended to be integrated on the same silicon to improve performance and power consumption. By the mid-2000s, these system-on-chips tended to increasingly accompany reference designs, or direction for building working smartphones. These designs were specialized both to the operating system and chip elements of the platform.
130. Telephone interview with Jim Ready, CEO of MontaVista (2005).

131. While market leaders in the industry had uniformly had widely integrated platform owners, those projects with highly interdependent partners – such as projects involving Go Corporation, General Magic and the Zoomer project – ended in ill-coordinated development and ill-willed partners (even apart from inefficiencies of over-modularization of designs).

132. Telephone interview with Jim Ready, CEO of MontaVista (2005).

133. At the same time, many of these arguments related to these points are overlapping. For example, the creation of rule-making facilities to orchestrate activity itself increased the productivity of outsiders (as in development kits). Further, the higher productivity of outside developers enhanced the value of the platform, and so on. Clearly, more research is required to investigate the control of assets and components of different kinds and their impact on rule-setting, influence, and orchestration beyond the platform owner's boundaries.

ACKNOWLEDGMENTS

Professor Michael Cusumano deserves special credit for ongoing conversations and exchange on questions of competition and innovation in this industry. This chapter benefitted greatly from sharp insights and careful reading by an anonymous referee and the editors. I would also like to thank industry participants who provided crucial insights, including executives at Diamond, Epocrates, Geofox, Lynuxworks, Mobilinux, NTT Docomo, Oregon Scientific, PalmSource, Psion, Sony, Symbian, Texas Instruments, Trolltech, and to Alessandro Araldi, Peter Bancroft, Larry Berkin, David Childers, Paul Crossley, Michael Davies, Andrie Devries, Richard Doherty, Chris Dunphy, Jerry Epplin, Jessica Figueras, David Gannon, Eric Geruldsen, Eric Giguere, Avner Goren, Todd Gort, George Grey, Randy Guisto, Dan Hantula, Oto Hiroyuki, Virginia Hodges, David Kipping, Evan Koeblentz, J Kuper, Timothy Lehane, Elisabeth Liddell, David Limp, Michael Mace, Steve Mann, Darrin Massena, Gary McFadden, Kip Meintzer, David Nagel, Derek Peachey, Kathleen Peters, Will Pinnell, Robert Quinn, Jim Ready, Inder Singh, Steve Strassman, Christopher Tilley, and Jason Wells. International Data Corporation and Gartner for contributed data used in this study. Several colleagues and friends provided helpful comments on an earlier dissertation chapter version of this chapter, including Ron Adner, Sinan Aral, Nicholas Argyres, Carliss Baldwin, Stefano Brusoni, Annabelle Gawer, Bob Gibbons, Ben Gomes-Casseres, Bernard Garrette, Andrei Hagiu, Alden Hayashi, John Henderson, Rebecca Henderson, Michael Jacobides, Lars Bo Jeppesen, Nicola Lacetera, Karim Lakhani, Richard Langlois, Michael Leiblein, Rafel Lucea, Claude Menard, Marc Meyer, Ramana Nanda, Joanne Oxley, Hazhir Rahmandad, Marc Rysman, Richard Schmalensee, Lourdes Sosa, Fernando Suarez, Marshall Van Alstyne, Eric Van den Steen, N. Venkat Venkatraman, Eric von Hippel and David Yoffie and seminar participants at Academy of Management, Atlanta Competitive Advantage Conference, Bocconi University,

Massachusetts Institute of Technology and the University of Paris. I wish to acknowledge funding and support provided by MIT, HEC-Paris, the Paris Chamber of Commerce, and MIT. All errors are my own.

REFERENCES

Adler, P. S., Mandelbaum, A., Nguyen, V., & Schwerer, E. (1995). From project to process management: An empirically-based framework for analyzing product development Time. *Management Science*, *41*(March), 458–484.

Adner, R., & Kapoor, R. (2010). Value creation in innovation ecosystems: How the structure of technological interdependence affects firm performance in new technology generations. *Strategic Management Journal*, *31*(3), 306–333.

Alchian, A., & Demsetz, H. (1972). Production, information costs, and economic organization. *American Economic Review*, *62*, 777–795.

Alexander, C. (1964). *Notes on the synthesis of form*. Cambridge, MA: Harvard University Press.

Altman, E. J., Nagle, F., & Tushman, M. (2015). Innovating without information constraints: Organization, communities, and innovation when information costs approach zero. In C. E. Shalley, M. A. Hitt, & J. Zhou (Eds.), *The Oxford handbook of creativity, innovation, and entrepreneurship* (pp. 353–379). Oxford: Oxford University Press.

Altman, E. J., & Tripsas, M. (2014). *Product to platform transitions: Organizational identity implications* (September 10, 2014). Harvard Business School Research Paper No. 14–045. Retrieved from SSRN: https://ssrn.com/abstract=2364523 or doi:10.2139/ssrn.2364523

Anonymous. (1996). SuperPsionic. *The Economist*.

Anonymous. (2000). New Kernel offers 3x improvement. *Electronic Engineering Times*.

Anonymous. (2004). *CEO interview: Jim Ready of MontaVista*. LinuxDevices.com.

Anonymous. (2006). The history of windows CE. *HPCFactor*.

Atkinson, P. (2008). *A bitter pill to swallow: the rise and fall of the tablet computer*. Sheffield Hallam University Research Archive (SHURA). Accessed on February 13, 2012.

Bakos, J. Y., & Brynjolfsson, E. (1993a). From vendors to partners: Information technology and incomplete contracts in buyer-supplier relationships. *Journal of Organizational Computing and Electronic Commerce*, *3*(3), 301–328.

Bakos, J. Y., & Brynjolfsson, E. (1993b). Information technology, incentives, and the optimal number of suppliers. *Journal of Management Information Systems*, *10*(2), 37–53.

Bakos, Y., & Brynjolfsson, E. (1999). Bundling information goods: Pricing, profits and efficiency. *Management Science*, *45*(12), 63–82.

Baldwin, C., & Clark, K. (2000). *Design rules volume 1: The power of modularity*. MIT Press.

Baldwin, C., & von Hippel, E. (2011). Modeling a paradigm shift: From producer innovation to user and open collaborative innovation. *Organization Science*, *22*(6), 1399–1417.

Baldwin, C. Y., & Woodard, C. J. (2009). The architecture of platforms: A unified view. In A. Gawer (Ed.), *Platforms, markets and innovations*. Cheltenham: Edward Elgar.

Barnett, S. (2000). Jeff Hawkins: The man who almost single-handedly revived the handheld computer industry. *Pen Computing*, 33.

Becchetti, L., & Paganetto, L. (2001). The determinants of subopT.al technological development in the system company–component producers relationship. *International Journal of Industrial Organization*, *19*(9), 1407–1421.

Bessen, J. E. (2016). *How computer automation affects occupations: Technology, jobs, and skills* (October 3, 2016). Boston University School of Law, Law and Economics Research Paper No. 15–49. Retrieved from SSRN: https://ssrn.com/abstract=2690435 or doi:10.2139/ssrn.2690435

Bessen, S. M., & Farrell, J. (1994). Choosing how to compete: Strategies and tactics in standardization. *Journal of Economic Perspectives*, *8*(2), 117–131.

Boudreau, K. (2007). *Does opening a platform stimulate innovation? The Effect on systemic and modular innovations* (August 2007). MIT Sloan Research Paper No. 4611-06. Retrieved from SSRN: https://ssrn.com/abstract=913402 or doi:10.2139/ssrn.913402

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

Boudreau, K. J. (2012). Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation. *Organization Science*, *23*(5), 1409–1427.

Boudreau, K. J., & Hagiu, A. (2009). Platforms rules: Multi-sided platforms as regulators. In A. Gawer (Ed.), *Platforms, markets and innovations*. Cheltenham: Edward Elgar..

Boudreau, K. J., & Jeppesen, L. B. (2015). Unpaid crowd complementors: The platform network effect mirage. *Strategic Management Journal*, *36*(12), 1761–1777.

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

Boudreau, K. J., & Lakhani, K. (2009). How to manage outside innovation. *MIT Sloan Management Review*, *50*(4), 69.

Boudreau, K. J., & Lakhani, K. R. (2013). Using the crowd as an innovation partner. *Harvard Business Review*, *91*(4), 60–69.

Branstetter, L. G., Drev, M., & Kwon, N. (2015). *Get with the program: Software-driven innovation in traditional manufacturing*. No. w21752. National Bureau of Economic Research.

Bresnahan, T. F., & Greenstein, S. (1999). Technological competition and the structure of the computer industry. *Journal of Industrial Economics*, *47*(1), 1–40.

Bresnahan, T., & Greenstein, S. (2014). Mobile computing: The next platform rivalry. *The American Economic Review*, *104*(5), 475–480.

Butters, A., & D. Pogue (2002). *Piloting palm: The inside story of palm, handspring and the birth of the billion dollar handheld industry*. New York, NY: Wiley.

Carlton, D., & Waldman, M. (2002). The strategic use of tying to preserve and create market power in evolving industries. *RAND Journal of Economics*, *33*(2), 194–220.

Casadesus-Masanell, R., & Yoffie, D. B. (2007). Wintel: Cooperation and conflict. *Management Science*, *53*(4), 584–598.

Cennamo, C., & Santalo, J. (2015). How to avoid platform traps. *MIT Sloan Management Review*, *57*(1), 12.

Chandler, A. D. (1962). *Strategy and structure: Chapters in the history of the American Industrial Enterprise*. Cambridge: MIT Press.

Chesbrough, H. (2003). *Open innovation: The new imperative for creating and profiting from technology*. Boston, MA: Harvard Business School Press.

Chesbrough, H. (2013). *Open business models: How to thrive in the new innovation landscape*. Boston, MA: Harvard Business Press.

Childe. (2000). Real-time claims questioned. *Electronic Engineering Times*.

Choi, J. P., Lee, G., & Stefanadis, C. (2003). The effects of integration on R&D incentives in systems markets. *Netnomics*, *5*(1), 21–32.

Choudary, S., Van Alstyne, M. W., & Parker, G. G. (2016). *Platform revolution: How networked markets are transforming the economy – And how to make them work for you*. NYC, NY: WW Norton & Company.

Church, J., & Gandal, N. (1992). Network effects, software provision and standardization. *Journal of Economic Behavior and Organization*, *40*(1), 85–104.

Cusumano, M. A., Mylonadis, Y., & Rosenbloom, R. S. (1992). Strategic maneuvering and mass-market dynamics: The triumph of VHS over Beta. *Business History Review*, *66*(01), 51–94.

David, P. A., & Greenstein, S. (1990). The economics of compatibility standards: An introduction to recent research 1. *Economics of Innovation and New Technology*, *1*(1–2), 3–41.

Davis, S. J., MacCrisken, J., & Murphy, K. M. (2002). Economic perspectives on software design: PC operating systems and platforms. *Microsoft, Antitrust and the New Economy: Selected Essays*, 361–419.

Davis, S., & Murphy, K. (2000). A competitive perspective on internet explorer. *American Economic Review*, *90*(2), 184–187.

Dillon, P. (May, 1998). The next small thing: What does it take to change the world? Obsession. Tenacity. And lots of mistakes. That's the untold story behind the PalmPilot – a 15-year saga that produced the kind of breakthrough that every startup dreams of., Fast Company no. 15: 97.

Dodson, S., & Hart, M. (1998). *Palm Computing, Inc., 1995: Financing Challenges*. Harvard Business School Case 9-898-090.

Edwards, L. (1997). *Programming Psion computers*. Cheshire, EMCC Software Limited.

Eisenmann, T., Parker, G., & Van Alstyne, M. (2011). Platform envelopment. *Strategic Management Journal*, *32*(12), 1270–1285.

Eisenmann, T., Parker, G., & Van Alstyne, M. W. (2006). Strategies for two-sided markets. *Harvard Business Review*, *84*(10), 92.

Ellison, G., & Ellison, S. F. (2005). Lessons about markets from the Internet. *The Journal of Economic Perspectives*, *19*(2), 139–158.

Eppinger, S., & Ulrich, K. (2015). *Product Design and Development*. McGraw-Hill Higher Education.

European Commission. (1998). IV/JV.12 Ericsson / Nokia / Psion / Motorola REGULATION (EEC) No 4064/89 MERGER PROCEDURE.

Evans, D. S., Hagiu, A., & Schmalensee, R. (2008). *Invisible engines: How software platforms drive innovation and transform industries*. Cambridge, MA: MIT press.

Evans, D. S., & Schmalensee, R. (2010). Failure to launch: Critical mass in platform businesses. *Review of Network Economics*, *9*(4).

Evans, D. S., & Schmalensee, R. (2016). *Matchmakers: The new economics of multisided platforms*. Boston, MA: Harvard Business Review Press.

Evans, P. C., & Gawer, A. (2016). *The rise of the platform enterprise: A global survey*. Retrieved from http://epubs.surrey.ac.uk/811201/

Farrell, J., & Katz, M. L. (2000). Innovation, rent extraction, and integration in systems markets. *Journal of Industrial Economics*, *48*(4), 413–432.

Farrell, J., & Klemperer, P. (2007). Coordination and lock-in: Competition with switching costs and network effects. *Handbook of Industrial Organization*, *3*, 1967–2072.

Farrell, J., & Saloner, G. (1985). Standardization, compatibility, and innovation. *RAND Journal of Economics*, *16*(1), 70–83.

Farrell, J., & Weiser, P. (2003). Modularity, vertical integration, and open access policies: Towards a convergence of antitrust and regulation in the internet age. *Harvard Journal of Law and Technology*, *17*(1).

Felin, T., & Zenger, T. R. (2014). Closed or open innovation? Problem solving and the governance choice. *Research Policy*, *43*(5), 914–925.

Fitzgerald, M. (1994). Microsoft reworks WinPad. *Computerworld*, *28*(48).

Gambardella, A., Raasch, C., & von Hippel, E. (2016). The user innovation paradigm: Impacts on markets and welfare. *Management Science*.

Garud, R., & Kumaraswamy, A. (1995). Technological and organizational designs for realizing economies of substitution. *Strategic Management Journal*, *16*(S1), 93–109.

Gawer, A. (Ed.). (2011). *Platforms, markets and innovation*. Cheltenham: Edward Elgar Publishing.

Gawer, A. (2014). Bridging differing perspectives on technological platforms: Toward an integrative framework. *Research Policy*, *43*(7), 1239–1249.

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

Gawer, A., & Henderson, R. (2007). Platform owner entry and innovation in complementary markets: Evidence from Intel. *Journal of Economics & Management Strategy*, *16*(1), 1–34.

Gibbons, R. (2005). Four formal(izable) theories of the firm? *Journal of Economic Behavior & Organization*, *58*(2), 200–245.

Grossman, S., & Hart, O. (1986). The costs and benefits of ownership: A theory of vertical and lateral integration. *Journal of Political Economy*, *94*(4), 691–719.

Gulati, R., Puranam, P., & Tushman, M. (2012). Meta-organization design: Rethinking design in interorganizational and community contexts. *Strategic Management Journal*, *33*(6), 571–586.

Hagiu, A. (2006). Pricing and commitment by two-sided platforms. *The RAND Journal of Economics*, *37*(3), 720–737.

Hagiu, A. (2007). Merchant or two-sided platform? *Review of Network Economics*, *6*(2).

Hall, R. (2001). *Digital dealing: How e-markets are transforming the economy*. Boston, MA: WW Norton & Co., Inc.

Hart, O. (1995). *Firms, contracts and financial structure*. Oxford: Oxford University Press.

Hart, O., & Moore, J. (1990). Property rights and the nature of the firm. *Journal of Political Economy*, *98*(6), 1119–1158.

Heeb, R. (2003). Innovation and vertical integration in complementary markets. *Journal of Economics and Management Strategy*, *12*(3), 387–417.

Hillman, R. A., & Rachlinski, J. J. (2002). Standard-form contracting in the electronic age. *NYUL Review*, *77*, 429.

Hölmstrom, B. (1979). Moral hazard and observability. *Bell Journal of Economics*, *10*, 74–91.

Hölmstrom, B. (1999). The firm as a subeconomy. *Journal of Law, Economics and Organization*, *15*, 74–102.

Hölmstrom, B., & Milgrom, P. (1991). Multitask principal-agent analyses: Incentive contracts, asset ownership, and job design. *Journal of Law, Economics, and Organization*, *7*, 24–52.

Huang, P., Ceccagnoli, M., Forman, C., & Wu, D. J. (2013). Appropriability mechanisms and the platform partnership decision: Evidence from enterprise software. *Management Science*, *59*(1), 102–121.

Iansiti, M., & Levien, R. (2004). *The keystone advantage*. Boston, MA: Harvard Business School Press.

Jacobides, M. G., Knudsen, T., & Augier, M. (2006). Benefiting from innovation: Value creation, value appropriation and the role of industry architectures. *Research Policy*, *35*(8), 1200–1221.

Jeppesen, L. (2005). User toolkits for innovation: Consumers support each other. *Journal of Product Innovation Management*, *22*(4), 347–362.

Jin, G., & Rysman, M. (2015). Platform pricing at sports card conventions. *The Journal of Industrial Economics*, *63*(4), 704–735.

Jose, A., & Tollenaere, M. (2005). Modular and platform methods for product family design: Literature analysis. *Journal of Intelligent Manufacturing*, *16*(3), 371–390.

Katz, M., & Shapiro, C. (1986). Technology adoption in the presence of network externalities. *Journal of Political Economy*, *94*(4), 823–841.

Katz, M., & Shapiro, C. (1994). Network externalities, competition, and compatibility. *American Economic Review*, *75*(3), 424–440.

Keller, J. (2000). MontaVista unveils real-time scheduler for Linux. *Military and Aerospace Electronics*, *11*(7).

Kim, D. J., & Kogut, B. (1996). Technological platforms and diversification. *Organization Science*, *7*(3), 283–301.

Klein, B., Crawford, R. G., & Alchian, A. A. (1978). Vertical integration, appropriable rents, and the competitive contracting process. *Journal of Law and Economics*, *21*(2), 297–326.

Kogut, B., & Zander, U. (1992). Knowledge of the firm, combinative capabilities, and the replication of technology. *Organization Science 3.3*, 383–397.

Kosnick, T. J., Atluru, R., & Wasserstein, K., (1998). *Palm computing: The pilot organizer*. Harvard Business School Case 9-599-040.

Krishnan, V., & Gupta, S. (2001). Appropriateness and impact of platform-based product development. *Management Science*, *47*(1), 52–68.

Krishnan, V., & Ulrich, K. T. (2001). Product development decisions: A review of the literature. *Management Science*, *47*(1), 1–21.
Langdon, C., & Manners, D. (2001). *Digerati Glitterati*. New York, NY: Wiley and Sons.
Langlois, R. (1992). External economies and economic progress: The case of the microcomputer industry. *Business History Review*, *66*(1), 1–50.
Lawrence, L. (1999). *Code and other Laws of Cyberspace*. New York.
Levin, J. D. (2011). *The economics of internet markets*. No. w16852. National Bureau of Economic Research.
Litchfield, S. (1998). The history of Psion. *Palmtop Magazine*, 20.
MacCormack, A., Rusnak, J., & Baldwin, C. Y. (2006). Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, Forthcoming.
Martin, A., & Orlando, M. (2007). Barriers to network-specific investment. *Review of Economic Dynamics*, *10*(4), 705–728.
Matsumoto, C. (1999). Startup, Moto join movement at Linux World – Linux dons Hard Hat for embedded play. *Electronic Engineering Times*.
Matutes, C., & Regibeau, P. (1998). "Mix and match": Product compatibility without network externalities. *The RAND Journal of Economics*, 221–234.
Meyer, M. H., & Lehnerd, A. P. (1997). *The power of product platforms*. Simon and Schuster.
Microsoft (1992). *Windows for Pen Computing*. Redmond, WA: Microsoft Press.
Microsoft (2004). *Microsoft Windows CE 5.0*. Fact Sheet.
Montavista (2000). *Montavista Software Inc.: The Embedded Linux Experts*. Company Presentation.
Morris, C. R., & Ferguson, C. H. (1992). How architecture wins technology wars. *Harvard Business Review*, *71*(2), 86–96.
Morrison, M. (2001). *Pocket PC: The Unauthorized Guide*. Indianapolis, Indiana, Que.
Murray, J. (1998). *Inside Windows CE*. Redmond, WA, Microsoft Press.
Niedermayer, A. (2013). On platforms, incomplete contracts, and open source software. *International Journal of Industrial Organization*, *31*(6), 714–722.
Nocke, V., Peitz, M., & Stahl, K. (2007). Platform ownership. *Journal of European Economic Association*, *5*(6), 1130–1160.
Northam, P. (2006). *How smartphones work: Symbian and the mobile phone industry*. West Sussex: John Wiley and Sons.
Parker, G. G., & Van Alstyne, M. W. (2005). Two-sided network effects: A theory of information product design. *Management Science*, *51*(10), 1494–1504.
Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, *15*, 1053–1058.
Perrons, R. K. (2009). The open kimono: How Intel balances trust and power to maintain platform leadership. *Research Policy*, *38*(8), 1300–1312.
Pfeffer, J., & Salancik, G. R. (2003). *The external control of organizations: A resource dependence perspective*. Redwood City, CA: Stanford University Press.
Porch, C., Timbrell, G., & Rosemann, M. (2015). *Platforms: A systematic review of the literature using algorithmic historiography*. ECIS.
Porter, M. E. (1985). *Competitive advantage: Creating and sustaining superior performance*. *1985*. New York, NY: FreePress.
Potter, D. (1998). Entrepreneurship: Psion and Europe. *Business Strategy Review*, *9*(1): 15–20.
Price, C. (1998). Psion reshapes product range as sales flag. *The Financial Times*, September 3.
Rajan, R. G., & Zingales, L. (1998). Power in a theory of the firm. *The Quarterly Journal of Economics*, *113*(2), 387–432.
Rhodes, N., & McKeehan, J. (1999). *Palm programming*. Sebastopol, CA: O'Reilley and Associates.
Robertson, D., & Ulrich, K. (1998). Planning for product platforms. *Sloan Management Review*, *39*(4), 19.
Rochet, J.-C., & Tirole, J. (2006). Two-sided markets: A progress report. *The RAND Journal of Economics*, *37*(3), 645–667.

Rysman, M. (2009). The economics of two-sided markets. *The Journal of Economic Perspectives*, *23*(3), 125–143.

Sanderson, S., & Uzumeri, M. (1995). Managing product families: The case of the Sony Walkman. *Research Policy*, *24*(5), 761–782.

Santos, F. M., & Eisenhardt, K. M. (2005). Organizational boundaries and theories of organization. *Organization Science*, *16*(5), 491–508.

Seamans, R., & Zhu, F. (2013). Responses to entry in multi-sided markets: The impact of Craigslist on local newspapers. *Management Science*, *60*(2), 476–493.

Segal, I. (1999). Contracting with externalities. *Quarterly Journal of Economics*, *114*(2), 337–388.

Shapiro, C., & Varian, H. (1998). *Information rules: A strategic guide to the network economy*. Boston, MA: Harvard Business School Press.

Simcoe, T. (2006). Open standards and intellectual property rights. In H. Chesbrough, W. Vanhaverbeke, & J. West (Eds.), *Open innovation: Researching a new paradigm* (pp. 161–183). Oxford: Oxford University Press.

Simon, H. A. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society*, *106*(6), 467–482.

Simpson, T. W., Maier, J. R., & Mistree, F. (2001). Product platform design: Method and application. *Research in Engineering Design*, *13*(1), 2–22.

Smith, T. (2000). S3 unveils badged Psion PDA. *The Register*, October 10.

Smith, T. (2001). SonicBlue at loggerheads with Psion over PDA stockpile. *The* Register.

Sosa, M. E., Eppinger, S. D., & Rowles, C. M. (2000). Designing modular and integrative systems. *ASME Design Engineering Technical Conference Proceedings, DETC00/DTM*. Vol. 14571.

Staudenmayer, N., Tripsas, M., & Tucci, C. L. (2005). Interfirm modularity and its implications for product development. *Journal of Product Innovation Management*, *22*(4), 303–321.

Strahilovetz, L. (2006). Information asymmetries and the rights to exclude. *Michigan Law Review*, *104*.

Suarez, F., & Eisenmann, T. (2004). Symbian: Setting the mobility standard. HBS Case (9-804-076).

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

Suh, E. S., De Weck, O. L., & Chang, D. (2007). Flexible product platforms: Framework and case study. *Research in Engineering Design*, *18*(2), 67–89.

Sundararajan, A. (2016). *The sharing economy: The end of employment and the rise of crowd-based capitalism*. Cambridge, MA: MIT Press.

Sunday Business. (2003). Nokia has Psion in the palm of its hands. *Sunday Business*.

Tarnacha, A., & Maitland, C. (2010). Structural effects of platform certification on a complementary product market: The case of mobile applications. *New Applications in IT Standards: Developments and Progress*. IGI Global, 284–300.

Tasker, M. (2000). *Professional Symbian programming*. Birmingham: Wrox Press.

Taylor, P. (1996). Psion to license operating software. *Financial Times*.

The Economist. (1994). Psion of the Times. *The Economist*.

Tucker, C., & Zhang, J. (2011). How does popularity information affect choices? A field experiment. *Management Science*, *57*(5), 828–842.

Von Hippel, E. (2005). *Democratizing innovation*. MIT press.

West, J. (2003). How open is open enough?: Melding proprietary and open source platform strategies. *Research policy*, *32*(7), 1259–1285.

Weyl, E. G. (2010). A price theory of multi-sided platforms. *The American Economic Review*, 1642–1672.

Whinston, M. D. (2001). Assessing the property rights and transaction-cost theories of firm scope. *The American Economic Review*, *91*(2), 184–188.

Wildstrom, S. (2005). Palm Taps Microsoft. *Business Week*.

Williamson, O. (1971). The vertical integration of production: Market failure considerations. *American Economic Review*, *61*, 112–123.

Williamson, O. (1975). *Markets and hierarchies: Analysis and antitrust implications*. New York, NY: Free Press.

Wilson, R. (1996). Psion, partners open palmtop architecture. *Electronic Engineering Time*.

Winton, G. (2001). *Palm OS network programming*. Sebastopol, CA: O'Reilley and Associates.

Woodard, C. J. (2008). Architectural control points. *3rd International Conference on Design Science Research in Information Systems and Technology (DESRIST 2008)*, May 7, 2008, Atlanta, GA.

Zhu, F., & Furr, N. (2016). Products to platforms: Making the leap. *Harvard Business Review*, *94*(4), 18.

Zhu, F., & Liu, Q. (2014, December). *Competing with complementors: An empirical look at Amazon.com*. Harvard Business School Working Paper, No. 15-044.