DYNAMICS OF PLATFORM-BASED NETWORKS DURING ARCHITECTURAL SHIFTS IN VIDEO GAMES: THEORY AND EVIDENCE

ABSTRACT

We studied the evolution of the platform-complementor network in the PC and home video game consoles as video games transitioned from the sixth to the seventh generation. We specifically sought to understand how the network positions of third-party developers combined with relative platform technology differences influenced launches of videogames that play critical roles in the acceptance of game platforms. We found that developers that were more embedded or locked in to single platform technologies were unable to leverage their existing knowledge and routines in new platform settings and hence, were less likely to port or re-launch their existing game titles. Further, developers were more likely to port titles to platforms that were architecturally similar to their previously supported platforms. As platform architectures converged, they afforded even previously embedded developers an opportunity to migrate and support disparate platforms. Theoretical and empirical insights of this study contribute to our understanding of how platform providers and third-party developers respond effectively to architectural shifts that drive convergence. Specifically, as platform firms increasingly deal with architectural shifts, their ability to translate their core platform architectures and standards to new platform settings to leverage their ecosystem of complementors becomes more critical than ever before.

Keywords: Platforms, network effects, convergence, architectural shifts

INTRODUCTION

Platform-based settings such as video games, mobile telephony, and packaged software often require that customers purchase components that subscribe to the same underlying system architecture (Baldwin & Clark, 1997), and hence, patterns of coordination between hardware and software are central to crafting successful strategies (Evans, Hagiu, & Schmalensee, 2008). Hardware and software firms continually refine and modify their routines to align with constantly shifting architecture of business systems. For example, individual video game software developers launch their games based on their resources and competencies (firm-specific drivers) but also reflect the dynamic characteristics of video game platforms and competitive actions of other developers (Venkatraman & Lee, 2004).

Despite strong concepts that specify mutual dependency between platform providers and complementors for success, empirical research on platform-complementor interactions is limited. Research has focused on the characteristics of platforms and/or those firms that offer them (e.g., Gawer and Cusumano, 2002), while others have focused on complementors supporting the platforms (Afuah, 2000; Afuah and Bahram, 1995). Empirical findings are mostly from studies that are focused only on one actor—either the hardware- or the software- firm but not on their interactions. Venkatraman & Lee (2004) took a first step in this direction to study how the preferential linkage decisions of third-party developers evolved as a function of platform attributes but they did not study how the patterns of support extended by developers varied with their own attributes and also how these patterns of linkage evolved as disparate platform domains converged.

This paper is based on a premise that we need better understanding of the patterns of coordination between platforms and third-party developers as platform technologies evolve and converge with other platforms in terms of functionality and architectures. With increasing overlap across platforms, platform providers translate their existing standards and routines in new settings by attracting favorable sets of complementors. Complementors, on the other hand, are continuously seeking opportunities to re-use and adapt their existing portfolio of applications across disparate platform settings. The impact of architectural shifts that are driving convergence across platforms on the dynamics of platform-complementor interactions, thus, cannot be underestimated and needs to be examined.

Our study adopts the perspective of third-party complementors to explain their decisions to launch games for specific platforms. As developers choose to support a platform, they have the choice to either launch a new game title or port an existing game title from another platform. This distinction is particularly important as game developers reveal their relationship intent with every game launch: they can magnify the importance of a platform by making the game exclusive on it; they can "port" a game from one platform to another and reduce the differentiation between them (Venkatraman and Lee 2004).

We, also, recognize the inherent dynamics in platform evolution due to technology evolution and architectural shifts. Platform providers enter into new but related platform markets (e.g., from PC to mobile or videogame consoles) and they seek opportunities to translate their competencies to the new domain. They do so by bringing their existing standards and routines to new domains and leveraging their complementor network. For instance, when Microsoft entered the video game console market, in order to leverage its ecosystem of complementors from the PC network, it introduced cross-boundary standards and middleware such as the DirectX APIs and

XNA Games Studio that enabled developers to adapt and migrate easily. This had important consequences for the dynamics of competition in the video game console space as Nintendo Wii and Sony PS3 had to alter their architectural strategies to compete with Microsoft's Xbox360 and its large ecosystem of PC complementors. Thus, architectural shifts result in the continuous co-evolution between the patterns of support extended by developers to platforms and the architectural strategies adopted by platforms.

We specifically address the following questions: One, how does developer heterogeneity influence the mode of support extended to platforms? In particular, this study focuses on understanding how developer network attributes (extent of embeddedness in a platform) influence their likelihood of support to a platform through innovative/unique or incremental/ported game titles above and beyond that explained by a firm's resource positions? Two: how do platform attributes influence which platform a developer chooses to support through ported titles? Specifically, we explore how platform network positions and the relatedness between platforms in terms of their technical architectures and complementary software portfolio influence developers' choice of platform to support through ported/incremental titles. Three, how do platform architectural strategies co-evolve with developer network positions to influence developers' choice of support to a platform? To understand these general questions, we study the evolutionary dynamics of platform-complementor networks in the PC and video game industry as the sixth generation technology of video game consoles evolves and transitions to the seventh generation.

BACKGROUND

Architectural Shifts towards Convergence across Platforms

The notion of convergence in the broadest sense is the "blurring of boundaries between

industries by converging value propositions, technologies, and markets (Choi & Valikangas, 2001)." Convergence can occur as products or technologies become increasingly substitutable. From a product viewpoint, Yoffie (1997) defines convergence as "the unification of functions and the coming together of previously distinct products." On the other hand, from a technology point of view, convergence implies that industries that are apparently unrelated from the point of view of nature and uses of the final product are very closely related on a technological basis (Stieglitz, 2002). The manifestation of these two viewpoints is the current coming together of different platform technologies, both in terms of technological convergence with platforms adopting similar standards and in terms of functional convergence with previously disparate platforms converging in terms of functionality and uses by the end customer. Typically, convergence is driven by architectural shifts, which in turn increase the scope of functionalities of products across markets.

Myriad examples of convergence exist in platform settings. For instance, the convergence of computer, telecommunications, electronics, media and entertainment industries (Cusumano & Selby, 1995; Yoffie, 1997) has resulted in firms in the PC industry extending their reach into newer markets to reduce risks associated with substitution and obsolescence. Microsoft released the Xbox console in 2001 to compete in the video game platform space and launched Windows Mobile operating system in 2003 to go beyond the PC architecture. Likewise, Apple was able to leverage its software competencies form PC to music, phones and other devices and functionality.

Architectural shifts have important consequences for platform owners and third-party developers alike as they significantly alter the dynamics of competition in platform-based settings. As architectural shifts blur distinctions between platforms, platform providers find

themselves competing against firms from other industries. Therefore, platform success during periods of convergence depends to a large extent on how platform firms leverage their existing competencies and developer networks in new settings. Degree of support of complementors play key part during this period of architectural convergence.

Convergence has consequences for third-party developers that support platforms. It presents an opportunity for developers to extend their reach and support multiple platforms that transcend industry boundaries simultaneously. Moreover, in order to compete effectively, they need to adapt to newer platform settings and realign their knowledge and routines to the shifting platform architectures. The ability of developers to migrate to newer platforms depends to a large extent on their prior experience with certain platform architectures. As platforms converge and become architecturally similar, they allow developers to re-use their extant knowledge and portfolio of applications in newer platform settings. Therefore, architectural convergence presents opportunities to developers even locked in to single platform technologies to migrate to similar architectures. They further influence the patterns of migration and the patterns of support extended to platforms. Given these important implications of architectural convergence, platform-based competition can be better informed by understanding how and why developers adapt to different platform architectures and how architectural shifts influence the patterns of support extended by developers. To address these important issues, this study uses the PC and video game sectors (sixth and seventh generation consoles) as the research setting.

Video game consoles have undergone significant technological advancement and these changes have resulted in seven distinct generations of video game consoles. Platforms with similar technological characteristics have been grouped together into same generations by industry observers (Corts & Lederman, 2009). Improvements in technology and the launch of

new platform generations present newer challenges to developers, as they have to spend considerable time and effort in learning to work with these new platform technologies. Moreover, as video games have transitioned from the sixth to the seventh generation there has been an increased level of convergence between PCs and video game consoles as gaming platforms which further presents challenges for developers in terms of which platforms they choose to support, the sequence of game launch decisions and the type of support they extend to platforms. Figure 1 demonstrates the architectural shift towards convergence in the PC and video game sectors and the evolution of convergence towards mobile platforms as well. As can be seen from the Figure, in the early days of the PC and console platforms (between 1972-2000) each continued to evolve technologically along their own trajectories without any major shift towards convergence. However, since the launch of the sixth generation consoles in 2000 we observe a shift that is driving convergence across these platform boundaries and this shift towards convergence continues to increase as the seventh generation consoles are launched.

{Insert Figure 1 about here}

THEORY AND HYPOTHESES

What explains a developer's choice to launch a new game title or alternatively re-launch (i.e., port) an existing game title to a platform at any given time? We recognize that since different developers perceive and respond to technology changes and architectural shifts differently, there is heterogeneity in when and how they extend support to competing platforms. The conceptual model used in this study draws on studies of firm network positions to understand developers' ability to support platforms through ported or unique ties. We hypothesize how developer embeddedness influences the pattern of support a developer extends to a platform. We, further hypothesize how platform network positions and the extent of

relatedness between platforms influences developers' choice with respect to which platform to support through ported ties.

Developer Attributes and Platform Support through Ported Ties

Developer Embeddedness

A developers' strategy, in addition to its own firm specific capabilities, evolves as a function of its relative network position in the platform-complementor ecosystem. This is in line with network perspective that recognizes the role of a firm's prior network positions in influencing its future strategic choices (Granovetter, 1985; Gulati, 1999; Powell, White, Koput, & Owen-Smith, 2005; Venkatraman & Lee, 2004).

Marsden defines embeddedness as "the exchanges within a group that have an ongoing social structure that operates by constraining the set of actions available to individual actors and by changing the dispositions of actors towards the actions they may take" (1981). Extending this concept, (Uzzi, 1996) in his study of the apparel industry found that the structural embeddedness of a firm influenced its mortality and reduced its ability to adapt to new requirements (Uzzi, 1997).

Within platform-based settings, actions of developers reflect their prior ties and patterns of embeddedness reflect the distribution of titles offered by developers to different platforms (Venkatraman & Lee, 2004). Therefore, a developer is tightly coupled or embedded when its set of software is spread over a small number of platforms; and as such, this developer does not support a large number of platforms. Consequently, more embedded developers are likely to be locked in to a single platform technology. In settings characterized by rapid advances in technology, embeddedness of a developer may be a liability as a developer is locked into its current offerings and is unable to adapt to changing technology requirements. Therefore,

embedded developers would be less likely to migrate to newer competing platforms and hence more likely to support platforms with unique game titles rather than ported game titles. Thus:

H1: The likelihood of support to platforms through ported ties decreases with developer embeddedness.

Platform Attributes and Support through Ported Ties

Platform Centrality

We now focus on the role of platform centrality in driving a developer's choice to support a platform through ported ties. Network theorists support the view that firms are attracted to more central or dominant actors in a network for reasons of status and legitimacy (Oliver, 1990). Baum & Oliver, (1992) in their study found that day cares were able to increase their legitimacy by partnering with prominent organizations in the community. Similarly, Podolny, (2001) found that the value of connecting to high status partners increased when there was high alter-centric uncertainty i.e. when there is a high amount of uncertainty regarding the quality of the output of the product brought to the market. Other studies have also focused on how ties with dominant actors influences performance outcomes (Ahuja, 2000b; Stuart, 2000b; Zaheer & Bell, 2005). These findings are consistent with the idea of network evolution through the "rich-get-richer" mechanism whereby the most dominant actors continue to attract new links (Barabási, et al., 2002).

Within platform-based settings, indirect network effects manifest through strong positive feedback (Arthur, 1989; Katz & Shapiro, 1994). Third-party developers would be likely to be attracted to more dominant platforms as they offer the biggest potential market for their games (Venkatraman & Lee, 2004). In addition, to benefit from positive feedback and increasing returns (Shapiro & Varian, 1998), developers are more likely to link to dominant platforms first. Therefore, as new technologies evolve and attain dominant positions, developers are more likely

to migrate to them first and support them through unique titles. On the other hand, they will be more likely to migrate to less dominant platforms later once the future viability of these platforms has been determined. Moreover, in order to benefit from the superior market positions enjoyed by dominant platforms, developers would be more willing to advance their know-how along the technological trajectory of the dominant platform and support them through unique titles. On the other hand, they would be more inclined to look for opportunities to reapply their existing know-how to less dominant platforms. Therefore:

H2: The likelihood of support to platforms through ported ties decreases with platform centrality.

Platform Porting Centrality

The next assertion, which also concerns the logic of attachment of a developer to a particular platform offers an alternative explanation to the platform centrality argument made above. Most prior studies within the platform literature have focused on the impact of direct network effects or the market positions of the platforms to explain why developers link to them (Arthur, 1989; Venkatraman & Lee, 2004). However, such explanations do not provide insight into how exogenous factors such as changes in the game development process and the actions of competing developers influence the attachment behavior of third-party developers to platforms.

Recent studies attempting to study the dynamics of network formation and evolution have focused on the rationale for attaching to various actors above and beyond the logic of attachment based on the "rich-get-richer" mechanism whereby actors are most likely to link to partners with the highest ex-ante links (Barabasi, 2002). For instance, Powell, White, Koput, & Owen-Smith (2005) propose an alternative explanation based on the logic of following the trend – here the actors observe others and attempt to match their actions to the dominant pattern of the overall population (White, 1981; DiMaggio and Powell, 1983). The pressure to follow the trend is

derived from a sense of necessity to keep pace with others by acting appropriately and also arises from participants reacting in similar ways to common exogenous factors (Powell, White, Koput & Owen-Smith, 2005).

Within platform-based settings developers are often likely to respond to the actions of their competitors and attachment bias based on the follow-the-trend logic are likely to accrue through two mechanisms. One, as platform architectures have shifted towards convergence due to the increased availability of common third-party standards and middleware, large developers such as Electronic Arts have sought to exploit such technologies to re-release their popular game titles across multiple gaming platforms. Therefore, to keep pace with the increasing competition, competing developers are also likely to be under increasing pressure to leverage the shifting platform architectures to port their existing titles to newer platforms. Two, in addition to increasing the extent of porting, "follow-the-trend" pressures are also likely to determine which platform a developer is most likely to port to. Developers would be more inclined to port first to the platforms that their competitors are porting to as it is likely to reflect the ease of porting to a platform and also the popularity of the platform. Therefore, this study argues that developers would be most likely to port to the platform that already has the most number of ported ties. Thus:

H3: The likelihood of support to platforms through ported ties increases with platform porting centrality.

Platform Architectural Distance

Research on alliances and networks has focused on how firms choose alliance partners and one basis for alliance formation capabilities is learning from prior experience (Gulati, 1999). Firms are driven by routines and once routines become established, firms engage in similar sets

of activities repeatedly that reinforce those routines (Nelson & Winter, 1982) and hence are likely to form ties with actors they had linked to in the past (Gulati, 1995b).

The importance of uncertainty mitigation in partner selection has been emphasized in much of the research on alliances. Previous research has demonstrated that firms observe partner attributes to reduce capability uncertainty and observe network attributes to reduce partner reliability uncertainty (Gulati, 1995a; Gulati, 1999; Shipilov, Rowley, & Aharonson, 2006). To reduce uncertainty associated with their partners, firms were likely to form ties with their past partners and their partners' partners, thereby creating stable networks of ties (Chung, Singh, & Lee, 2000; Gordon Walker, Kogut, & Shan, 1997; Li & Rowley, 2002; Podolny, 1994). For instance, (Podolny, 1994) showed that in markets with high uncertainty, investment bankers tended to interact with those they had worked with in the past.

In platform markets with no formal contractual agreement between platform providers and complementors, extent of partner reliability is likely to be low. Instead, in making their choices, third-party developers are more likely to rely on cues that allow them to reduce their uncertainty with respect to their partners' capabilities. Therefore, in order to reduce uncertainty associated with new platforms, the propensity of developers to form ties with platforms that are architecturally similar to platforms that they have previously supported is likely to be high. Since understanding the architecture of platforms and the associated APIs is crucial to working with these platforms, developers spend considerable time and resources in learning and understanding the platform architectures and APIs. Although platforms typically have their own set of APIs, there are often overlaps in the API sets of platforms. The extent of overlap in their APIs influences the degree of similarity between platforms. When the extent of overlap between platforms is high, developers can support multiple platforms through their offerings with

minimal costs and maximum resource-based synergies (Tanriverdi & Lee, 2008). On the other hand, when a new platform is launched that is architecturally distinct from existing platforms, to reduce the uncertainty associated with the new architecture, developers have to invest significant resources in renewing their know-how and design skills to match those required for the new platform (Nobeoka & Cusumano, 1997).

Therefore, as the architectural similarity between platforms increases, it provides developers with opportunities to reapply their existing knowledge base and exploit resource-based synergies (Meyer & Seliger, 1998). This, therefore, affords more opportunities for developers to be able to re-use their software code, and routines and thus "port" their applications to new settings with minimum modifications (Baldwin & Clark, 1997). Therefore:

H4a: The likelihood of support to platforms through ported ties decreases with platform architectural distance.

During periods of uncertainty, firms seek partners that allow them to mitigate this uncertainty (Beckman, Haunschild, & Phillips, 2004). As a result, when uncertainty is high, firms seek to reinforce their existing relationships and their embeddedness increases. In platform markets, as developers learn the know-how/routines associated with a platform, they are likely to get more embedded in that platform and are hence, unlikely to migrate to new platforms. However, as newer platforms are introduced that adopt common standards and have an increasing overlap in their architectures and APIs with existing platforms, it reduces uncertainty and provides opportunities for developers to adapt quickly to new settings and reapply their knowledge and applications. As a result, even previously embedded developers would find it easier to migrate across platforms and support them through ported ties. Therefore, this study theorizes the role of platform relatedness as a moderator of the relationship between developer embeddedness and likelihood of support through ported ties:

H4b: The impact of complementor embeddedness on likelihood of support through ported ties increases with platform architectural distance.

Platform Genre Distance

Previous research on alliances and networks has demonstrated the extent of partner similarity to an actor's earlier partners as an important determinant of partner choice (Gulati, 1995a; Gulati, 1999; Shipilov, Rowley, & Aharonson, 2006). The following assertion also concerns the role of partner similarity in platform choice but with an important distinction from the platform architecture distance made above.

In platform-based settings characterized by both direct and indirect effects, developer choices are often driven by both architectural and market-based choices. While developers are more likely to want to support platforms that are architecturally similar to platforms they have supported previously, their choice of platform is also likely to be driven by the extent to which the platforms support similar customer markets (Cottrell & Nault, 2004; Tanriverdi & Lee, 2008). Under this argument, platforms are more similar from a market point-of-view rather than from an architectural point-of-view.

Within gaming platforms, the market similarity of platform stems from the extent to which they overlap in terms of the game genres they support. Platforms that support a similar variety of genres are more likely to cater to a similar user base than those platforms that support a different set of genres. Therefore, the choice of porting is often likely to be governed by the user base of the platform. For instance, in the early days of the PC and video game industry, PC games were primarily in the first person shooter category while that genre had a relatively smaller market among consoles. As a result, developers acquired fewer synergies by porting first person shooter games to consoles. On the other hand, by migrating and porting their existing game titles to platforms with the greatest degree of genre overlap, third-party developers can

achieve better synergies by serving the multiple product needs of a similar customer base. Therefore:

H5: The likelihood of support to platforms through ported ties decreases with platform genre distance.

METHODS

Research Setting

This paper studies the network of platform-complementor interactions in the U.S. home videogame and PC sectors from January 2000 to December 2008. This time period allowed us to focus on the time-varying changes in the patterns of support extended to platforms by third-party developers during the transition from the sixth generation to the seventh generation. As discussed previously, the transition of video game platforms from the sixth to seventh generation saw an increased convergence between PCs and video game platforms and their ecosystems of game developers. In addition to the PC, all the major platform players active during this time period were included, namely: Sony, Microsoft and Nintendo and their six platforms spanning the two generations: Playstation 2 (PS2), Xbox, GameCube (GC), Xbox 360, Wii and Playstation 3 (PS3).

Data

The data for this study was obtained from multiple sources. The primary source of data was the website www.gamespot.com. A list of every game developer that has ever launched a game for a platform during the time period of the study (January 2000-December 2008) was compiled and we arrived at 316 developers. Since the intent of this study is to understand the platform choices that developers make over time, every game title that was released by every

developer for every console during the time period of the study was considered. Thus, the primary database used in this study consists of a unique data set of platform, developer and game title combinations for 7 platforms, 316 game developers and 3799 game titles. A tie between a developer and a platform was then coded as each game title launched by the developer on the platform. We further captured the sequence of launch of game titles to determine whether a game title was unique to a platform or ported from a competing platform. The data was further validated with the lists provided on the website www.ign.com. An observation was included only if it was consistent across both websites to ensure robustness and validity to the quality of the data.

Data Analysis

Construct Operationalization

Pattern of support. The dependent variable – a developers' choice to support a platform through either ported ties or unique ties was defined as categorical and time varying. Each time a developer made a decision to launch a game for a platform, they could choose to launch a unique game title for the platform or they could choose to port an existing title to the platform. Moreover, when porting an existing title they needed to decide which platform to port the title to first. Therefore, in the situation that a developer decided to port a game title to a platform, $y_{ij,\,t}$ equaled1 if developer *i* on date *t* ported a game title to console *j*, where console *j* is a console in developer *i*'s choice set $J_{it.}$ For the remaining consoles in the developer's choice set, $y_{ij,\,t}$ equaled 0. In addition, the option to launch a unique title was also included in the developer's choice set and took a value of 0 when a developer ported an existing title. On the other hand, when a developer launched a unique title on a platform, that option in the choice set equaled 1 and all the other platform choices $y_{ij,\,t}$ in the developer's choice set equaled 0. For example, when Electronic

Arts launched the Madden NFL 2002 in August 2001, it launched it as a unique title on Sony PS2. Therefore, the choice of support through a unique tie in the developers' choice set was assigned a value of 1 and the remaining platforms (PC, GC and Xbox) in the developer's choice set equaled 0. For the next release of Madden NFL 2002, Electronic Arts had the option of porting the game to the PC, GC or Xbox platforms. Therefore, PC, GC and Xbox formed the developers' choice set in that time period. Note that PS2 was not part of the choice set anymore as the title had already been launched as a unique tie on it. Since Madden NFL 2002 was ported to the Xbox console in October 2001, $y_{ij,\,t}$ for Xbox equaled 1 and was 0 for GC and PC. Also, since Madden NFL 2002 for Xbox was a ported title, the unique tie option in the choice set equaled 0. As demonstrated in the example, the choice set available to a developer did not remain constant. As new platforms were launched over time, the choice set available to the developers increased. For instance, when the Xbox360 was launched in November 2005, the choice set available to developers after that time period increased as it now included the Xbox360 as well.

Developer Embeddedness. Developer embeddedness was defined using Uzzi's (1996) and Venkatraman and Lee's (2004) measure of first-order coupling. Developer embeddedness was $\sum_j(P_{ij,\,t})^2$ where P_{ij} was the proportion of titles launched by developer *i* on platform *j* at time period *t*. A value of 1 implied that all of a developer's titles were launched for a single platform while a value close to 0 implied that a developer's offerings were spread across multiple platforms.

Platform Centrality. Each game title launched by developer i on platform j was coded as a tie. Platform centrality at time t was then defined as the cumulative number of titles available for platform j divided by the total number of titles in the platform-complementor network.

Platform Porting Centrality. This measure sought to capture the extent to which the dominant porting trend at time t influenced a developer's choice to port to a platform. Platform porting centrality at time t was defined as the number of ported titles for platform j divided by the total number of ported titles in the platform-complementor network.

Platform Architecture Distance. The platform architecture distance measure reflected the impact of a developer's prior architectural choices on their subsequent porting decisions. The first platform that developer *i* launched a game title on was defined as the source platform *k* and the platform that the developer chose to port the same game title to was the target platform *j*. Platform architecture distance was defined as the distance in the platform environments of the source platform *k* and target platform *j*. Differences in hardware, OS and platform APIs and development environments were used to compute the architectural distance between platforms in two steps. In Step 1, the total distance between the hardware and OS and the total distance between the development environments and APIs between each set of platforms was computed as:

$$\text{Hardware}_{(jk)} = (0.2) \sum(\text{Hardware and OS differences}) \text{ and}$$

$$\text{API}_{(jk)} = (0.2) \sum(\text{Dev. Environment and API differences})$$

Here, j denotes the source platform while k denotes the target platform. Also, each component within the hardware and OS differences and within the API and development environment differences was assigned equal weights of 0.2.

In Step 2, the aggregate platform architecture distance measure was computed and is defined as:

$$\text{Platform Architecture Distance} = w_{jk}(\text{Hardrware}_{jk}) + w_{jk}(\text{API}_{jk})$$

Here, w_{jk} is the weight assigned to hardware/OS and API/development environment differences. Specifically, w_{jk} was assigned values of 0.8-0.9 for the API and development environment differences and values of 0.1-0.2 for the OS and hardware differences. Two alternative schemes were also used to measure $Hardware_{jk}$ and API_{jk}. In the first measure $Hardware_{jk}$ and API_{jk} were assigned a value of 0 or 1 depending on whether the source and platform were based on the same standards. The alternative weighting scheme used a more nuanced measure of the differences across platforms – $Hardware_{jk}$ and API_{jk} were assigned a weight of 0 if the source and target platform were based on the same standards, a weight of 0.5 if they were based on similar yet different standards and a value of 1 if they were based on completely different standards.

Platform Genre Distance. Platform genre distance was a variable introduced to measure for the impact of the product market relatedness of the source and target platforms. It was defined as the difference in the proportion of titles in genre *g* of the game between the source platform *k* and target platform *j* in time *t*. This measure captured the extent to which developers' were more likely to port game titles to platforms that had the most similar portfolio of games to the target platform.

Developer Age. Developer age was defined in months at release date *t* of the game title as the difference between date t and the date of developer *i*'s first release of a game title.

Developer Centrality. Each game title launched by developer *i* on platform *j* was coded as a tie. Developer centrality at time t was then defined as the cumulative number of titles available released by developer *i* divided by the total number of titles in the platform-complementor network.

Platform Age. Platform age for target platform *j* was defined in months as the difference between the release date *t* of a game title and the launch date of the console.

Platform firm fixed effects. The seven platforms that were the focus of this study were launched by three firms – Microsoft (PC, Xbox and Xbox360), Sony (PS2 and PS3) and Nintendo (GC and Wii). Firm level dummy variables were introduced for the source platform *k* to control for firm specific differences on the likelihood of porting to competing platforms.

Nested Logit Model for Support through Ported Ties vs. Unique Ties

The unit of analysis for this study was a developer's choice to support a specific platform through the launch of unique or ported game titles. Thus, the econometric approach used was the nested logit model (Ben-Akiva, 1973). The nested logit model is an extension of the most commonly used choice model – the multinomial logit model (McFadden, 1986). This model is commonly used when the dependent variable consists of a choice from an unordered set of alternatives and the choice is a function of both the attributes of the alternatives in the choice set (alternative specific variables) and also the attributes of the individual making the choice (case specific variables).

Another useful attribute of the nested logit model is the ability to model the "no-choice option". An important motivation for including this "no-choice" option in studies is that it is a signal of a preference structure where an individual making a choice has an option of not selecting any of the alternatives offered to them. For instance, consider the situation where an individual makes a choice out of three options. The nested logit model would regard the two alternatives A and B as closer substitutes than the no-choice option. Therefore, any increase in the probability of choosing option A will result in a larger decrease in the probability of choosing B than in the probability of choosing the no-choice option (De Blaeij, Nunes, & Van, 2007).

In this study, the goal was to understand how developer attributes and platform attributes influence a developers' choice to support a platform through either ported ties or unique ties. This study, also, further sought to explain which platform a developer was most likely to choose once the decision to port was made. Modeling the developer's choice as a nested logit model was appropriate for two key reasons: first, the nested logit model allowed the modeling of a developer's choice of support as a function of both the alternative specific (platform specific) variables and case specific (developer specific) variables. Second, in the context of this study, the no choice option was akin to the option of the developer choosing not to port an existing title to a platform and instead launch a unique title for the platform. This formulation allowed analyzing the entire data set of game-developer-platform combinations to understand why developers were more or less likely to port their existing titles and if they decided to port, which platform they were most likely to choose to port to. Under this formulation of developer choice, the likelihood that a developer i chose to port to platform j was defined as:

Prob(dev *i* chooses platform *j* to port)=*Prob*(*i* chooses *j*|*i* chooses to port)**Prob*(*i* chooses to port)

Here, the probability of developer i choosing to port was a function of the developer specific attributes and the probability that developer i chose to port to platform j was a function of the platform specific attributes. In this study with the choice to support platforms through unique ties (the no choice option in this study), the nesting structure allowed the platform-specific attributes for the support through unique tie choice to drop to zero. The nesting structure used in this study is shown in Figure 2.

The data sample for this study consisted of 3599 unique developer-title-platform combinations. The estimation sample included all of the consoles not chosen in each of the 3599

cases. In addition, the estimation also sample also included the choice of supporting a platform through a unique tie. Moreover, the maximum number of consoles available in the choice set varied over time as new consoles were added to the choice set. The final estimation samples thus consisted of 20,254 observations. Summary statistics for this analysis are included in Table 1.

{Insert Figure 2 and Table 1 about here}

RESULTS

In this section, the results of the nested logit model for support by developers through ported and unique ties are presented. The analysis was built through a series of three models. All models were estimated using Stata 10.0. In Model 1, all platform specific variables and two developer specific control variables – developer age and developer centrality were included. In Model 2, the third developer specific variable – developer embeddedness was added and finally, in Model 3 the interaction between developer embeddedness and platform architecture distance was added to the analysis. The results discussed below are drawn from Model 3. The results are shown in Table 2. In this analysis, the architecture distance measure used was with $w_{jk} = 0.9$ for API and development environment differences and 0.1 for hardware and OS differences; $s_{jk} = 0$, 0.5 or 1.

Hypothesis 1 focuses on the impact of developer embeddedness on the likelihood of support to platforms through ported ties or unique ties. Since the developer embeddedness variable along with the other developer specific control variables (age and centrality) are case specific variables and do not vary with each of the alternatives chosen, in the nested logit model they are interpreted at the level of the nest. Therefore, in this analysis, the ported tie nest was the base and all coefficients were interpreted with respect to this nest.

Hypothesis 1 highlights the relationship between developer embeddedness and the type of support extended by developers to platforms. The thesis that more embedded developers are likely to support platforms through unique titles was strongly supported by the analysis (coefficient was 55.88 which was positive and significant ($p<0.001$) for the unique tie nest). Our explanation is that developers that are locked into a single platform technology find it difficult to adapt and migrate to newer platforms. Hence, they support platforms through unique ties.

Hypotheses 2, 3, 4a, 4b and 5 focus on the relationship between platform attributes and the likelihood of support to a platform through ported ties. Hypothesis 2 predicts that platform centrality negatively influences the likelihood of support to a platform through ported ties. The coefficient for platform centrality was 23.24, which was positive and significant ($p<0.001$). The interpretation is that developers were more likely to port their existing titles to more dominant platforms. This is contrary to the expectation that as platforms are more dominant, developers are more likely to want to support them first through unique titles and then migrate to less dominant platforms. One potential explanation for this finding could be that developers are actually making support decisions based on the platform architectures and are most likely to develop applications for the simplest platform architecture first. This finding underscores the role of architectures in determining the patterns of support developers' extend to platforms.

Hypothesis 3 theorizes the relationship between platform porting centrality and the likelihood of support through ported ties. The platform porting centrality measure served to explain the extent to which developers employed a "follow-the-trend" strategy (Powell, White, Koput, & Owen-Smith, 2005) in their porting choices. The platform porting centrality, thus, reflected the extent to which a developer was most likely to port to platforms that already had the most number of ported titles at the time of the decision. The coefficient of platform porting

centrality was 61.77, which was positive and significant ($p<0.001$) indicating that developers did employ a "follow-the-trend" strategy in their porting decisions.

Hypothesis 4a focuses on the impact of the architectural distance between the source and target platforms in influencing a developer's choice of platform to port an existing game title to. The coefficient of architectural distance was -39.37, which was negative and significant ($p<0.001$) and thus strongly supports the hypothesis that developers were most likely to port to platforms that were architecturally most similar to platforms that they had supported in the past. This finding emphasizes that in addition to making their initial choice decisions based on platform architectures, the sequence of launch of game titles is also a function of platform architectures.

Hypothesis 4b predicts how the architectural distance between platforms moderates the impact of developer embeddedness on the pattern of support extended to a platform. The coefficient of the interaction between developer embeddedness and architectural distance was 84.94, which was positive and significant ($p<0.001$). The interpretation is that the impact of architectural distance in determining the pattern of support is higher for more embedded developers. Therefore, as platform architectures converge and become more similar, even developers that are locked in to a platform technology can adapt and migrate to newer platforms sooner than when platforms architectures are radically different.

Hypothesis 5 focuses on the impact of the genre distance between the source and target platforms in driving developer support through ported ties. The variable genre distance was included to understand the extent to which the platform market relatedness influenced a developers' choice of platform to port its existing titles to. The thesis is that developers' would be most likely to port to platforms that catered to the similar user base. The distribution of genres

available across these platforms served as a proxy for the market relatedness of platforms. The coefficient of genre distance was -73.53, which was positive and significant ($p<0.05$). This supports the expectation that developers would be more likely to port to platforms with a similar portfolio of genres.

In order to test the robustness of the analysis, we also re-estimated the models with the alternative measures of platform architecture distance. The results with each of this these alternative measures were consistent with the results obtained from Model 3, thereby, underscoring the robustness of the analysis.

FURTHER VALIDATION

We further ventured to understand at the macro level how architectural shifts during the transition of video game consoles from the sixth to the seventh generation influenced the dynamics of platform-complementor network evolution. For this purpose, we focused only on the support of platforms through ported ties and how this support extended by developers evolved as consoles transition to the newer generation. Therefore, in order to understand the impact of architectural shifts on the dynamics of network evolution, an alternative nesting structure was used (Figure 3). Under this nesting structure, the video game consoles were nested according to the generation they belonged to while the PC platform constituted a separate nest. Nesting platforms in this manner helped in explaining the extent to which the convergence between PCs and seventh generation consoles helped developers reuse their existing knowledge and code base and support platforms through ported ties.

{Insert Figure 3 about here}

The results obtained from this analysis broadly reflected the patterns of porting during the transition from the sixth to the seventh generation. We found that the extent of porting increased

as video game consoles underwent technological shifts and transitioned to the seventh generation. The second key finding was that in addition to an increase in the extent of porting, specifically the extent of porting between the seventh generation consoles and the PC platform increased. This clearly emphasized the role of architectural convergence between the seventh generation consoles and PC in driving the dynamics of support extended by third-party developers.

DISCUSSIONS

Our goal in this study has been to contribute to our understanding of the evolution of platform-based competition during periods of architectural convergence and explain how networks of coordination between developers and platform providers co-evolve with architectural shifts. Our analysis adopted the perspective of developers that delivered complementary software products to understand how and why they chose to embrace certain platforms. The study specifically sought to empirically examine how the extent of developer embeddedness influenced their decision to support platforms through ported ties or unique ties. The analysis also further examined how platform attributes (centrality and architecture distance) influenced the choice of platform to port to. Finally, the impact of the interplay between decreasing architectural distances between platforms and developer embeddedness on platform choice of support was examined.

The single most dominant factor in driving the choice of support to a platform was the extent to which developers' were embedded within certain platform architectures. The concept of embeddedness has been adopted extensively in network research (for instance, (Granovetter, 1985; Uzzi, 1996; Gulati & Gargiulo, 1999; Venkatraman & Lee, 2004; Powell, White, Koput, & Jason Owen-Smith, 2005)) in an attempt to understand the preferential attachment choices of

actors. Previous research suggests that embeddedness constrains the set of alternatives available to actors (Marsden, 1981) and hence, results in the formation of repeat ties with existing partners (Gulati & Gargiulo, 1999). The finding of this study was that developers that were more embedded in certain platform technologies were less likely to adapt to newer platforms and hence support platforms through ported ties. This result adds credence to the role of embeddedness in driving network evolution albeit with an important distinction. Within platform-based settings embeddedness arises as a result of lock-in to platform architectures, which differs from the notion of embeddedness as emerging from trust as has been the focus of prior studies. These findings emphasize that developer actions cannot be studied in isolation and are largely a function of where they are positioned relative to existing platforms and other competing developers in the platform-complementor network.

The analysis further attempted to understand how platform attributes influenced developers' sequence of launch decisions. Our analysis revealed that developers supported platforms in superior market positions with ported ties. One potential explanation for this finding could be that developers are actually making support decisions based on the platform architectures and are most likely to develop applications for the simplest platform architecture first. The findings also emphasize that platform dominance or market viability is not permanent (Venkatraman & Lee, 2004) and developers need to balance their commitment to platforms based not only on platform market positions but also based on their relative architectural positions with respect to competing platforms.

Our study also highlighted the importance of platform architectures in reinforcing the evolutionary dynamics of platform-complementor networks. The propensity of developers to form ties with platforms that were architecturally similar to platforms that they had previously

supported was strongly validated in this study. When the extent of overlap between platforms is high, developers can support multiple platforms through their offerings with minimal costs and maximum resource-based synergies (Tanriverdi & Lee, 2008).

By empirically examining the time-varying changes in the patterns of support extended by developers as the technical characteristics of platforms evolved, the analysis also sought to address the interplay between platform architectural strategies and developer network positions. Specifically, the question of how the interplay between platform architectures and developer embeddedness in these architectures influenced the patterns of network evolution was addressed. The results corroborated the thesis that as newer platforms that adopt common standards and have an increasing overlap in their architectures and APIs with existing platforms are introduced, it reduces uncertainty and provides opportunities for developers to adapt quickly to new settings and reapply their knowledge and applications. As a result, even previously embedded developers would find it easier to embrace newer platforms and support them through ported ties.

These findings further emphasize the evolving and dynamic nature of the coordination between platform providers and their ecosystem of complementors and the crucial role of platform architectures in driving network dynamics. By delving into the architectural nuances of individual platforms relative to competing platforms, this study provides insight into the intricacies of how platform provider and third-party developer actions co-evolve. From a strategic management point of view, the findings urge platform providers to pay attention to their strategies with respect to architectural coordination. Their architectural strategies need to reflect the constant balance between their "competition-cooperation" choices. Specifically, they need to balance the trade-off between having highly similar architectures relative to competing platforms that encourage developers to support their platforms with ported game titles versus launching

platforms with radically different architectures, which compel developers to support them with exclusive game titles.

We further ventured to understand how the architectural shift from the sixth to seventh generation video game consoles altered the overall dynamics of network evolution. Specifically, the analysis sought to explain the consequences of the architectural convergence between PC and consoles driven during the transition to the seventh generation on the patterns of support provided by developers. Table 3 demonstrates the overall patterns of porting during this period of transition from the sixth to seventh generation. The extent of porting is denoted as positive or negative with respect to sixth generation platforms as the base. The analysis revealed that the extent of porting increased for seventh generation video game consoles relative to the previous generation. The second key finding was that in addition to the extent of porting increasing, specifically the extent of porting between the seventh generation consoles and the PC platform increased.

{Insert Table 3 about here}

The change in the patterns of tie formation – specifically the overall increase in the level of ported ties could be attributed to the architectural shifts resulting from the transition to the seventh generation consoles. As consoles have improved technologically over time, so has the availability of third-party middleware to support the game development process. In the early days of the video game industry, consoles were incompatible with each other and developers needed to spend a considerable amount of time and effort in redesigning, rewriting and retesting games for different consoles making it cumbersome to develop similar game titles for multiple platforms. However, over time, while the hardware has remained incompatible, the rise of sophisticated middleware has reduced the costs associated with porting games across platforms.

This implies that software code, designs and components developed for one console can be ported within minimum modifications to other consoles (Corts & Lederman, 2009). The ability to port games across consoles has, thus, resulted in an overall reduction in unique ties between platforms and complementors and has instead led to a greater overlap of game titles across platforms.

The overall increase in ported ties between PC and seventh generation video game consoles highlights to a large extent the role of the entry of Microsoft into the video game console space. As Microsoft entered the video game sector it brought with it dominant PC standards such as the DirectX APIs. In addition, it also launched the XNA Game Studio, which is a set of tools that allows developers to build games for both the Microsoft Windows and Xbox 360. This allowed its existing ecosystem of developers to support PC and video game platforms contemporaneously. Therefore, the extent of porting between Microsoft platforms was significantly higher than porting among seventh generation consoles. Even though Microsoft made its initial foray into the video game console space with the launch of the sixth generation Xbox console, part of the reason for the increase in porting in the seventh generation in addition to the launch of the XNA platform could be attributed to the initial inertia associated with supporting new platforms. Developers were more likely to adopt a wait-and-see approach before committing to a new innovation and hence migrated once Microsoft had established a presence in the console space (Ginsberg & Venkatraman, 1992).

While this study focused only on the convergence between PCs and video game consoles, the phenomenon of convergence is not unique to this setting and thus, these findings present important implications for other platform-based settings as well. Specifically, as platform providers expand their horizons and strive to enter new platform markets, their success depends

upon the extent to which they are able to leverage their existing ecosystem of complementors and complementary software. Consequently, platform firms often provide the same standards and APIs across the SDKS for the repertoire of platforms they support. This provides developers the opportunity to rapidly deploy games across seemingly disparate platforms without even having to invest in expensive middleware to assist in game porting. For instance, when Apple opened up the iPhone platform to third-party developers, it based the iPhone OS on the Mac OS kernel and it's SDKs and toolkits included the same development environment and tools such as XCode and APIs as those for the Mac. As a result, a large number of Mac developers such as Aspyr that were not able to adapt to other platforms such as PCs and video game consoles were able to migrate and port their existing Mac applications to the iPhone platform. This further presents evidence of the extent to which architectural convergence drives the migration path and launch patterns for developers. Migration paths such as Mac → iPhone → iPad are likely to be more common than migration across traditionally defined platform boundaries such as from Mac to PC or iPhone to other competing mobile platforms such as Android and Blackberry.

CONCLUSIONS

We contributed to the understanding of platform-based competition by focusing on why and how third-party developers choose to link to platforms as they evolve technologically and converge across industry boundaries. A holistic empirical examination of the dynamics of platform-complementor interactions was possible because the PC and the video game sectors together provided a unique context in which to examine the impact of specific architectural shifts and the consequent convergence on network evolution.

Another key contribution of this study has been the acknowledgement that the propensity to link to platforms through different modes of support varies across developers and that the

choice of mode of support has serious implications for platform performance. Specifically, developers supporting multiple platforms often have the choice of extending support in the form of original/unique applications or in the form of ported applications (where the same application is re-used in a new setting). Recent research in the video game sector has demonstrated that platform dominance is driven largely by the extent to which a platform is supported by unique applications released exclusively for a platform (Corts & Lederman, 2009). Therefore, the choice to support platforms through either unique or exclusive software is a critical strategic move that has so far received scant attention in platform-based research.

Additionally, this study has contributed to our understanding of the role of technological change and software standards in altering the dynamics of network evolution in platform-based settings. Most network studies have focused on the role of network properties such as degree centrality, and network embeddedness in driving partner selection. However, few attempts have been made to study the combined role of network characteristics and technological changes in changing the dynamics of network topology. This study is a first attempt to allow us to discern the role of architectural compatibility between platforms and the role of design operators like porting on the preferential attachment decisions of complementors. To this end, the development of the architectural distance measure used in this study is an important first step towards understanding the co-evolution between architectural convergence and platform-complementor network evolution. To our knowledge, no prior studies have attempted to develop such a detailed and nuanced measure of the architectural distance for each platform dyad at each level of the software stack.

Finally, this study has leveraged some of the theoretical and empirical insights to contribute to our understanding of how platform providers and third-party developers can

respond effectively to architectural shifts and convergence. Specifically, this study contends that as platform firms shift away from competing within traditionally defined platform boundaries to competing across platform boundaries, their ability to translate their core platform architectures and standards to new platform settings to leverage their ecosystem of complementors will become more critical than ever before.

This study is a valuable first step in discerning the dynamics of platform-complementor network evolution during periods of architectural shifts. The results of this study provide insights that go beyond simply observing platform-complementor interactions in a single platform setting and urge researchers dealing in settings characterized by high-tech, platform-based competition to embrace concepts and approaches from network perspectives. The results, however, need to be seen against the backdrop of some limitations that point directions for further refinement and extensions.

The setting for this study was the PC and the sixth and seventh generation video game consoles and their ecosystem of third-party game developers. While this setting provided a unique context within which to examine the impact of architectural shifts and convergence on network evolution, the phenomenon of architectural convergence is not unique to this setting. Indeed, in recent times there have been numerous instances of convergence across myriad platform settings such as mobile platforms, handhelds, tablets and e-book readers. Therefore, a useful extension to this paper could be to test the hypotheses in a new setting such as smartphones where platforms like Windows Mobile, Apple iPhone and Google Android are competing for third-party applications. This would allow us to test the generalizability of the results obtained within the context of the PC and video game sectors.

Our study examined how developer and platform attributes together influenced how and why developers chose to support a platform. However, there may be factors above and beyond those addressed in this study that influence why a developer chooses to link to a platform. For instance, future research could focus on how platform providers can increase the heterogeneity of their platforms in the form of toolkits they provide to make them more attractive for third party developers to develop exclusive game titles. Additionally, another important variable that drives developer support in platform settings are direct network effects or the installed base of a platform. However, due to data limitations on the availability of installed base data for the PC platform, the impact of this variable was not included in this study. Therefore, in order to best control for direct network effects platform firm specific control variables were introduced. However, future research could attempt to introduce the installed base variable as such data becomes available.

While this paper opened up the 'black-box' of platform architectural differences by focusing on the commonality of development architecture and similarity of development tools, further research is clearly needed to develop superior and more generalizable metrics to measure architectural distance. More nuanced operationalizations will allow us to better understand how the technical architecture impacts the choices made by developers to provide the requisite applications that give rise to the network effects.

To summarize, although additional research that more closely examines the co-evolution between architectural shifts and developer choice of support to a platform across multiple platform boundaries is warranted, this study nonetheless contributes to a better understanding of the dynamics of competition in platform-based settings.

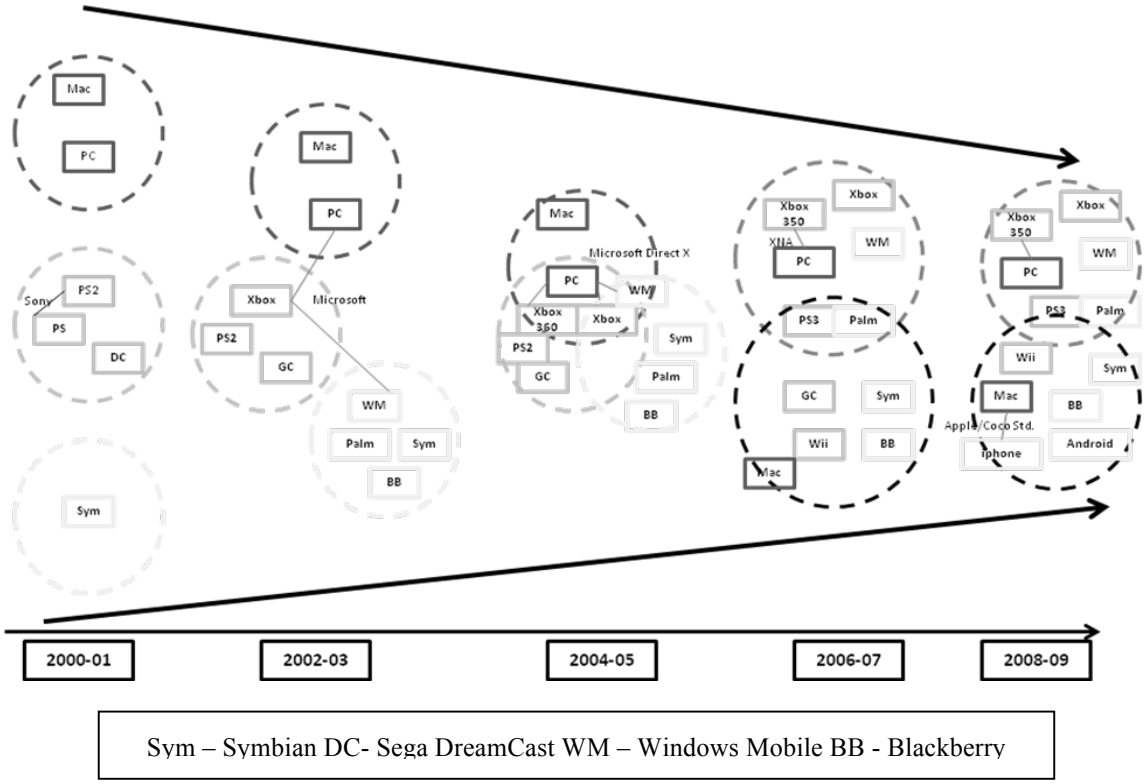

Figure 1. Architectural Convergence in PC, Video Game and Mobile Platforms

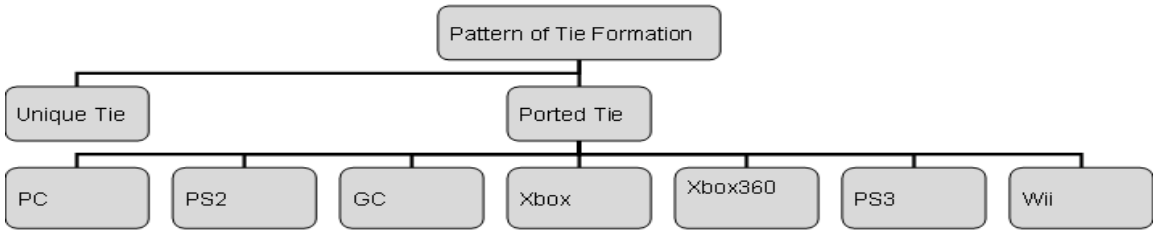

Figure 2. Nested logit model for developer support through porting vs. unique tie

Table 2. Summary Statistics

	Dev. Age	Dev. Centrality	Dev. Embeddedness	Platform Centrality	Platform Architecture Distance	Platform Age	Platform Genre Distance
Dev. Age	1						
Dev. Centrality	-0.22*	1					
Dev. Embeddedness	0.44*	-0.58*	1				
Platform Centrality	-0.18*	0.25*	-0.32*	1			
Platform Architecture Distance	-0.19*	0.32*	-0.44*	0.58*	1		
Platform Age	-0.16*	0.27*	-0.42*	0.73*	0.71*	1	
Platform Genre Distance	-0.08*	0.29*	-0.40*	0.69*	0.76*	0.73*	1
Mean	*26.26*	*0.02*	*0.63*	*0.04*	*0.11*	*11.14*	*0.007*
S.D.	*17.44*	*0.02*	*0.31*	*0.11*	*0.23*	*23.60*	*0.02*

N= 20,254; p<0.05

Table 2. Results for Analysis 1

	Model 1	Model 2	Model 3
Platform-Specific			
Platform Centrality	0.95† (0.76)	7.12*** (1.45)	23.24*** (2.33)
Architecture Distance	-3.18*** (0.28)	-3.93*** (0.72)	-39.37*** (2.97)
Platform Age	*-0.05**** *(0.004)*	*-0.05**** *(0.007)*	*-0.03**** *(0.01)*
Platform Porting Centrality	19.03*** (1.01)	30.83*** (2.04)	61.77*** (4.03)
Platform Genre Distance	-100.74*** (9.94)	-99.88*** (16.14)	-73.53*** (6.63)
Architecture Distance*Dev Embeddedness			84.94*** (6.71)
Developer-Specific: Ported ties base			
Developer Age	*0.0015**** *(0.00006)*	*0.0008**** *(0.00009)*	*0.0004**** *(0.0001)*
Developer Centrality	*6.9* *(21.86)*	*33.88* *(42.46)*	*16.76* *(18.42)*
Developer Embeddedness		13.05*** (0.80)	55.58*** (4.21)
Sony	*1.42**** *(0.21)*	*0.77** *(0.28)*	*0.92** *(0.37)*
Nintendo	*3.76**** *(0.21)*	*2.61**** *(0.33)*	*1.37** *(0.64)*
Log-likelihood	-2550.54	-2120.68	-1931.06
Δ(-2LL)		879.72***	379.32***

N = 20,254; ***p<0.001; **p<0.01, *p<0.05; †p<0.1

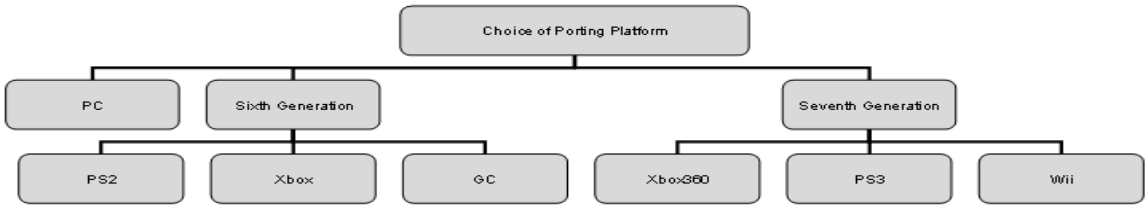

Figure 3. Nested logit model for developer support through ported ties during transition from sixth to seventh generation

Table 3. Patterns of Porting during Transition from Sixth to Seventh Generation

		Target Platform Generation		
		PC	**Sixth**	**Seventh**
Source Platform Generation	**PC**	N/A	-	+
	Sixth	-	base	+
	Seventh	+	-	+

REFERENCES

Ahuja, G. 2000b. The duality of collaboration: Inducements and opportunities in the formation of interfirm linkages. ***Strategic Management Journal***, 21(3, Special Issue: Strategic Networks): 317-343.

Arthur, W. B. 1989. Competing technologies, increasing returns, and lock-in by historical events. ***The Economic Journal***, 99(394): 116-131.

Baldwin, C. Y., & Clark, K. B. 1997. Managing in an age of modularity. ***Harvard Business Review***, 75(5): 84-93.

Barabási, A. L., Jeong, H., Néda, Z., Ravasz, E., Schubert, A., & Vicsek, T. 2002. Evolution of the social network of scientific collaborations. ***Physica A: Statistical Mechanics and its Applications***, 311(3-4): 590-614.

Baum, J. A. C., & Oliver, C. 1992. Institutional embeddedness and the dynamics of organizational populations. ***American Sociological Review***, 57(4): 540-559.

Beckman, C. M., Haunschild, P. R., & Damon J. Phillips. 2004. Friends or strangers? firm-specific uncertainty, market uncertainty, and network partner selection. ***Organization Science***, 15(3): 259-275.

Ben-Akiva, M. 1973. ***Structure of passenger travel demand models.*** Ph.D., Department of Civil Engineering, MIT, Cambridge.

Choi, D., & Valikangas, L. 2001. Patterns of strategy innovation. ***European Management Journal***, 19(4): 424.

Chung, S., Singh, H., & Lee, K. 2000. Complementarity, status similarity and social capital as drivers of alliance formation. ***Strategic Management Journal***, 21(1): 1.

Clements, M. T., & Ohashi, H. 2005. Indirect network effects and the product cycle: Video games in the U.S., 1994-2002. ***The Journal of Industrial Economics***, 53(4): 515-542.

Corts, K. S., & Lederman, M. 2009. Software exclusivity and the scope of indirect network effects in the U.S. home video game market. ***International Journal of Industrial Organization***, 27(2): 121-136.

Cottrell, T., & Nault, B.R. 2004. Product variety and firm survival in the microcomputer software industry. ***Strategic Management Journal***, 25(10): 1005-1025

Cusumano, M. A., & Selby, R. W. 1995. What? Microsoft weak? ***Computerworld***, 29(40): 105.

De Blaeij, A. T., Nunes, P. A. L. D., & Van, D. B. 2007. 'No-choice' options within a nested logit model: One model is insufficient. ***Applied Economics***, 39(10): 1245-1252.

DiMaggio, P.J., & Powell, W. 1983. The iron cage revisited: Institutional isomorphism and collectivity rationality in organizational fields. ***American Sociological Review*****,** 48(2): 147-160.

Eisenmann, T. R. 2008. Managing proprietary and shared platforms. ***California Management Review***, 50(4): 31-53.

Eisenmann, T. R., Parker, G., & Van Alstyne, M. 2008. Opening platforms: How, when and why? ***Working Papers -- Harvard Business School Division of Research***: 1-27.

Evans, D. S., Hagiu, A., & Schmalensee, R. 2008. ***Invisible engines how software platforms drive innovation and transform industries***. Boston: MIT Press.

Gallagher, S., & Park, S. H. 2002. Innovation and competition in standard-based industries: A historical analysis of the U.S. home video game market. ***IEEE Transactions on Engineering Management***, 49(1): 67.

Gandal, N., Kende, M., & Rob, R. 2000. The dynamics of technological adoption in

hardware/software systems: The case of compact disc players. ***RAND Journal of Economics***, 31(1): 43-61.

Gawer, A., & Cusumano, M. A. 2002. Platform leadership: How Intel, Microsoft, and Cisco drive industry innovation (hardcover). In : 1.

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.

Ginsberg, A., & Venkatraman, N. 1992. Investing in new information technology: The role of competitive posture and issue diagnosis. ***Strategic Management Journal***, 13: 37-53.

Gordon Walker, Kogut, B., & Shan, W. 1997. Social capital, structural holes and the formation of an industry network. ***Organization Science***, 8(2): 109-125.

Granovetter, M. 1985. Economic action and social structure: The problem of embeddedness. ***The American Journal of Sociology***, 91(3): 481-510.

Gulati, R. 1995a. Does familiarity breed trust? the implications of repeated ties for contractual choice in alliances. ***The Academy of Management Journal***, 38(1): 85-112.

Gulati, R. 1995b. Social structure and alliance formation patterns: A longitudinal analysis. ***Administrative Science Quarterly***, 40(4): 619-652.Gulati, R. 1998. Alliances and networks. ***Strategic Management Journal***, 19(4): 293.

Gulati, R. 1999. Network location and learning: The influence of network resources and firm capabilities on alliance formation. ***Strategic Management Journal***, 20(5): 397-420.

Gulati, R., & Gargiulo, M. 1999. Where do interorganizational networks come from? ***The American Journal of Sociology***, 104(5): 1439-1493.

Katz, M. L., & Shapiro, C. 1994. Systems competition and network effects. ***Journal of Economic Perspective***s, 8(2): 93-115.

Li, S. X., & Rowley, T. J. 2002. Inertia and evaluation mechanisms in interorganizational partner selection: Syndicate formation among U.S. investment banks. ***Academy of Management Journa***l, 45(6): 1104-1119.

Marsden, P. V. 1981. Introducing influence processes into a system of collective decisions. ***The American Journal of Sociology***, 86(6): 1203-1235.

McFadden, D. 1986. The choice theory approach to market research. ***Marketing Science***, 5(4, Special Issue on Consumer Choice Models): 275-297.

Meyer, M. H., & Seliger, R. 1998. Product platforms in software development. ***Sloan Management Review***, 40(1): 61-74.

Nair, H., Chintagunta, P., & Dubé, J. 2004. Empirical analysis of indirect network effects in the market for personal digital assistants. ***Quantitative Marketing & Economic***s, 2(1): 23-58

Nelson, R. R., & Winter, S. G. 1982. The schumpeterian tradeoff revisited. ***American Economic Revie***w, 72(1): 114.

Nobeoka, K., & Cusumano, M. A. 1997. Multiproject strategy and sales growth: the benefits of rapid design transfer in new product development. ***Strategic Management Journal***, 18(3): 169-186.

Oliver, C. 1990. Determinants of interorganizational relationships: Integration and future directions. ***Academy of Management Revie***w, 15(2): 241-265.

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

Podolny, J. M. 1994. Market uncertainty and the social character of economic exchange. ***Administrative Science Quarterl***y, 39(3): 458-483.

Podolny, J. M. 2001. Networks as the pipes and prisms of the market. ***The American Journal of Sociology***, 107(1): 33-60.

Powell, W. W., White, D. R., Koput, K. W., & Jason Owen-Smith. 2005. Network dynamics and field evolution: The growth of interorganizational collaboration in the life sciences. ***The American Journal of Sociology***, 110(4): 1132-1205.

Shapiro, C., & Varian, H. R. 1998. Information rules: A strategic guide to the network economy (hardcover). In : 1.

Shipilov, A., Rowley, T., & Aharonson, B. 2006. When do networks matter? A study of tie formation and decay. In ***Advances in Strategic Management***, vol. 23: 481-519Emerald.

Stieglitz, N. 2002. ***Industry dynamics and types of market convergence: The evolution of the handheld computer market in the 1990s and beyond.*** Paper presented at Druid Summer Conference Proceedings, Copenhagen.

Tanriverdi, H., & Chi-Hyon Lee. 2008. Within-industry diversification and firm performance in the presence of network externalities: Evidence from the software industry. ***Academy of Management Journa***l, 51(2): 381-397.

Toby E. Stuart. 2000b. Interorganizational alliances and the performance of firms: A study of growth and innovation rates in a high-technology industry. ***Strategic Management Journal***, 21(8): 791-811.

Uzzi, B. 1996. The sources and consequences of embeddedness for the economic performance of organizations: The network effect. ***American Sociological Review***, 61(4): 674-698.

Uzzi, B. 1997. Social structure and competition in interfirm networks: The paradox of embeddedness. ***Administrative Science Quarterly***, 42(1): 35-67.

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

Von Hippel, E., & Katz, R. 2002. Shifting innovation to users via toolkits. ***Management Science***, 48(7): 821-833.

West, J. 2003. How open is open enough?: Melding proprietary and open source platform strategies. ***Research Policy***, 32(7): 1259-1285.White, H. 1981. Where do markets come from? ***American Journal of Sociology***, 81(4): 730-779.Yoffie, D. B. 1997. ***Competing in the age of digital convergence***. Harvard Business School Press.

Zaheer, A., & Bell, G. G. 2005. Benefiting from network position: Firm capabilities, structural holes, and performance. ***Strategic Management Journal***, 26(9): 809-825.