

## **NOTE TO USERS**

**This reproduction is the best copy available.**

UMI<sup>®</sup>



BOSTON UNIVERSITY  
SCHOOL OF MANAGEMENT

Dissertation

**THE DYNAMICS OF PLATFORM-BASED COMPETITION  
DURING PERIODS OF ARCHITECTURAL SHIFTS**

by

**ARATI SRINIVASAN**

B.S., Anna University, Chennai, India, 1999  
M.S., M.B.A, Georgia Institute of Technology, Atlanta, 2004

Submitted in partial fulfillment of the  
requirements for the degree of  
Doctor of Business Administration

2010

UMI Number: 3430425

All rights reserved

**INFORMATION TO ALL USERS**

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3430425

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC

789 East Eisenhower Parkway

P.O. Box 1346

Ann Arbor, MI 48106-1346

Approved by

First Reader



---

N. Venkatraman, PhD  
David J. McGrath, Jr. Professor in Management  
Boston University School of Management  
Chair of Dissertation Committee

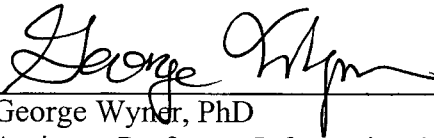
Second Reader



---

Fernando Suarez, PhD  
Associate Professor/Chair, Strategy and Innovation  
Boston University School of Management

Third Reader



---

George Wyner, PhD  
Assistant Professor, Information Systems Department  
Boston University School of Management

Fourth Reader



---

Shuba Srinivasan, PhD  
Associate Professor, Marketing Department  
Boston University School of Management

## ACKNOWLEDGEMENTS

This dissertation would not have been completed without the generous support and guidance of many people. I am especially grateful to my advisor, N. Venkatraman for his constant encouragement, supervision and support throughout the course of my doctoral studies. He always patiently offered his expertise and criticisms and encouraged me to focus on the broader and important implications of my research and communicate my ideas effectively.

I would also like to express my gratitude to the other members of my committee: Professor Fernando Suarez, Professor George Wyner and Professor Shuba Srinivasan. Professor Fernando Suarez generously provided his expertise and substantive critiques of my research. I also wish to acknowledge Professor Wyner's invaluable guidance in helping me bridge the gap between the technical and managerial content of my thesis and in developing the critical measures for my research model. Professor Shuba Srinivasan was a great help in honing my empirical research skills – she provided invaluable input into the empirical models used in my thesis and assessing their rigorousness.

I have also been fortunate to have had tremendous support and encouragement of many others to help shape this dissertation. I owe my thanks to all the faculty members in the Information Systems and Strategy and Innovation departments especially Professors Paul Carlile, Stine Grodal and Samina Karim for their feedback and helpful comments as I continued to develop my research. I am also deeply grateful to my fellow doctoral students who made this journey particularly special and fruitful.

On a personal level, I would like to thank my sister Sunita and my friends Anahita, Meeta and Shivanshu for being a great support system and always generous with their encouragement. I also cannot thank my parents enough because whatever I am today is because of their unconditional love and belief in me. I would also like to thank my dear son, Aneesh who is the light of my life and the greatest blessing I could have asked for. His wonderful smiles and warm hugs have helped me get through the most difficult of times. Finally, I am ever grateful to my husband Rajit without whose love, patience and forbearance this dissertation would never have been complete.

**THE DYNAMICS OF PLATFORM-BASED COMPETITION  
DURING PERIODS OF ARCHITECTURAL SHIFTS**

(Order No. )

**ARATI SRINIVASAN**

Boston University School of Management, 2010

Major Professor: N. Venkatraman, David J. McGrath, Jr. Professor in Management

**ABSTRACT**

This dissertation is motivated by the question of how the convergence of disparate platform architectures over time drives the evolution of platform-complementor networks. Academic research on platform-based competition has so far focused primarily on the relationship between the availability of third-party complements and platform success and on the actions of platform providers to attract third-party complementors to stimulate indirect network effects. However, very few studies have examined the interactions between platform providers and third-party complementors and how these interactions evolve with time-varying architectural shifts.

The goal of this dissertation has thus been to contribute to our understanding of the evolution of platform-based competition in periods of architectural convergence. Specifically, this dissertation is a first step towards recognizing the time-varying changes in platform-based competition during periods of architectural shifts with a focus on complementors who support and shape the characteristics of platform-complementor network structure. This study also brings to the forefront the differences in the type of



support provided by complementors, namely: ported ties vs. unique ties and that this type of support influences the dynamics of platform-based competition. This dissertation, thus, specifically seeks to address the following questions: Do complementor network positions (embeddedness in platform-complementor networks) influence their patterns of support to platforms over and above that explained by firm specific attributes such as age and centrality? Finally, how do platform characteristics (platform dominance and platform relatedness) co-evolve with complementor attributes to influence the developers' choice of platform to support?

To answer these and related questions, I empirically examine the evolution of patterns of linkage between platform providers and complementors in the PC and video game industry as the sixth generation technology of video game consoles transitions to the seventh generation from January 2000-December 2008. This setting allows us to understand the role of architectural shifts that have driven convergence between PC and video game platforms such as the introduction of cross-boundary standards (such as DirectX Graphics API) and middleware (for example, XNA Games Studio) in influencing related shifts in platform-based competition.

Specifically, the findings revealed strong support for the relationship between developer network positions (embeddedness) and patterns of tie formation. Further, developers were most likely to port titles to platforms that were architecturally similar to their previously supported platforms. The results of this study also underscored the role of architectural shifts driving convergence by demonstrating that as platform

architectures converged, the impact of developer embeddedness on patterns of linkage decreased. The theoretical and empirical insights of this study contribute to our understanding of how platform providers and third-party developers can respond effectively to architectural shifts that drive convergence. Specifically, this dissertation 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.

## TABLE OF CONTENTS

LIST OF TABLES .....	xii
LIST OF FIGURES .....	xiv
I. INTRODUCTION.....	1
1.1. Platform-based settings.....	1
1.2. Motivation .....	3
1.3. Research Overview and Research Questions .....	5
1.4. Research Contributions.....	7
1.5. Dissertation Organization .....	8
II. BACKGROUND .....	9
2.1. Current Research on Platform-based Competition .....	9
2.2. Recent Trends: Architectural Shifts towards Convergence across Platforms.....	12
2.2.1. Conceptual Specification of Drivers of Architectural Shifts towards Convergence in Platform Settings.....	14
2.2.2. Platform-based Competition during Periods of Architectural Shifts .....	15
2.3. Research Setting: Architectural Shifts driving Convergence in PC and Video Game Platforms.....	18
2.4. Network Evolution during Periods of Architectural Shifts .....	23
2.4.1. Patterns of complementor support with platform architecture convergence ..	23
III. LITERATURE REVIEW .....	28
3.1. Network-based view .....	28

3.1.1. The Influence of Network Structure on Performance Outcomes.....	30
3.1.2. The Influence of Network Structure on Tie Formation .....	33
3.1.3. Network Concepts in the Context of Platform-Complementor Networks.....	37
IV. THEORY AND HYPOTHESES.....	40
4.1. Developer Attributes and Platform Support through Ported Ties .....	40
4.1.1. Developer Embeddedness.....	40
4.2. Platform Attributes and Support through Ported Ties .....	42
4.2.1. Platform Centrality .....	42
4.2.2 Platform Porting Centrality .....	44
4.2.3. Platform Architectural Distance .....	46
4.2.4. Platform Genre Distance.....	48
V. METHODS.....	51
5.1. Research Setting .....	51
5.2. Data.....	52
5.3. Analysis 1 .....	53
5.3.1. Construct Operationalization.....	53
5.3.2. Nested Logit Model for Support through Ported Ties vs. Unique Ties.....	58
5.4. Analysis 2 .....	63
5.4.1. Construct Operationalization.....	63
5.4.2. Nested Logit Model for Support through Ported Ties during Transition from Sixth to Seventh Generation Consoles.....	68
5.5 Analysis 3 .....	70

5.5.1. Construct Operationalization.....	70
5.5.2. Nested Logit Model for Support through Ported Ties with firm level nests...	75
5.6. Measuring Architectural Distance between Platforms .....	77
5.6.1. Platform Architectures.....	77
5.6.2. Porting across Platforms .....	79
5.6.3. Hardware and processor architecture differences.....	80
5.6.4. OS differences.....	82
5.6.5. Development environment and API differences .....	83
5.6.6. Platform Architecture Distance Measure .....	85
VI. RESULTS.....	96
6.1 Results of Analysis 1 .....	96
6.2. Results of Analysis 2.....	102
6.3 Results of Analysis 3 .....	108
VII. DISCUSSIONS.....	113
7.1. Developer attributes and adaptation and support to platforms.....	116
7.2. Platform architecture convergence and platform-complementor network evolution at the macro level.....	122
7.3. Architectural Shifts and Dynamics of Platform-Complementor Network Interactions: Platform Firm Level Effects .....	126
7.3.1. Sony and the evolution of its network of complementors.....	128
7.3.2. Nintendo and the evolution of its network of complementors.....	130
7.3.3. Microsoft and the evolution of its network of complementors.....	131

7.4. Consequences of Architectural Shifts for Third-Party Developers: A Case Analysis of Electronic Arts and Valve Software.....	134
7.4.1. Responses to Architectural Shifts by Electronic Arts and Valve Software ..	135
VIII. CONCLUSIONS.....	140
8.1. Contributions .....	141
8.2. Limitations and Future Work.....	147
IX. APPENDICES .....	151
9.1. Appendix A: Results for Analysis 1 with Alternative Weighting Schemes.....	151
BIBLIOGRAPHY .....	152
CURRICULUM VITAE .....	165

## LIST OF TABLES

Table 1. Variables in Statistical Analysis for Analysis 1.....	58
Table 2. Summary Statistics for Analysis 1.....	62
Table 3. Variables in Statistical Analysis for Analysis 2.....	67
Table 4. Summary Statistics for Analysis 2.....	69
Table 5. Variables in Statistical Analysis for Analysis 3.....	74
Table 6. Summary Statistics for Analysis 3.....	76
Table 7. Platform Hardware and OS Standards.....	83
Table 8. Platform APIs and Development Environment.....	85
Table 9. Platform API and Dev. Environment Distance ( $API_{jk} = 0$ or $1$ ).....	89
Table 10. Platform OS and Hardware Distance ( $Hardware_{jk} = 0$ or $1$ ).....	90
Table 11. Total Platform Architecture Distance ( $Hardware_{jk}$ and $API_{jk} = 0$ or $1$ ).....	91
Table 12. Platform API and Dev. Environment Distance ( $API_{jk} = 0, 0.5$ or $1$ ).....	92
Table 13. Platform OS and Hardware Distance ( $Hardware_{jk} = 0, 0.5$ or $1$ ).....	93
Table 14. Total Platform Architecture Distance ( $Hardware_{jk}$ and $API_{jk} = 0, 0.5$ or $1$ ).....	94
Table 15. Results for Analysis 1.....	102
Table 16. Results of Case Specific (Generation specific) Variables for Analysis 2.....	106
Table 17. Results of Platform-specific variables for Analysis 2.....	107
Table 18. Results of case-specific variables for Analysis 3.....	111
Table 19. Results for Platform Specific Variables for Analysis 3.....	112
Table 20. Patterns of Porting during Transition from Sixth to Seventh Generation.....	122
Table 21. Firm Level Patterns of Porting.....	128

Table 22a. Evolution of Patterns of Support for Electronic Arts.....	136
Table 22b. Evolution of Patterns of Support for Valve Software.....	138
Table 23. Results of Sensitivity Analysis for Analysis 1 with Alternative Architecture Distance Weights .....	151



## LIST OF FIGURES

Figure 1a. Architectural Shifts towards Convergence in the PC and Video Game Sectors .....	19
Figure 1b. Architectural convergence between PC and video game consoles .....	23
Figure 2. Patterns of support by cross-platform developers .....	25
Figure 3. Patterns of support by exclusive console developers .....	25
Figure 4. Patterns of porting by cross-platform developers.....	26
Figure 5. Conceptual Research Model.....	50
Figure 6. Nested logit model for developer support through porting vs. unique tie.....	62
Figure 7. Nested logit model for developer support through ported ties during transition from sixth to seventh generation.....	69
Figure 8. Nested logit model for developer support through ported ties for firm level nests .....	76
Figure 9. Software Stack for Gaming Platforms.....	78
Figure 10. Architectural Distance between Platforms .....	95
Figure 11. Architectural Convergence in PC, Video Game and Mobile Platforms.....	126

## I. INTRODUCTION

### 1.1. Platform-based settings

In many platform-based settings such as video games, mobile telephony, and packaged software the actions of firms are not autonomous but need to be coordinated with complementors. Under such settings, individual firms need to continually refine and modify their routines to align with constantly shifting architecture of business systems—which themselves are shaped by the actions of other players. For example, individual video game software developers launch their games based on their resources and competencies (firm-specific drivers) but also launch games reflecting the dynamic characteristics of video game platforms and competitive actions of other developers (Venkatraman & Lee, 2004). Moreover, under such settings since customers purchase components that subscribe to the same system architecture (Baldwin & Clark, 1997), patterns of coordination between hardware and software are central to crafting successful strategies (Evans, Hagiu, & Schmalensee, 2008).

Relationships with complementors, although implicit, provide resources critical to the success of platforms due to inherent mutual dependency between the parties within the system (Venkatraman & Lee, 2004). Platform providers invest significant resources to attract developers to their platform and in turn, third-party developers commit resources to develop software for a platform over time. Platform providers attract developers to their platforms not just by creating hardware of superior quality but also by

providing them with toolkits that simplify the software development process. Such toolkits in addition to providing basic training to the developers also contain libraries of commonly used modules that the developer can incorporate into their design (Von Hippel & Katz, 2002). The software developers, on the other hand, invest significant resources in understanding the toolkits and customizing their applications to a specific platform. As a result, platforms that are able to derive the maximum benefits from such relationships are more likely to achieve dominant positions than those that do not.

The relationship between platform owners and complementors becomes even more pronounced as platform technologies evolve rapidly and newer platform generations are launched at regular intervals. The rapid rate of technological evolution implies that third-party complementors need to continuously adapt and learn the new routines and knowledge associated with the newer platform technologies.

Besides rapid technological advances across single platform technologies, third-party developers also face challenges with respect to contemporaneously supporting platforms that transcend traditionally defined platform boundaries. Technological and market shifts such as improvements in hardware, the emergence of common third-party standards and the overlap in functionality across platforms have driven a trend towards convergence across platforms. In particular, the role played by architectural shifts that drive convergence across distinct platforms has been significant in altering the dynamics of platform-based competition. These architectural shifts have resulted in the morphing and converging of previously disparate platform technologies as diverse as PCs, video game consoles, handheld devices, and smartphones and e-book readers in terms of their

functionality and portfolio of available applications. As a result, there is increasing pressure on third-party developers to adapt across these new platform settings. Likewise, platform providers in order to compete effectively also seek to extend their reach to new platform markets that allows them to leverage their existing competencies and complementor networks.

## **1.2. Motivation**

Despite the strong mutual dependency between platform providers and complementors for success, relatively little research on platform-based competition has focused on platform-complementor interactions. Typically, the focus of extant research has been on the actions of platform providers and particularly on how they can attract third-party complementors to stimulate indirect network effects (Gawer & Cusumano, 2002; Gawer & Henderson, 2007; Eisenmann, 2008; Evans, Hagiu, & Schmalensee, 2008). For instance, one stream of research has focused on the strategies of platform owners and how their decisions regarding the optimal openness of their platforms influences their ability to attract third-party complementors and the ensuing competitive dynamics (West, 2003; Parker & Van Alstyne, 2005; Eisenmann, Parker, & Van Alstyne, 2008; Lee & Mendelson, 2008). Another related but separate stream of research has drawn on network-based theories to understand the role of platform network positions in influencing developers' choice to support a platform (Venkatraman & Lee, 2004).

However, presently there is little work examining the perspective of complementors - no study has so far attempted to reconcile the fact that there is heterogeneity in complementor attributes and experiences and hence their ability to leverage the resources

provided by platform sponsors to support multiple contemporaneously existing platforms. In addition, there have also been no studies that recognize that there is also heterogeneity in how developers choose to support platforms and that the choice of support has important implications for the competitive dynamics of platform-based settings. 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).

Additionally, most research on platform-based competition has adopted a static or cross-sectional view and has not focused on how platform-complementor interactions evolve dynamically over time. The role of technological evolution, the introduction of new platform architectures and the emergence of new cross-boundary standards that present opportunities for developers to migrate quickly to new settings in altering platform-complementor relationships have not been studied. Specifically, the question of why there is still heterogeneity in the rate at which complementors adapt to new settings (in terms of which complementors migrate, the rate at which they migrate and where they migrate to) even as cross-platform standards emerge and platform architectures converge has not been examined.

Finally, another limitation of extant research on platform-based competition is that it has typically focused on platform-complementor relations and their performance implications in a single setting. For instance, Venkatraman & Lee (2004) and Afuah & Werner (2007) study the platform-complementor interactions in the video game industry. However, such a view assumes that platforms and complementors operate in stable

industry structures and are not affected by technology changes that transcend industry boundaries. I argue that such an assumption does not always hold true in platform-based settings like the PC, video game and mobile platform sectors, which are becoming increasingly convergent and exist contemporaneously as gaming platforms. Therefore, additional research on strategic actions under dynamic evolution of platform-based competition is needed that acknowledges convergence across industry boundaries that causes competing systems exist contemporaneously and also calls for complementors to continually re-evaluate their strategies and decisions to support a platform.

### **1.3. Research Overview and Research Questions**

This dissertation is, therefore, a first step towards adopting a more holistic, dynamic approach that recognizes the role of developer network positions, and platform architectures and network positions during periods of architectural shifts in explaining how and why a developer chooses to support a platform. I specifically seek to address the following questions: One, how does developer heterogeneity influence the mode of support extended to platforms? Specifically, this study focuses on understanding how developer network attributes (extent of embeddedness in a platform) influence their likelihood of support to a platform through ported or unique game titles above and beyond that explained by a firm's resource positions? This dissertation also focuses specifically on the distinction between support through ported and unique game titles due to their crucial role in driving the outcomes of platform-based competition. Recent research on indirect network effects has demonstrated that platform dominance is driven largely by the extent to which a platform is supported by exclusive applications (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.

The second question that this dissertation seeks to address is: how do platform attributes influence which platform a developer chooses to support through ported titles? In particular, I am interested in exploring how platform network positions (platform centrality and platform porting centrality) and platform architectural and genre distance influence developers' choice of platform to support through ported titles. Finally, I also address the question of how platform architectural strategies co-evolve with developer network positions to influence developers' choice of support to a platform. These questions lie at the intersection of different strands of research: platform-based competition (Shapiro & Varian, 1998); resource-based view and firm capabilities (Tushman & Anderson, 1986; Klepper, 1996; Sørensen & Stuart, 2000) and network perspectives of organizational evolution and growth (Gulati, Nohria, & Zaheer, 2000; Powell, White, Koput, & Owen-Smith, 2005). By adopting the perspective of these three distinct streams of research, our study recognizes a) the importance of the interaction between platforms and complementors that is crucial for success under platform-based settings, and b) that these networks of relationships are source knowledge about partners (Ahuja, 2000a) and therefore need to evolve in the face of technological change.

To answer these general questions, this dissertation studies 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. I choose the PC and video game platforms and their ecosystem of third-party game developers as the setting of this study for the following reasons: in addition to exhibiting strong indirect network effects and hence being a classic example of platform-based competition, the PC and video game consoles are becoming increasingly convergent and substitutable as gaming platforms. For instance, the launch of cross-industry standards such as the XNA platform by Microsoft has resulted in increased overlap in developers and games between PC and video game consoles. Also since the launches of video game consoles clearly fall into generations based on technology improvements, looking at platform-complementor evolutions across the PC and two distinct video game generations enables us to understand the impact of technology changes on complementor choices and hence, platform-complementor network evolution.

#### **1.4. Research Contributions**

In summary, this dissertation makes the following contributions: One, by focusing on complementor attributes coupled with platform characteristics and their impact on the complementors' choice of platform and mode of support, this study seeks to extend research on platform-based competition that has so far mostly adopted the platform providers' perspective. Two, prior studies on understanding complementors' choice of a platform, have not differentiated between the type of support extended by complementors to platforms (for example, Venkatraman & Lee, 2004). This dissertation extends such research by recognizing that the mode of support extended by developers' influences competitive dynamics in platform-based settings – as a result I develop a more nuanced conceptualization of the form of support extended by developers. This study, specifically,



distinguishes between support of a platform through unique ties versus ported ties and studies the likelihood of support to a platform through ported ties. Three, this study adopts a longitudinal perspective and studies the dynamics of platform-based competition during periods of technology change. This allows us to understand how even in periods of technology change and architectural convergence there is still heterogeneity in the adaptation by complementary developers.

### **1.5. Dissertation Organization**

The rest of the dissertation is organized as follows: In the next section (Chapter 2), I present an overview of the extant research on platform-based competition, the role of architectural shifts in altering the dynamics of platform-based competition and finally, introduce the research setting used in the study. Chapter 3 reviews the literature on the resource-based views and network views that form the conceptual anchors of this dissertation. Chapter 4 presents the theory and hypotheses. Chapter 5 includes the research methods while Chapter 6 presents the results. Finally, chapter 7 presents the discussions and chapter 8 presents the conclusions including the contributions, limitations and future areas of extension.

## II. BACKGROUND

### **2.1. Current Research on Platform-based Competition**

Platform-based technologies such as video games, PDAs and personal computers consist of a core technology platform and interchangeable complementary applications built upon it (Evans, Hagiu, & Schmalensee, 2008). Under such settings, no single company has the resources to meet heterogeneous consumer demands (Baldwin & Clark, 1997; Shapiro & Varian, 1998; Garud, Jain, & Kumaraswamy, 2002) and platform owners seek to encourage complementary third-party innovation to increase the platform's appeal to the end consumer (Bresnahan & Greenstein, 1999; Evans, Hagiu, & Schmalensee, 2008). In platform markets indirect network effects, thus, operate where value of a platform accrues due to the availability of complements.

A review of the extant research on platform-based competition reveals that most studies broadly fall into two categories: 1.) a focus on the value of indirect network effects to platform owners by economists and marketing researchers and 2.) a focus on platform owner strategies to attract third-party complementors by technology management researchers.

Economists and marketing researchers have sought to understand the impact of indirect network effects – specifically the impact of the presence of a large number of complements on hardware sales in a number of industries. For instance, Gandal, Kende, & Rob (2000) in their study of the compact disc industry analyzed the impact of the number of CD titles available on the diffusion of CD players. Similarly, Nair,

Chintagunta, & Dubé (2004) used the PDA industry as their setting and modeled the relationship between the number of software titles available for different PDAs and PDA sales. In the video game industry, Clements & Ohashi (2005) studied the impact of the number of game titles available on video game console sales. More recently, studies on understanding the impact of indirect network effects on platform sales have begun to recognize that advantages may not accrue only from the number of complements available and have hence, begun to focus on more nuanced conceptualizations of indirect network effects such as number of exclusive complementary software titles available for a platform (Corts & Lederman, 2009).

More recently, a growing body of literature within the technology management stream has begun to focus on the actions of platform owners and specifically on how they can attract third-party complementors to their platforms to stimulate indirect network effects (Gawer & Cusumano, 2002; West, 2003; Eisenmann, 2008). Gawer & Cusumano (2002) through a set of case studies on Microsoft, Intel, Cisco, Palm and NTT DoCoMo developed a framework of how platform firms can build their complementor ecosystem to attain platform leadership. Similarly, West (2003) examined how the extent of creating open platform architectures as opposed to maintaining proprietary or closed platforms influenced platform success. Eisenmann, Parker, & Van Alstyne (2008) also examined factors that favor proprietary versus shared platforms and how management challenges differ for each of these strategies.

However, thus far most studies on platform-based competition adopting a platform provider perspective have been limited to conceptual theorizing or useful case

studies. Moreover, few studies have addressed complementor perspectives and the ability of complementors to respond to technological change. While Venkatraman & Lee (2004) take a first step in this direction and study how the preferential linkage decisions of third-party developers evolve as a function of platform attributes, they do not study how the patterns of linkages vary with developer attributes and also how these patterns of linkage evolve as disparate platform domains converge.

This dissertation seeks to extend prior research on platform-based competition by building on the fundamental assertion that the dynamics of platform-complementor interactions are continually evolving in response to technology evolution and architectural shifts. Architectural shifts that drive convergence across platform boundaries have important implications for the dynamics of platform-based competition. As platform providers enter new but related platform markets, they seek opportunities to translate their competencies to the new platform domain. As a result, they are likely to bring in their existing standards and routines to new settings in order to leverage their existing complementor network. For instance, when Microsoft entered the video game console market, in order to leverage their 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 the Wii and PS3 had to alter their architectural strategies to compete with Xbox360 and its large ecosystem of PC complementors. Similarly, the ability of Apple to use its developer network in the mobile platform market by adapting its existing Mac

standards and toolkits such as XCode to the iPhone platform meant that existing mobile platform provider such as Symbian and Microsoft Windows Mobile had to rapidly alter their architectural strategies. 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 developers.

The goal of this dissertation, thus, is to contribute to the research on platform-based competition by adopting the perspective of third-party complementors and understanding why and how they respond to architectural shifts that drive convergence across different yet overlapping platform domains. In order to understand how the dynamics of platform-complementor interactions evolve in domains characterized by converging platform architectures, in the following section I introduce the concept of architectural shifts towards convergence and develop a conceptual specification of the factors that drive convergence across platform boundaries.

## **2.2. Recent Trends: Architectural Shifts towards Convergence across Platforms**

The notion of convergence has been around for decades and in the broadest sense is defined as 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 view point, 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 players in the PC industry extending their reach into newer markets to reduce risks associated with substitution and obsolescence. Microsoft released the Xbox console in late 2001 to compete in the video game platform space. It also strove to translate its competencies and developer network in the mobile platform space by launching the Windows Mobile operating system 2003. Likewise, Apple was able to leverage its developer networks to succeed in the mobile space. Other examples of convergence include increased overlap among handheld devices such as PDAs, smartphones and netbooks. For instance, platform providers in the PDA and smartphone space such as RIM, Palm, Google and Apple have attempted to compete in the netbooks market. The possibility of increasing their reach into the netbooks space has not only been enabled by their ability to put their smartphone chips and operating systems into these devices but also by the ability to use their own standards and third-party standards such as OpenGL to enable third-party software development.

These examples clearly highlight the role of technology shifts in driving convergence across platform settings. Of primary interest in this dissertation are those technological shifts that make architectural convergence possible and expand the basis of competition in platform markets. The following section develops a conceptual specification of the technological shifts that drive convergence.

### **2.2.1. Conceptual Specification of Drivers of Architectural Shifts towards Convergence in Platform Settings**

*Advances in Hardware:* Advances in hardware and the architectures of the core product or platform have to a large extent driven convergence across platforms. Moore's law (Moore 1965) and continuous improvements in semiconductor technologies have increased the market demand for convergence. For instance, improvements in processing power, memory management and graphics imply an increase in the capabilities of devices such as netbooks, smartphones and PDAs. Moreover, like PCs, these devices come equipped with operating systems and are hence able to offer many features of a computer. For instance, netbooks like ASUS Eee PCs include PC like processors (Intel Atom) and Windows Operating Systems and are hence able to satisfy the same user needs as PCs.

*Launch of third-party standards:* The availability of common third-party standards has further increased the rate at which convergence occurs across platforms. For instance, the cross-language, cross-platform graphics API – the OpenGL eases the process of application development for developers across different platform settings. As a result, large number of platform owners such as Nintendo, Sony, and Apple support the

OpenGL API on their platforms to attract third-party application development. Similarly, the increased availability of third-party middleware such as the Havok and PhysX game engines allows developers to support multiple platforms simultaneously. More recently, firms such as Google and Symbian have adopted dominant standards like Linux to attract a larger ecosystem of developers from across multiple platform settings.

***Launch of cross-platform standards by platform owner:*** The third shift that has driven convergence is the launch of common standards and APIs by a single platform firm. In such situations platform firms vie for control across the value chain and their ecosystem of complementors (Utterback & Suárez, 1993). As a result, they launch standards and interfaces that can be reused across their entire range of platforms. For example, Apple has made its operating systems compatible across all its devices and hence is poised to meet the challenge for convergence. Similarly, Microsoft supports the proprietary DirectX API across its PC, video game and mobile platforms. By bringing in the same standards to work across a large portfolio of devices, platform firms such as Apple and Microsoft seek to extend their reach and control across related yet disparate platform markets. Such strategies that drive convergence are in line with the argument by Gawer & Cusumano (2002) that “since platforms are made of components that interact following standard interfaces, standard wars are necessarily a part of platform strategies.”

### **2.2.2. Platform-based Competition during Periods of Architectural Shifts**

Architectural shifts that drive convergence across platform boundaries 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 accelerate the blurring of 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. As a result, platform owners strive to continually evaluate their architectural strategies to enable them to attract third-party developers. Such strategies include evaluating the extent of openness of a platform and whether to provide support to developers by using proprietary or third-party APIs and standards. For instance, while Amazon launched its eBook reader - the Kindle - as a closed platform, competition from the newly launched Apple iPad forced it to enhance the value of the Kindle through third-party applications. As a result, Amazon released its Kindle Development Kit to attract third-party developers. For instance, Amazon says Handmark is building a Zagat guide for restaurant reviews and ratings, and Sonic Boom is building word and puzzle games. EA Mobile is also developing games for the Kindle.

Convergence also has important consequences for third-party developers that support platforms. It presents an opportunity for developers to extend their reach and support multiple platforms that cross industry boundaries simultaneously. Moreover, in order to compete effectively, developers 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 developer attributes including their age and size and also on their prior experience with certain

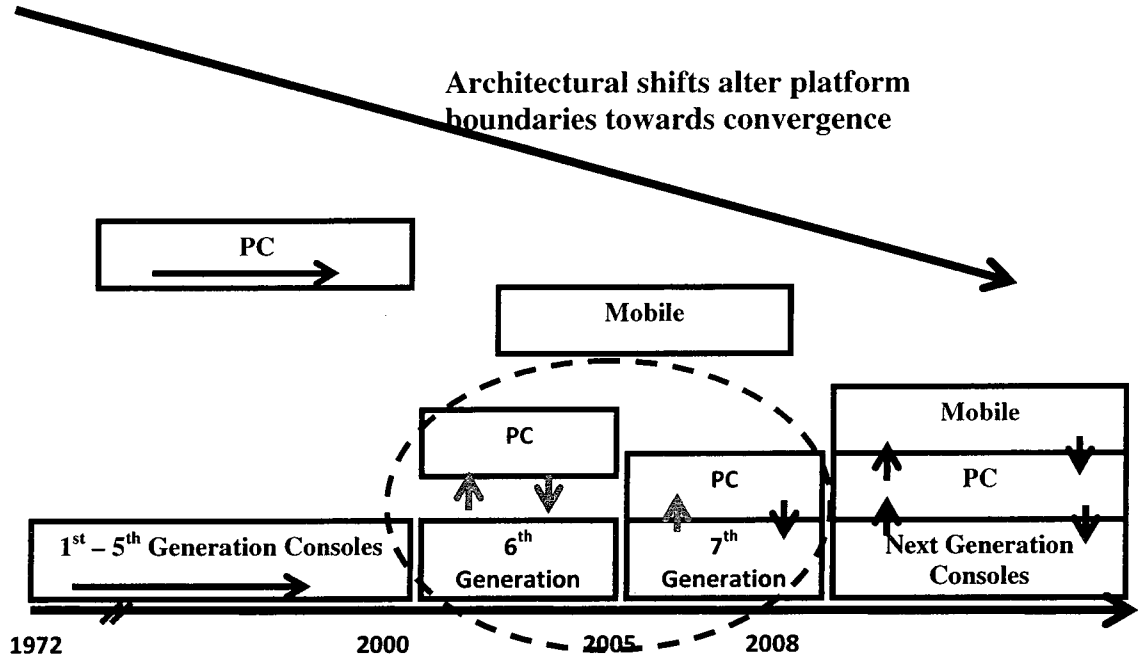
platform architectures. For instance, large developers such as Electronic Arts that support multiple platforms are able to adapt and migrate quickly to newer platforms while niche developers typically are slower to adapt and migrate to newer platform settings. Developer attributes in addition to influencing their ability to migrate also co-evolve with platform architectures to influence the patterns of support extended to platforms. 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. For instance, developers are more likely to launch game titles on platforms that are more similar to platforms they have supported in the past. They are also more likely to port their existing applications to newer platforms rather than launch exclusive titles across 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 dissertation uses the PC and video game sectors as the research setting.

## **2.3. Research Setting: Architectural Shifts driving Convergence in PC and Video**

### **Game Platforms**

The research setting for this dissertation is the PC and the sixth and seventh generation video game console sectors and their ecosystem of third-party game developers. Video game consoles have undergone significant technological advancement and these technological 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 1a 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. In the follow section the key drivers of the architectural shift towards convergence in the PC and video game sectors are discussed in detail.



**Figure 1a. Architectural Shifts towards Convergence in the PC and Video Game Sectors**

The key factors that have driven convergence between PCs and video games during the transition from the sixth to the seventh generation are discussed below:

**Hardware Advances:** The sixth generation of video game consoles began with the launch of Sega Dreamcast and Sony’s PlayStation 2 in October 2000. The Sony PlayStation 2 was touted as being fundamentally different than current PC architectures (Stokes, 2000). These differences required significant efforts on the part of the developer to design their code to work simultaneously on both platforms. The launch of the PS2 was followed by

the launch of the Nintendo GameCube and the entry of Microsoft into the video game consoles market with the launch of the Xbox console in November 2001. While, the GameCube also was a radically different architecture than the PC, the Xbox was a PC turned game console (Shimpi, 2001) as it was based on technology from the personal computer industry (Gallagher & Park, 2002). The Xbox's 32-bit x86 architecture was equivalent to current PC architectures and its GPU was also based on the GPU of PCs. As a result, the launch of the Xbox marked the beginning of the convergence between the PC and sixth generation consoles. The seventh generation of consoles began with the launch of the Xbox360 by Microsoft in November 2005 and was followed by the launch of the Nintendo Wii and Sony PlayStation 3 consoles in November 2006. Seventh generation consoles were characterized by more PC-like architectures with higher definition graphics and processing power. Moreover, seventh-generation consoles came equipped with online gaming capabilities and hard drive storage for game data thereby, making them more similar to PCs in architecture. Simultaneously, this time period has also witnessed significant improvements in the PC architectures with faster graphics accelerators and improving CPU technologies. While early PCs were typically designed with mainly spreadsheets, word processors and other static processing applications in mind, recent PCs have undergone significant changes with the addition of special-purpose hardware like 3D accelerators and sound processors (Stokes, 2000). Simultaneous improvements in PC hardware implied that genres such as first person shooter games that were typically played either on PCs could now be played on both PCs and seventh generation consoles. Therefore, the hardware improvements in PC and

seventh generation consoles are the first architectural shift that drove convergence between these platforms.

***Launch of third-party standards:*** Another important architectural change that has driven convergence between PC and video game consoles is the greater availability of middleware. With increasing pressure to rewrite the same games across multiple platforms, there has been a surge in third-party game engines or middleware that support cross-platform game development between PCs and video game consoles (Reimer, 2005). For instance, common third-party physics engines such as PhysX and Havok are supported by both PCs and seventh generation consoles. In addition, companies such as GarageGames specialize in providing engines such as TorqueX that are built upon higher level languages such as Java and C# that assist in cross-platform game development. Besides the availability of cross-platform game engines or middleware, the availability of common third-party 3D graphics libraries such as the OpenGL has matured and knocked proprietary interfaces out of the market, thereby further easing the process of cross-platform game development.

***Launch of common cross-platform standards by Microsoft:*** The earliest sixth generation console, the PS2 in addition to having fundamentally different hardware architecture from the PC also had its own proprietary OS. Moreover, its SDKs and development environment were considerably distinct from the PC game development environment with no support for Windows. On the other hand, when Microsoft launched the Xbox, in its attempt to control the video game sector, it included in its SDK, the DirectX that were a standard set of APIs for tasks related to multimedia and already widely used for PC

application development. The use of common APIs by Windows and Xbox platforms further eased the process of cross-platform development. In addition, in order to leverage their expertise and their established network of game developers in the PC platform, Microsoft launched the XNA Game Studio. The XNA Game Studio is a set of tools allowing developers to build games for both the Microsoft Windows and Xbox 360. The ability to develop games simultaneously for PC and Xbox360 platforms has resulted in an increase in the amount of porting by developers. For instance, prior to the launch of the XNA platform, 31% of titles were ported between PC and video games. On the other hand after the launch of the XNA, 62.7% of game titles were ported between PC and video games. The launch of common standards by Microsoft can, thus, be considered a major architectural shift driving convergence in the PC and video game sectors.

Each of these architectural shifts has driven convergence between PCs and video game consoles as gaming platforms and has presented opportunities for developers to support these platforms with similar game titles contemporaneously. The factors driving the architectural shift towards convergence is demonstrated in Figure 1b. The impact of these shifts on the patterns of support is examined in the following section.

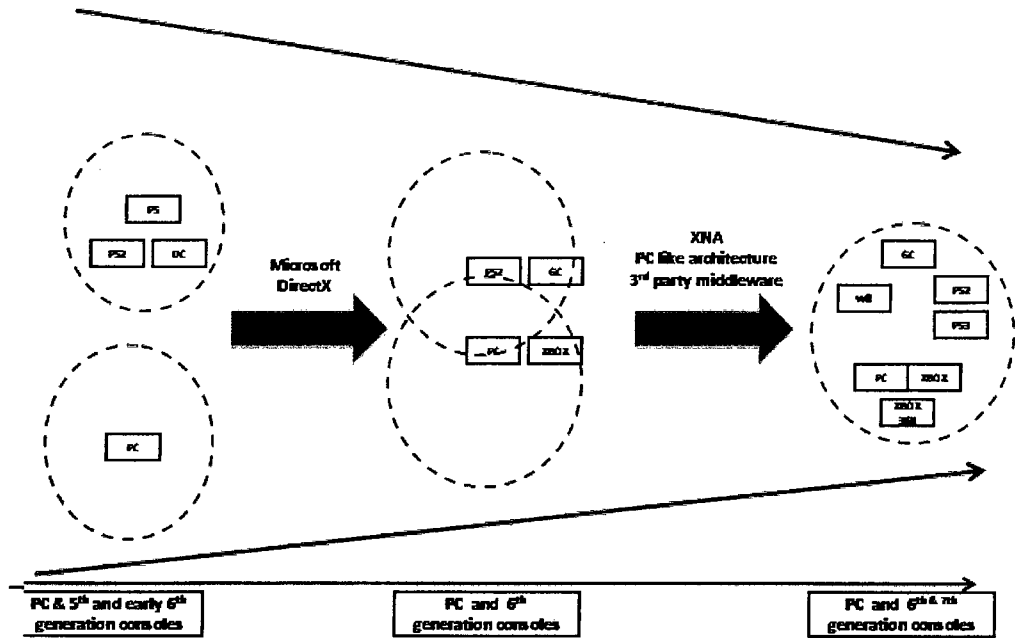


Figure 1b. Architectural convergence between PC and video game consoles

## 2.4. Network Evolution during Periods of Architectural Shifts

### 2.4.1. Patterns of complementor support with platform architecture convergence

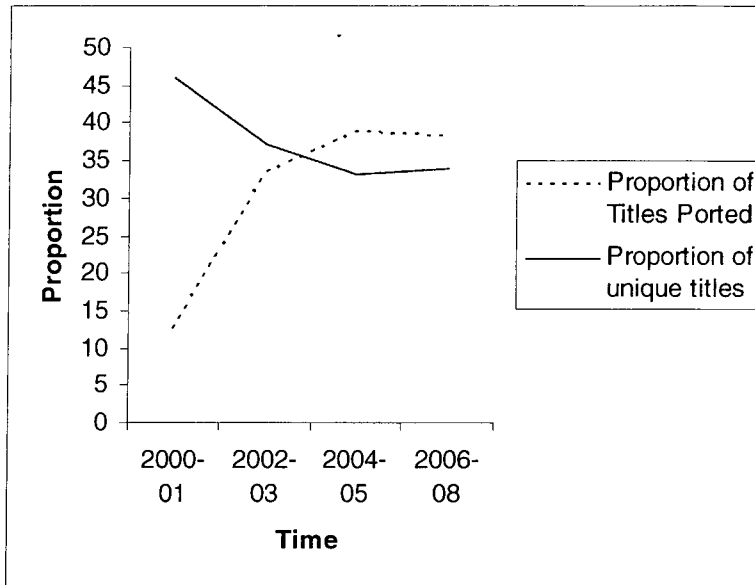
While the architectural shifts described in the previous section have provided opportunities for developers to re-use their knowledge and routines in new settings and thereby, support multiple competing platforms simultaneously, an analysis of the patterns of migration exhibited by third-party complementors revealed interesting insights.

A preliminary review of the patterns of network evolution as video game consoles transitioned from the sixth generation to the seventh generation showed that the extent of overlap in game titles between PCs and video game consoles increased considerably with

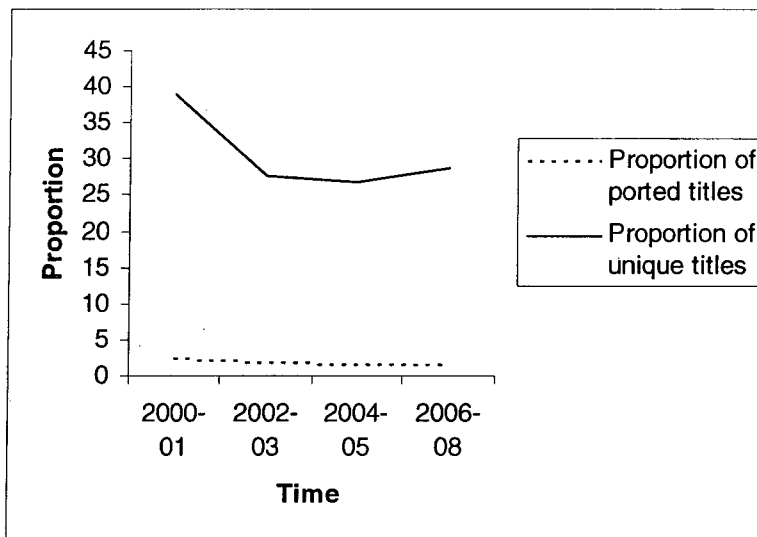


developers becoming more inclined to port their existing game titles across platforms. Specifically, the proportion of game titles ported between PCs and video game consoles increased from 12.7% at the end of 2001 to 38.23% at the end of 2008. Figure 2 demonstrates the patterns of support by cross-platform developers while figure 3 shows the evolution in the patterns of support by exclusive console developers. The figure depicting the patterns of support by exclusive console developers shows that the extent of ported titles went down significantly while the extent of unique titles remained fairly constant as consoles transitioned to the seventh generation indicating that exclusive console developers were more likely to be embedded in a single platform technology and hence unlikely to support platforms through ported titles. Moreover with convergence, these exclusive developers were also more likely to now migrate to PCs and hence port titles across PCs and consoles rather than exclusively across consoles.

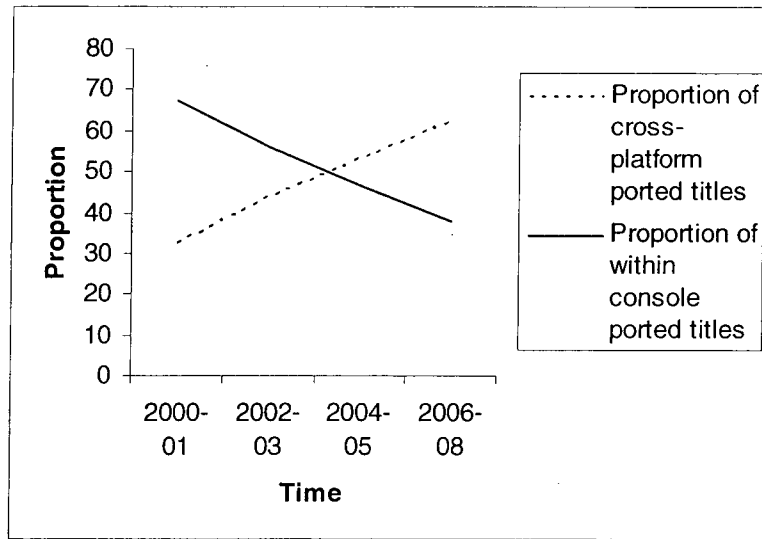
The analysis also revealed that cross-platform game developers who developed game titles for both PCs and video games mainly ported titles between existing sixth generation video game consoles (67.4% between video game consoles and 32.6% between PC and video game consoles in 2001). On the other hand, with the launch of seventh generation platforms, these developers were more likely to develop same game titles for PCs as well (62.2% between PC and video game consoles at the end of 2008). This once again highlights how convergence has altered the patterns of support extended by developers across the PC and video game consoles. Figure 4 demonstrates the time-varying changes in the patterns of porting exhibited by third-party cross-platform developers.



**Figure 2. Patterns of support by cross-platform developers**



**Figure 3. Patterns of support by exclusive console developers**



**Figure 4. Patterns of porting by cross-platform developers**

While this clearly demonstrates the role of architectural shifts in driving the dynamics of platform-complementor interactions in the PC and video game sectors, an analysis of the migration patterns of developers also revealed some important insights. The migration patterns of third-party developers between the PC and sixth and seventh generation consoles were then analyzed. Specifically, with the launch of the sixth generation approximately 15% developers migrated between the PC and video game industries. After the launch of the 7<sup>th</sup> generation platforms, however, 35% of developers migrated between PC and video game industries. While the increase in migration can be attributed to architectural convergence across industry boundaries, it is interesting to note that approximately 65% of third-party developers have not been able to adapt and take advantage of these architectural shifts to migrate to newer platform settings.

This clearly underscores the need to develop a holistic view of the patterns of support extended by developers to platforms by taking into consideration the

heterogeneity in developer attributes and their ability to leverage architectural shifts to enable strategically beneficial moves. Moreover, the impact of the developers attributes on the patterns of support extended to platforms needs to be examined in light of the platform attributes and architectural differences to further shed light into the logic behind the sequence of launches of game titles. Therefore, the goal of this dissertation is to understand why certain developers are able to adapt to architectural shifts and why some are not and also to understand the time varying changes in patterns of support driven by convergence. I, thus, use the network-based views as the conceptual anchors of this dissertation to explain how developer specific attributes and platform attributes together co-evolve to influence the patterns of support extended by developers to platforms in platform-complementor ecosystems.

### III. LITERATURE REVIEW

#### **3.1. Network-based view**

This dissertation has chosen the network-based view of the firm as the dominant conceptual anchor for investigating the patterns of support extended by third-party game complementors to the PC and video game platforms. It thus, uses complementors and platform provider network positions as the initial building block of the conceptual model to help answer the question: “Why and how do developers differentially respond to architectural shifts across platform boundaries?”

Resource-based theories (Penrose, 1959; Wernerfelt, 1984; Barney, 1986; Lippman, McCardle, & Rumelt, 1991; Amit & Schoemaker, 1993) have been used extensively in the strategic management literature to understand how firm heterogeneity, impacts its ability to adapt to change and performance outcomes. The heterogeneity derives from a firms’ bundle of idiosyncratic and hard to imitate resources and capabilities (Wernerfelt, 1984; Rumelt, 1991; Barney, 1996). While the resource-based view has primarily focused on how a firm’s resources and capabilities improve performance, another related stream has focused on the “liability” of firms’ resource positions. Within the context of such research, the focus is on the problem of strategic transformation by organizations (Aldrich & Auster, 1986).

*Strategic competitive response capability* (Helfat et al., 2007) refers to the ability of firms to respond to exogenous change by scanning the environment, identifying new opportunities, assessing its competitive position and responding to competitive strategic

moves. However, a large number of empirical studies have demonstrated that changes related to even minor technological shifts are often hard to respond to (Tushman & Anderson, 1986; Henderson & Clark, 1990). As a result, factors such as a firms' age and size and their impact on firms' ability to adapt their resources and established organizational routines to new settings has been examined. Within the context of platform-based settings, this study seeks to examine how the age and dominance of third-party complementors (which serves as a proxy of size) influences their ability to support competing platforms and their choice of the mode of support extended to platforms. While age and size serve to understand the influence of complementor specific attributes, they do provide any insight into how a complementors' prior experience influences its ability to support a platform. A network perspective allows us to understand how a complementors' experience in a prior platform, enables or constrains its ability to adapt to technological change and support newly emerging platforms.

Therefore, the network-based view extends the resource-based view by challenging the fundamental assumption that the impact of firm resources can be examined in isolation and recognizing that they are often a function of their relative network positions. The ability of a firm to modify its resources/routines depends to a large extent on its prior experiences, as Teece, Pisano, & Shuen (1997a) note: "Where a firm can go is a function of its current position and the paths ahead...The notion of path dependencies recognizes that history matters."

The notion of prior dependence is a fundamental assumption in network theory where the focus is on how network structure enables and constrains actions. As a result,

the last few years have witnessed an increase in research on inter-organizational networks and the role of network positions in the formation of ties (Gulati, 1995a; Dyer & Singh, 1998; Venkatraman & Lee, 2004; Powell, White, Koput, & Jason Owen-Smith, 2005) and in driving performance outcomes of actors (Podolny, Stuart, & Hannan, 1996; Powell, Koput, & Smith-Doerr, 1996; Ahuja, 2000a). The following sections review the two main areas that have been the focus of inter-organization networks:

### **3.1.1. The Influence of Network Structure on Performance Outcomes**

In the last decade, scholars have drawn on network theories to highlight the value of a firm's external resources available through its network of relationships (Gulati, 1999; McEvily & Marcus, 2005; Zaheer & Bell, 2005). The fundamental contention of the network perspective is that a firm's embeddedness in its network of relationships with other organizations has significant implications for firm performance (Gulati, Nohria, & Zaheer, 2000). Moreover, since access to resources, capabilities, and knowledge (Uzzi, 1996; Burt, 1997) are often acquired from networks of inter-firm ties, it is valuable to examine how network structure influences firm performance (Zaheer & Bell, 2005).

The structure of network linkages provides both opportunities and constraints on the actions of participants. The relational links between actors in a network serve as conduits for the flow of a broad variety of resources, in either the tangible form of money or specific skills or the intangible, but no less important, form of information, social support, or prestige (Powell, White, Koput, & Jason Owen-Smith, 2005). As a result a large body of research has documented the wide-ranging effects of network ties on the

behavior of both individuals and organizations (Knoke, 1990; Powell, Koput, & Smith-Doerr, 1996; Podolny & Page, 1998).

Studies have explored the impact of network structure on different aspects of firm outcomes. For instance, some studies have focused on how the location of a firm in its network of relationships influences the adoption of innovation and corporate strategies (Palmer, Jennings, & Zhou, 1993; Westphal, Gulati, & Shortell, 1997). Other studies have explored the impact of networks of relationships on financial performance. For instance, studies have found that firms with ties to elite or high-status partners experience considerable economic benefits such as superior profitability, rates of growth and survival (Oliver, 1990; Baum & Oliver, 1992; Podolny, 1993; Dyer & Singh, 1998). Studies have also focused on the relationship between embeddedness and firm performance. Most of these studies have found a positive impact of embeddedness (Uzzi & Gillespie, 2002; Echols & Tsai, 2005). Additionally, recent years have witnessed a surge in studies devoted to the relationship between structural holes, social capital and firm performance (Burt, 1997; Coleman, 1988). For instance, Burt's classic work (Burt, 1997) finds that firms occupying the position of bridging structural holes are likely to perform better due to their superior access to information. Other researchers have focused on how partner diversity influences firm performance (Baum, Calabrese, & Silverman, 2000; Beckman & Haunschild, 2002; Hoang & Rothaermel, 2005). For instance, (Beckman & Haunschild, 2002) find that a firm's acquisition performance increases with the diversity of its partners.

Yet another stream of research has focused on the relationship between network



structure and innovation output. For instance, (Ahuja, 2000a) examines how the network positions of firms influence their innovation output with a longitudinal study of the chemical industry. He finds that while structural holes have a negative impact on innovative output, both direct and indirect ties have a positive effect on innovation. Similarly Rowley, Behrens, & Krackhardt (2000) use the semiconductor and steel industry as settings to reveal that weak ties are beneficial for exploration while strong ties are beneficial for exploitation.

While the studies discussed above have discussed how network positions enhance firm performance, some studies have also focused on how network positions constrain firm performance. For example, Uzzi (1997) in his study of the New York fashion industry highlighted the “paradox of embeddedness” by demonstrating that more embedded firms had a reduced ability to adapt due to the reduction of non-redundant ties and over-embeddedness. Likewise, other studies have also demonstrated that over-embeddedness can hinder the ability of firms to alter their portfolio of network ties, thereby, resulting in network inertia (Tai-Young Kim, Oh, & Swaminathan, 2006). Similarly, Bae & Gargiulo (2004) analyze the impact of alliance network structure and partner substitutability on firm performance in the US telecommunications industry. They find that ties with powerful partners may reduce their control over resources, which in turn outweighs the advantages derived from the alliance.

This wide-ranging literature on the relationship between network structure and performance reveal some interesting implications. One, firms that are located more centrally in a network typically exhibit superior performance as such locations facilitate

access to resources. Two, access to benefit rich networks is a form of social capital that increases in value over time (Burt, 1997; Coleman, 1988; Smith-Doerr, Manev, & Rizova, 2004). Much of the studies on the relationship between network configuration and firm performance have, thus, given rise to the need to establish the conditions for network relationships and focus on which ties are the most likely and most consequential (Powell, White, Koput, & Owen-Smith, 2005). The following section discusses the process of inter-organizational tie formation and partner selection.

### **3.1.2. The Influence of Network Structure on Tie Formation**

While a large number of studies adopting the network perspective have focused on the relationship between network structure and firm performance, an equally large body of research on inter-organizational networks and alliances has been dominated by studies attempting to understand the partnering process. The focus of such studies has been typically on the formation of ties and subsequently how patterns of tie formation influence overall network structure and dynamics (Shipilov, Rowley, & Aharonson, 2006). Researchers studying the phenomenon of tie formation have typically found that this process is a function of mitigating the uncertainty associated with partner selection. The most commonly explored types of uncertainty are partner reliability uncertainty, partner capability uncertainty and more recently, partner competitiveness uncertainty (Shipilov, Rowley, & Aharonson, 2006).

Typically, success in inter-organizational ties requires that partners are willing to cooperate and reciprocate the contributions made by other actors (Oxley, 1997; Khanna,

Gulati, & Nohria, 1998). As a result, inter-organizational partnering patterns show that firms use their past experience to reduce partner reliability uncertainty and to make future partnership decisions. A growing body of research on inter-organizational networks argues that in making their partnership choices firms rely on information provided by existing networks (Powell, Koput, & Smith-Doerr, 1996; Gulati, 1998; Gulati, 1999). The reliance on past information draws on sociological theories of embeddedness. Embeddedness refers to “the fact that exchanges and discussions within a group typically have a history, and that this history results in the routinization and stabilization of linkages among members (Marsden, 1981)”. Granovetter (1985) extends this concept and argues that economic actions of firms are influenced by the social structure of ties within which they are embedded. A firms’ network of prior alliances serves as a source of trustworthy information about the reliability and capabilities of partners (Gulati, 1995a; Kogut, Walker, & Kim, 1995; Gulati & Gargiulo, 1999). Relational embeddedness highlights the impact of strong/cohesive ties between actors. Cohesive ties amplify trust and create more certainty regarding future partnerships (Podolny, 1993; Gulati, 1995a). They also prompt organizations to become less willing to engage in partnerships outside of these close relationships (Uzzi, 1996; Uzzi, 1997; Venkatraman & Lee, 2004).

A related form of embeddedness – structural embeddedness captures the impact of the structure of the network in which the actors operate on their tendency to form ties with one another (Granovetter, 1985; Gulati & Gargiulo, 1999). Studies adopting this perspective argue that organizations tied to common partners can utilize reliable information about each other from the common partner (Baker, 1990; Burt & Knez,

1995). As a result, organizations are also more likely to engage in ties with their partners' partners to reduce reliability uncertainty (Gulati, 1998; Gulati, 1999).

Another stream of research has focused on firm resource and network positions and their impact on reducing partner capability uncertainty. Prior research on the dynamics of inter-organizational networks has found support for the idea that firms strive to partner with firms that have complementary resources. Nohria & Garcia-Pont (1991), in their study of the automobile industry, found that firms in different strategic groups formed alliances to gain access to complimentary resources. Similarly, Gulati & Gargiulo (1999) and Chung, Singh, & Lee (2000) in their studies found that firms were more likely to form alliances with partner based on geographical similarity. Both studies found that firms were more likely to engage in alliances when they served in different geographical areas. These findings are in line with the argument that firms utilize partnerships to access capabilities that they do not possess and hence search for dissimilar partners (Li & Rowley, 2002). Moreover, to mitigate partner capability uncertainty, firms tend to consider firm attributes such as size, scope, and product mix (Greve, 1999). Larger firms typically signal the possession of more organizational capabilities and financial legitimacy (Levinthal, 1991). Shipilov, Rowley, & Aharonson (2006) extend research on the tie formation process by disaggregating partner selection into new and ongoing relationships. As a result, they differentiate factors influencing initial emergence of the network from its subsequent evolution and find that firms tend to engage in tie formation when functional dissimilarity is high and are also less likely to terminate such ties.

Likewise studies have also focused on network positions of actors and their ability to attract more ties. Such research on the tie formation process has emphasized a recursive relationship between partnering decisions and network structure: partnering decisions lead to the emergence of the network and once network structure emerges; it determines its own evolution and guides partnering choices (Gulati, 1999; Li & Rowley, 2002; Baum, Rowley, Shipilov, & Chuang, 2005). For instance, Oliver (1990) argues that firms choose to form partnerships with high status partners to increase legitimacy. Similarly, Podolny (2001) extends this argument by stating that a partner's status acts as a proxy for quality. He further argues 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. Stuart, (2000a) argues that ties with prestigious partners are often perceived by others to be associated with valuable skills, resources and capabilities. Other studies, focusing on the macro-structure of the network also find support for the tie formation process as a function of specific moves by individual actors over time (Venkatraman & Lee, 2004; Powell, White, Koput, & Jason Owen-Smith, 2005). This is in line with the "rich-get-richer" argument of preferential attachment (Barabási, Jeong, Néda, Ravasz, Schubert, & Vicsek, 2002), wherein the propensity to link to nodes with more ex-ante links is higher.

This dissertation is in line with studies that have focused on understanding partnership choices but with important distinctions: One, we focus on a relatively new setting: platform-complementor networks that has not received much attention in prior research with the exception of (Venkatraman & Lee, 2004) and their study of the

preferential attachment choices of video game developers. Two, no study has so far attempted to understand the role of complementor network positions in driving their choice of a platform. Even Venkatraman & Lee (2004) in their study focus on platform characteristics and their network positions in driving complementor decisions. This study focuses on the role complementor network positions coupled with platform attributes in driving complementor choice. Three, unlike most studies that treat all types of ties as equal, this dissertation acknowledges that there is heterogeneity in the type of support extended by developers and that this has subsequent performance implications. As a result, I focus on the propensity of developers to engage in two types of tie formation with complementors: unique ties and ported ties.

### **3.1.3. Network Concepts in the Context of Platform-Complementor Networks**

Networks have become an important area of focus in management research and a number of conceptualizations and constructs have been used to understand firms' positions in the network and the consequences of those positions. In its broadest conceptualization, a network is a set of nodes and relationships that link two nodes together (Barabási, Jeong, Néda, Ravasz, Schubert, & Vicsek, 2002). The nodes are actors and can be individuals, groups of people or organizations depending upon the context of the study. A tie, also known as a link connects two actors together and multiple ties combine to form the network (de Nooy, Mrvar, & Batagelj, 2005). In the context of platform-based settings, the actors are platform owners and third-party complementors and the ties are the applications/software that are developed by complementors for a

platform. The constructs that we are interested in for this research are degree centrality, and embeddedness. Centrality is perhaps the most commonly analyzed network measure. Degree centrality of a node reflects the number of ties/linkages an actor has to other actors. Centrality of an actor is important as it reflects the power/status of an actor (Oliver, 1990) and also implies that actors with more ties have access to more knowledge resources and hence more likely to succeed (Gulati, 1995a; Ahuja, 2000a). In the context of platform-based settings, the number of software titles launched on a platform represents platform centrality while the number of titles launched by a developer represents developer centrality.

Ties can also differ in their type and this in turn is likely to influence the impact the tie has on performance. In platform-based settings, the variety of ties can be reflected in the mode of support extended by a developer to a platform. Each game title that is launched first or is exclusive to a platform is a unique tie whereas game titles that are re-released or ported to another platform are ported ties. The extent of ported ties reflects the degree of overlap between platforms. A high degree of overlap between actors reduces the competitive position of the actors as they have access to the same resources and also have nothing to unique to offer as opposed to their competitors. In platform-based settings industry, the degree of overlap is the extent to which two platforms have the same titles launched on their respective platforms. Other network measures include network embeddedness, which reflects the pattern of distribution of ties for an actor. In the context of the platform settings, embeddedness manifests in the form of the distribution of game titles by a game developer for a platform (Venkatraman & Lee,

2004). High platform embeddedness would imply that a larger proportion of a platform's game titles are developed by a small number of developers while low embeddedness implies that a platform is supported by a large number of developers. For instance, by December 2006, approximately 57% of PS2s' games were developed by the ten largest game developers while only about 20% of GC's games were developed by the ten largest developers, implying that the PS2 platform was more strongly embedded in the top-ten developer network as opposed to GC. Similarly, high developer embeddedness implies that a large proportion of a developer's titles are launched for a small number of platforms.

With this review of key network concepts, I present the hypotheses of the study in the following chapter.



## **IV. THEORY AND HYPOTHESES**

In platform-based settings, the relationship between platform firms and their ecosystem of complementors is crucial to driving platform success. Under such settings, therefore, complementors need to continually refine and modify their knowledge and routines to align with shifting architectures of platforms. This is particularly true in platform settings where technology evolves very rapidly and multiple competing platform exist contemporaneously vying for support from the same set of complementors. 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 dissertation draws on theories of firm network positions to understand developers' ability to support platforms through ported or unique ties. I specifically hypothesize how the role of developer embeddedness influences the pattern of support a developer extends to a platform. I, 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.

### **4.1. Developer Attributes and Platform Support through Ported Ties**

#### **4.1.1. Developer Embeddedness**

A complementors' strategy, in addition to evolving as a function of its firm

specific capabilities, also often evolves as a function of its relative network position in the platform-complementor ecosystem. Therefore, this dissertation adopts a network perspective that recognizes the role of a firm's prior network positions in influencing its future strategic choices. It has been well established in prior research that organizational actions are embedded in networks of relationships (Granovetter, 1985; Gulati, 1999; Powell, White, Koput, & Owen-Smith, 2005; Venkatraman & Lee, 2004).

Marsden defines explains the concept of 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” (Marsden, 1981). Extending this concept of embeddedness, (Uzzi, 1996) in his study of the apparel industry found that the structural embeddedness of a firm influenced its mortality. He further argued that high embeddedness of a firm typically reduced its ability to adapt to new requirements (Uzzi, 1997).

Within platform-based settings, the actions of developers are a reflection of their prior ties and patterns of embeddedness (Venkatraman & Lee, 2004). Embeddedness in this setting is manifested as the pattern of 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. In such a situation, a developer does not support a large number of platforms. As a result 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.*

## **4.2. Platform Attributes and Support through Ported Ties**

### **4.2.1. Platform Centrality**

In addition to the role of developers' network positions in driving their patterns of support extended to platforms, platform attributes themselves are likely to influence when and how a complementor extends its support to a platform. This dissertation, therefore, first theorizes about the role of platform centrality in driving a complementors' choice to support a platform through ported ties. A large number of studies have focused on the role of market positions in driving outcomes. For instance, Arthur (1989) demonstrates that market dominance creates strong positive feedback and gives rise to increasing returns to scale.

Network theorists have also supported 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) also 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 adopting the network perspective 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 complementors 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, complementors 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.*

#### **4.2.2 Platform Porting Centrality**

The next assertion, which also concerns the logic of attachment of a developer to a particular platform provides 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 dissertation 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.*

#### **4.2.3. Platform Architectural Distance**

A large stream of research on alliances and networks has focused on how firms choose alliance partners and its implication on performance. An important 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 and 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 interacted in the past.

In platform markets, there is no formal contractual agreement between platform providers and complementors and as a result the extent of partner reliability uncertainty is likely to be low. Instead, in making their choices, third-party complementors are more likely to rely on cues that allow them to reduce their capability uncertainty. 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.*

#### **4.2.4. Platform Genre Distance**

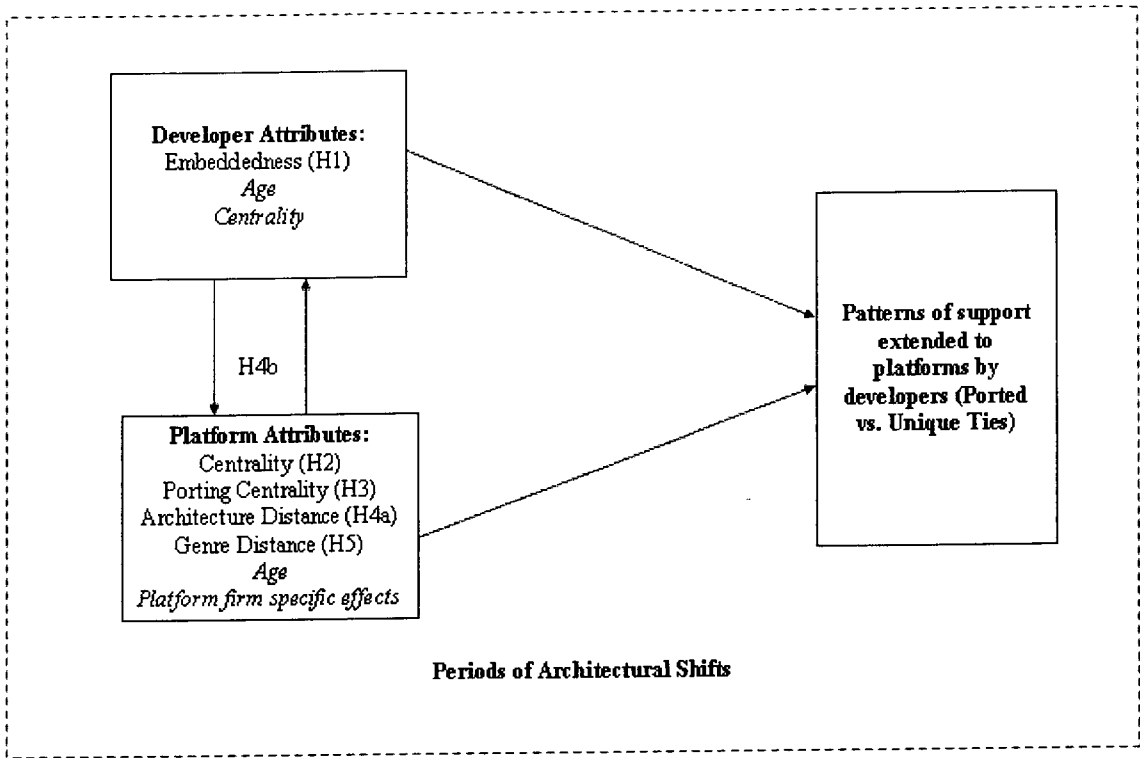
Previous research on alliances and networks has demonstrated the extent of partner similarity to an actor's earlier partners as an important determinant of partner choice (Gulati, 1995a; Gulati, 1999; Shipilov, Rowley, & Aharonson, 2006). The following assertion also concerns the role of partner similarity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

Figure 5 shows the conceptual research model of the dissertation.



**Figure 5. Conceptual Research Model**

## V. METHODS

This section describes details of the research design used in testing the proposed research model and hypotheses. Most prior empirical research on platform-based competition has examined the outcomes of having a large library of complementary software on platform provider success. Relatively little research (for example, Venkatraman & Lee, (2004)) has examined the launch decision of games for each third-party developer over time. The goal of this dissertation is to extend current research on platform-based competition by adopting a longitudinal lens to understand how developer and platform characteristics the choice of support extended by developers to platforms and how architectural shifts alter the choice of support over time. Such a longitudinal analysis required data collection at a highly granular level. The research setting, data sources and data collection process are described below.

### **5.1. Research Setting**

This dissertation 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 resulted in an architectural shift that 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, namely: Playstation 2 (PS2), Xbox, GameCube (GC), Xbox 360, Wii and Playstation 3 (PS3).

## **5.2. Data**

The data for the dissertation was obtained from multiple sources. The primary source of data was the website [www.gamespot.com](http://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 I arrived at 316 developers. Since the intent of this dissertation 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 dissertation 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. The data was further validated with the lists provided on the website [www.ign.com](http://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.

In addition to studying developers' choice of a platform, this dissertation also seeks to offer insight into the pattern of support extended to consoles. As a result, information on whether a title was unique to a platform or ported from another platform

was also collected by capturing the sequence of launch of game titles on platforms. The rationale for capturing the sequence of launches was to help understand the pattern of support provided by the developer to a console. The sequence of launch of titles reflects how third-party developers respond to the architectural capabilities of platforms. A tie between a developer and platform was coded as unique if the game was launched first or exclusively for that developer. A tie was coded as a ported tie when a game was ported from an existing platform.

### **5.3. Analysis 1**

One goal of this dissertation is to unravel the dynamics of platform-based competition by understanding how developers respond to architectural shifts initiated by platform providers. Analysis 1, therefore, seeks to understand how developer attributes influence their ability to adapt to architectural shifts and support a platform. In addition to providing insight into how developer attributes influence a developer's ability to leverage the architectural shifts in platforms and support platforms through either ported ties or unique ties, this analysis also sheds light on how platform attributes (architectural and network positions) influence developers' choice of platform to port to. The measures used for Analysis 1 are reviewed below.

#### **5.3.1. 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 a 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,  $y_{ij, t}$  equaled 1 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) 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 control for 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:

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

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  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$ . In the first measure  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$  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<sub>jk</sub> and API<sub>jk</sub> 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. The details of this measure are discussed in Section 5.6.

**Platform Genre Distance.** Platform genre distance was a variable introduced to control 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 dissertation 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.

Table 1 summarizes the operationalizations of variables included in the study.

**Table 1. Variables in Statistical Analysis for Analysis 1**

Label	Unit of Observation	Description
<i>Dependent Variable:</i> Pattern of Support	Tie	Binary indicator of whether a platform is supported through a ported tie or unique tie
<i>Independent Variables:</i> Developer Embeddedness	Developer	Sum of square of proportion of titles launched by developer $i$ for platform $j$ in time period $t$
Platform Centrality	Target Platform	Proportion of titles launched on platform $j$ in time period $t$
Platform Porting Centrality	Target Platform	Proportion of titles ported to platform $j$ in time $t$
Platform Architecture Distance	Dyad: Source and Target platform	Architecture distance in the Development environment, hardware and OS between the source and target platforms
Platform Genre Distance	Dyad: Source and Target platform	Difference in the proportion of titles in genre $g$ between source platform $k$ and target platform $j$
<i>Control Variables:</i> Developer Age	Developer	Age of developer in months at time $t$
Developer Centrality	Developer	Proportion of titles launched by a developer in time $t$
Platform Age	Target Platform	Age of platform $j$ in months
Platform Firm Fixed Effects	Source Platform	Firm dummy indicating which firm launched a platform

### 5.3.2. 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 in this dissertation 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).

The nested logit model extends the multinomial logit model by relaxing the restriction that the distribution of the random error terms is independent and identical over alternatives (IIA property). The nested logit model relaxes the requirement that the error terms of a pair or group of alternatives cannot be correlated (Koppelman & Chieh-Hua Wen, 1998). Therefore, this model is useful when a set of alternatives is more similar than the other alternatives in some unobservable way (Ben-Akiva & Lerman, 1985; Train, 2003). Under this model, similar alternatives are grouped into nests, thereby creating a hierarchical structural of the alternatives.

The nested logit approach has been predominantly used in the areas of transportation research and logistics (Bhat, 1997; Train, 1980). This approach has also been used extensively in marketing research primarily in purchase incidence decisions (Chintagunta, 1993; Chintagunta & Vilcassim, 1998) and in brand-choice modeling (Chib, Seetharaman, & Strijnev, 2004; Kamakura, Kim, & Lee, 1996; Sun, Neslin, & Srinivasan, 2003). For instance, Sriram, Chintagunta, & Neelamegham, (2006) studied consumers' choice of technology products where the products are nested by brand (eg.

Sony) and then by its various models (e.g. Mavica, DSC, FD) at the lower level of the nest.

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). For instance, Haaijer, Kamakura, & Wedel, (2001) modeled an individual’s choice of a technology product as a function of the attributes of the technology such as brand, speed and price. The model formulation also included the no-purchase option as one nest where respondents could choose from one of 20 technology products or choose not to purchase the product at all.

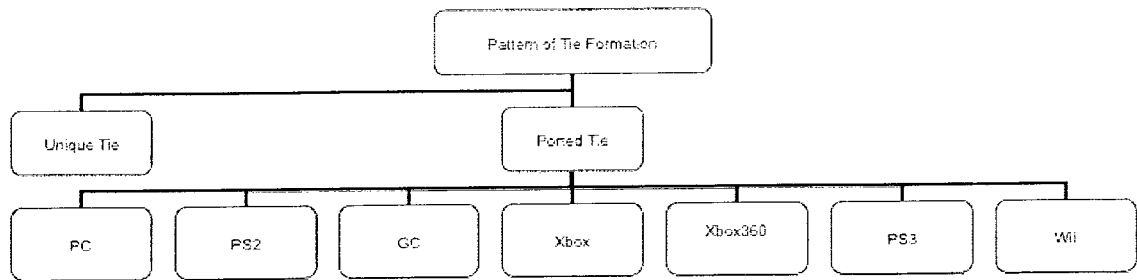
In this study, the goal was to understand how developer attributes and platform attributes influence a developers’ choice to support a platform through either ported ties or unique ties. This study, also further sought to explain which platform a developer was most likely to choose once the decision 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:

$$Prob(\text{dev } i \text{ chooses platform } j \text{ to port}) = Prob(i \text{ chooses } j | i \text{ chooses to port}) * Prob(i \text{ chooses to port})$$

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 6.

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 2.



**Figure 6. Nested logit model for developer support through porting vs. unique tie**

**Table 2. Summary Statistics for Analysis 1**

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

**N= 20,254; p<0.05**

## **5.4. Analysis 2**

In this dissertation, in addition to explaining the process of tie formation and the influence of developer attributes and platform provider strategies on the pattern of support extended by developers, I also seek to understand how the patterns of tie formation between developers and platforms evolve over time with architectural shifts that have driven convergence between PC and video game consoles. The goal of Analysis 2 is therefore to explain at the macro level how architectural shifts during the transition of video game consoles from the sixth to the seventh generation influence the dynamics of platform-complementor network evolution. For this purpose, I focus only on the support of platforms through ported ties and how this support extended by developers evolves as consoles transition to the newer generation. The measures used for Analysis 2 are described below.

### **5.4.1. Construct Operationalization**

*Likelihood of support through ported ties.* The dependent variable – a developers' choice to support a platform through ported ties was defined as categorical and time varying. Each a time a developer made a decision to port a game to a platform, 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,  $y_{ij, t}$  equaled 1 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. For example, when Electronic Arts launched the Madden NFL 2002 in August 2001, it launched it first on Sony PS2. 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. Moreover 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.

*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 control for 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:

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

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  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$ . In the first measure  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$  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 –  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$  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. The details of this measure are discussed in Section 5.6.

**Platform Genre Distance.** Platform genre distance was a variable introduced to control 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.

**Source platform generation.** Source platform generation for platform  $k$  was defined as a categorical variable that captured whether a game title was ported from a sixth generation platform, seventh generation platform or from the PC platform.

**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.

**Developer Prior Ties.** Developer prior ties (Granovetter 1985) on date  $t$  are defined as the proportion of titles released by developer  $i$  for platform  $j$ .

**Platform firm fixed effects.** Three firms launched the seven platforms that were the focus of this – Microsoft (PC, Xbox and Xbox360), Sony (PS2 and PS3) and Nintendo (GC and Wii). Firm level dummy variables were introduced for the target platform  $j$  to control for firm specific effects on the likelihood of porting.

**Launch of new platform generation.** The launch of a new platform generation by the same firm was also controlled for by introducing a step dummy variable in the time periods that two competing generation consoles continued to co-exist.

Table 3 summarizes the operationalizations of variables included in the study.

**Table 3. Variables in Statistical Analysis for Analysis 2**

<b>Label</b>	<b>Unit of Observation</b>	<b>Description</b>
<i>Dependent Variable:</i> Support through ported ties	Tie	Binary indicator of which a platform is supported through a ported tie
<i>Independent Variables:</i> Platform Centrality	Target Platform	Proportion of titles launched on platform j in time period t
Platform Porting Centrality	Target Platform	Proportion of titles ported to platform j in time t
Platform Architecture Distance	Dyad: Source and Target platform	Architecture distance in the Development environment, hardware and OS between the source and target platforms
Platform Genre Distance	Dyad: Source and Target platform	Difference in the proportion of titles in genre g between source platform k and target platform j
<i>Control Variables:</i> Platform Generation	Source Platform	Dummy for generation that source platform belonged to
Platform Age	Target Platform	Age of platform j in months
Developer Prior Ties	Tie	Proportion of ties launched by developer i for platform j
Platform Firm Fixed Effects	Target Platform	Firm level dummy variable for target platform j
Launch of new generation	Target Platform	Dummy variable indicating time periods after the launch of new generation platform

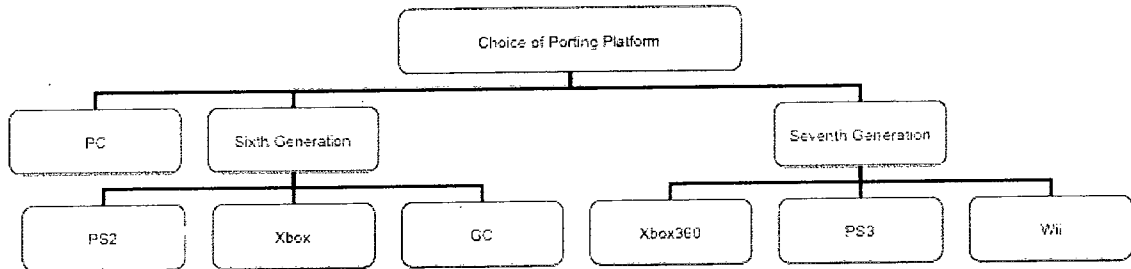
#### 5.4.2. Nested Logit Model for Support through Ported Ties during Transition from Sixth to Seventh Generation Consoles

In this analysis, the focus is only on the support extended by developers to platforms through ported ties. The motivation for focusing on ported ties was to unravel at the macro level the changes in the patterns of support as video game consoles underwent technological shifts and transitioned from the sixth to the seventh generation. Therefore, in order to understand the impact of architectural shifts on the dynamics of network evolution, an alternative nesting structure was used. 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. Under this formulation of the nests, the likelihood of a developer  $i$  porting to platform  $j$  was defined as:

$$Prob(\text{dev } i \text{ chooses platform } j \text{ to port}) = Prob(i \text{ chooses } j | i \text{ chooses to port in generation } g) * Prob(i \text{ chooses to port})$$

The nesting structure used in this study is shown in Figure 7. The data sample for this study consisted of all ported game titles that constituted 1241 unique developer-title-platform combinations. The estimation sample included all of the consoles not chosen in each of the 1241 cases. 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 5112 observations. Summary statistics for this analysis are included in Table 4.



**Figure 7. Nested logit model for developer support through ported ties during transition from sixth to seventh generation**

**Table 4. Summary Statistics for Analysis 2**

	Platform Centrality	Platform Architecture Distance	Platform Age	Platform Genre Distance	Platform Porting Centrality	Developer Prior Ties
Platform Centrality	1					
Platform Architecture Distance	-0.04	1				
Platform Age	0.42*	-0.07*	1			
Platform Genre Distance	0.36*	0.02	0.12*	1		
Platform Porting Centrality	0.09*	-0.33*	0.23*	-0.18*	1	
Developer Prior Ties	0.58*	-0.10*	0.25*	0.12*	0.23*	1
Mean	0.18	0.49	11.13	0.03	0.19	0.17
S.D.	0.17	0.22	24.27	0.01	0.13	0.16

N = 5112; p<0.05

### **5.5 Analysis 3**

In this study, the goal was to shed further light into the dynamics of platform-complementor network interactions by understanding how individual platform provider strategies influence the pattern of support extended to them by developers. When platform providers launch newer generation platforms, they need to devise architectural strategies that make it easier for their existing third-party complementors to migrate to their newer platforms and at the same time also attract newer complementors that support competing platforms. Their architectural strategies in turn influence how and when developers are able to make the corresponding technological shift and support platforms through ported ties. In this analysis, therefore, I intend to uncover how the support extended by third-party developers to the three platform firms – Microsoft, Sony and Nintendo – varies and to what extent this variation is a function of the architectural differences between the platforms. For this purpose, I focus only on the support of platforms through ported ties and whether firm level differences influence the extent to which developers support platforms through such ties. The measures used for Study 3 are described below.

#### **5.5.1. Construct Operationalization**

*Likelihood of support through ported ties.* The dependent variable – a developers' choice to support a platform through ported ties was defined as categorical and time varying. Each a time a developer made a decision to port a game to a platform, 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,  $y_{ij,t}$  equaled 1 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. For example, when Electronic Arts launched the Madden NFL 2002 in August 2001, it launched it first on Sony PS2. 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. Moreover 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.

***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 control for 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:

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

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  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$ . In the first measure  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$  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 –  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$  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. The details of this measure are discussed in Section 5.6.

***Platform Genre Distance.*** Platform genre distance was a variable introduced to control 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.

***Source platform firm.*** Source platform firm for platform  $k$  was defined as a categorical variable that captured whether a game title was launched first on a Sony, Microsoft or Nintendo platform.

***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.

***Developer Prior Ties.*** Developer prior ties (Granovetter 1985) on date  $t$  are defined as the proportion of titles released by developer  $i$  for platform  $j$ .

**Platform firm fixed effects.** Three firms launched the seven platforms that were the focus of this – Microsoft (PC, Xbox and Xbox360), Sony (PS2 and PS3) and Nintendo (GC and Wii). Firm level dummy variables were introduced for the target platform  $j$  to control for firm specific effects on the likelihood of porting.

**Launch of new platform generation.** The launch of a new platform generation by the same firm was also controlled for by introducing a step dummy variable in the time periods that two competing generation consoles continued to co-exist.

Table 5 summarizes the operationalizations of variables included in the study.

**Table 5. Variables in Statistical Analysis for Analysis 3**

<b>Label</b>	<b>Unit of Observation</b>	<b>Description</b>
<i>Dependent Variable:</i> Support through ported ties	Tie	Binary indicator of which a platform is supported through a ported tie
<i>Independent Variables:</i> Platform Centrality	Target Platform	Proportion of titles launched on platform $j$ in time period $t$
Platform Porting Centrality	Target Platform	Proportion of titles ported to platform $j$ in time $t$
Platform Architecture Distance	Dyad: Source and Target platform	Architecture distance in the Development environment, hardware and OS between the source and target platforms
Platform Genre Distance	Dyad: Source and Target platform	Difference in the proportion of titles in genre $g$ between source platform $k$ and target platform $j$
<i>Control Variables:</i> Source Platform Firm	Source Platform	Dummy for firm that source platform belonged to
Platform Age	Target Platform	Age of platform $j$ in months
Developer Prior Ties	Tie	Proportion of ties launched by developer $i$ for platform $j$
Platform Firm Fixed Effects	Target Platform	Firm level dummy variable for target platform $j$
Launch of new generation	Target Platform	Dummy variable indicating time periods after the launch of new generation platform

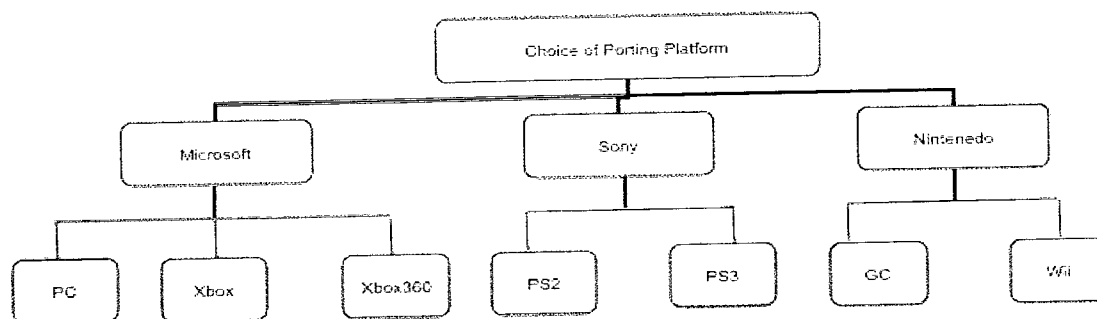
### 5.5.2. Nested Logit Model for Support through Ported Ties with firm level nests

The focus of this analysis was to explain how the patterns of support extended by third-party developers varied for each firm as they launched newer generation platforms. This analysis was motivated by the need to understand how the architectural shifts by individual platform firms influenced the type of support that they received from their complementors. Therefore, in order to understand the impact of architectural shifts at the firm level on the dynamics of network evolution, an alternative nesting structure was used. Under this nesting structure, the platforms were nested according to the platform provider that launched them. Nesting platforms in this manner helped in explaining the extent to which platform firms were able to leverage the support that had from their existing ecosystem of complementors as they transitioned to newer generation platforms. Under this formulation of the nests, the likelihood of a developer  $i$  porting to platform  $j$  was defined as:

$$Prob(\text{dev } i \text{ chooses platform } j \text{ to port}) = Prob(i \text{ chooses } j | i \text{ chooses to port in platform firm } p) * Prob(i \text{ chooses to port})$$

The nesting structure used in this study is shown in Figure 8. The nesting structure used in this study is shown in Figure x. The data sample for this study consisted of all ported game titles that constituted 1241 unique developer-title-platform combinations. The estimation sample included all of the consoles not chosen in each of the 1241 cases. 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 5112 observations. Summary statistics for this analysis are included in Table 6.



**Figure 8. Nested logit model for developer support through ported ties for firm level nests**

**Table 6. Summary Statistics for Analysis 3**

	Platform Centrality	Platform Architecture Distance	Platform Age	Platform Genre Distance	Platform Porting Centrality	Developer Prior Ties
Platform Centrality	1					
Platform Architecture Distance	-0.04	1				
Platform Age	0.42*	-0.07*	1			
Platform Genre Distance	0.36*	0.02	0.12*	1		
Platform Porting Centrality	0.09*	-0.33*	0.23*	-0.18*	1	
Developer Prior Ties	0.58*	-0.10*	0.25*	0.12*	0.23*	1
<i>Mean</i>	<i>0.18</i>	<i>0.49</i>	<i>11.13</i>	<i>0.03</i>	<i>0.19</i>	<i>0.17</i>
<i>S.D.</i>	<i>0.17</i>	<i>0.22</i>	<i>24.27</i>	<i>0.01</i>	<i>0.13</i>	<i>0.16</i>

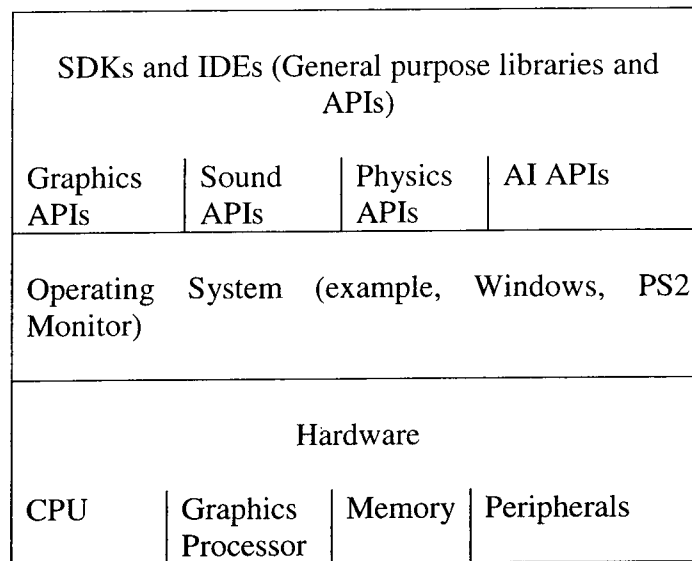
## **5.6. Measuring Architectural Distance between Platforms**

### **5.6.1. Platform Architectures**

A platform consists of an architecture of related standards – for computer platforms, the architectural standards comprise the processor, operating system (OS) and associated peripherals (West, 2003). The control of a platform’s complementary assets is determined by the ability to create application-programming interfaces (APIs) that specify how software can work with a platform (West & Dedrick, 2000). APIs provide applications with access to common system services. They take arguments from applications and call on system services to produce desired tasks and return results (Tanriverdi & Lee, 2008). Platforms are designed using principles of modularity (Baldwin & Clark, 1997), with each layer allowing some level of abstraction of the layer below.

In the case of gaming platforms, at the hardware or the processor level, platforms vary in terms of the type of processor, memory, and graphics capabilities. Video game consoles also typically have proprietary operating systems, which implies that developers have to often write hardware level code. Platform providers often provide game developers with software development kits (SDKs) that ease the development process. These SDKs consist of libraries, APIs, debugging tools, compilers and other utilities in a specific development environment called the Integrated Development Environment (IDE). In addition, game developers often use game engines or game middleware that includes a set of game development tools that allows some level of hardware abstraction.

The game engines typically include graphics, sound, physics and AI engines (Gal, Le Prado, Natkin, & Vega, 2004). The ability to use game engines for cross-platform development depends on platform differences and the APIs and IDEs that they support. Figure 9 shows the stack for gaming platforms.



**Figure 9. Software Stack for Gaming Platforms**

The extent of differences in the APIs and the development environments supported by the SDKs of different platforms determines the amount of time and effort developers have to expend in writing the same games for different platforms. Developers incur significant costs in customizing applications to work with different sets of APIs (Tanriverdi & Lee, 2008). While the APIs provided in the platform providers' SDKs provide a software abstraction of the hardware and OS details, hardware and OS differences become more important when developers consider cross-platform game development. Often the software code needs to be optimized for the different

hardware/processor capabilities of the platforms. Moreover, the peripherals used by these different gaming platforms vary and this also influences the extent to which a game's code needs to be modified to work optimally for the different peripherals. Therefore, differences at each layer of the stack influence the extent to which third-party developers can re-use their know-how and software code base in new settings.

### **5.6.2. Porting across Platforms**

Porting is defined as a series of processes required to move a program from one environment to another (Hakuta & Ohminami, 1997). The ease with which a program can be ported to a new environment is a function of the differences between the target and source environments and also the extent to which porting is taken into consideration when the program was written (Hakuta & Ohminami, 1997). Since delving into the details of the programming language used for each game is beyond the scope of this study, this dissertation focuses only on the differences between the source and target environments in measuring the architectural distance between platforms.

In making porting choices developers often have to take into consideration key system environment differences between the source and target platforms. (Hakuta & Ohminami, 1997) identify three important factors that address these system environment differences: differences in hardware and processor architectures; differences in operating system and finally, differences in programming language and development environments. The relative role of each of these factors is underscored by Jon Peddie, President of Jon Peddie Research and game industry expert in his comments on the ease of porting games from PC to Mac. He notes that:



*“In developing for the Mac, developers have to overcome three key problems – hardware is part of the problem, the OS and APIs are the biggest barrier.”*

In order to better understand the architectural differences between PC and sixth and seventh generation video game consoles, a triangulation of methods was used. First, an extensive research of technical articles on porting, video game developer trade journals, video game developer forums and blogs was undertaken to understand porting trends in general and the specific challenges and problems associated with porting in particular. Two, five video game developers were interviewed to understand how they addressed the issue of porting and to what extent the architectural differences between platforms influenced their porting choices. Thus, each developer offered their perspective on factors and architectural components that influenced their decision and ability to port across platforms. Finally, technical articles and papers that provided in-depth discussions between the different platform architectures were used to compute the architectural distance metric discussed in the subsequent sections. The influence and relative importance of each of these factors in determining porting choices is now discussed below:

### **5.6.3. Hardware and processor architecture differences**

Hardware differences across platforms manifest through variations in CPU and GPU architectures, memory constraints, and access to peripheral devices such as controllers. The size and range of basic data types, the internal representation of characters, the endianness, memory data alignment schemes and padding of data are a

function of the processor architecture and hence, while porting applications across platforms the code often needs to be modified with respect to processor architectures. Developers identified differences in the x86 architecture of PCs and the PowerPC architecture of consoles as influencing the extent of code that needed to be rewritten when porting between PC and consoles. For instance, one developer stated,

*“all major consoles are PowerPC-based and therefore, big-endian while most PC games or on little-endian windows. So game code and data would have to be written out in the appropriate endianness of the platform.”*

Here endianness is defined as the ordering of bytes stored in external memory. In the little-endian format, the least significant byte is stored in the lowest memory address while in the big-endian format; the most significant byte is stored in the lowest memory address (Cohen, 1981). To account for the difference in endianness, developers have to either rely on compilers to do the conversion or they need to rewrite part of the code to swap the bytes.

Game code also needs to be modified based on GPU (Graphics Processing Unit) differences across platforms. As explained by a developer:

*“The way you actually get the character skinning information onto the GPU differs from platform to platform. The basic algorithmic principle is the same, but the actual implementation details can be very different.”* (Hardwidge, 2009).

Other hardware differences that influence the ease of porting are the memory differences across platforms. A key memory consideration identified by a developer in porting between PC and consoles was,

*“Probably the most important difference between PC and any console game is memory constraints; consoles have much less memory than the typical PC. For instance, PS3 has only 256MB of main memory (and 256MB of VRAM). I would imagine this is the biggest long-term headache: making the game fit into much less main memory than you would usually have on a PC.”*

Similar memory management differences become crucial when porting between Wii and other consoles such as PS3 and Xbox360 since the Wii has significantly lesser memory than the competing platforms. Another factor that influences the ease of porting is the differences in controllers used by the platforms. For instance, as a developer stated:

*“Both X360 and PS3 have similar controllers, the Wii is different. As a result, game code needs to be modified to work optimally with different controllers.”*

#### **5.6.4. OS differences**

OS disparities between the source and target platforms include the scope of system call support and the implementation of system calls (Hakuta & Ohminami, 1997). As a result, game code would have to be recompiled and relinked with completely different OS libraries. In addition, OS differences also influence how memory constraints are managed and optimized. OS differences also influence how thread organization is managed. For instance, one of the biggest issues cited by developers in porting games to PS3 is the need to re-organize threading to work on the Cell processor. Table 7 highlights the key hardware and OS differences between PCs and sixth and seventh generation video game consoles.

**Table 7. Platform Hardware and OS Standards**

	<i>PC</i>	<i>PS2</i>	<i>Xbox</i>	<i>GC</i>	<i>Xbox360</i>	<i>PS3</i>	<i>Wii</i>
<i>OS</i>	Windows	Proprietary Linux based	Windows	Proprietary	Windows	Proprietary Linux based	Proprietary Linux based
<i>CPU</i>	Intel's x86	Emotion Engine	Intel Pentium III	IBM PowerPC "Gekko"	IBM PowerPC (triple core)	Cell Architecture	IBM PowerPC (single core)
<i>Memory</i>	Variable (64-256 MB)	32MB	64MB	40MB	512MB	256MB	88MB
<i>GPU</i>	ATI	Graphics Sythesizer	GeForce3 (Nvidia)	Flipper (NEC DRAM)	ATI	NVidia	ATI
<i>Input Devices</i>	Keyboard /Mouse	controller	controller	controller	controller	controller	Remote

### 5.6.5. Development environment and API differences

Platform providers often provide third-party developers with a set of tools (SDKs) that allow developers to simplify the software development process by abstracting some of the lower level hardware and OS details. The SDKs typically include the key libraries and APIs for rendering, audio, file system and memory management (Hardwidge, 2009). For instance, the DirectX API supported by Microsoft platforms includes support for graphics, audio, controller input, network communication. The library support range varies across platforms and the extent of variation determines how much effort developers have to expend in learning the APIs and also how much code needs to be rewritten across platforms. Typically depending upon the extent of differences in the APIs provided by the platforms, up to 90% of the code can be shared across platforms

(Hardwidge, 2009). This was further validated by a developer who stated,

*“the biggest factors (in making the porting decision) are the tools, libraries and engines in use. With the availability of compatible tools and cross-platform engines almost 99% of the code can be shared by platforms.”*

Within gaming platforms, the APIs that are most critical in determining the extent of rework required are the lower level API calls such as graphics, physics and sound APIs. As one developer argued,

*“High-level things like AI, animation, game play logic, loading/spawning, and high-level rendering logic are probably fairly safe, but all the low-level stuff like rendering setup and API calls, input handling, and sound will need major rework.”*

For instance, since the PC, the Xbox and the Xbox360 all use the DirectX Graphics API it is easiest to port between them. On the other hand, the PS3 supports a competing graphics API – the OpenGL ES - variation of the OpenGL API and hence is significantly different from the PC and Xbox360. In addition to differences in APIs and libraries supported by the platforms, the integrated development environment that includes the compilers and key debugging tools also influences the extent to which code can be easily reused across platforms. For example, since the Wii and Xbox360 support different IDEs (Codewarrior and Visual Studio respectively), the compilers and debugging tools are significantly different which in turn influences the ease of testing code across platforms. Table 8 shows the differences in the development environment and key APIs between PC and video game consoles.

In the following section, I discuss how each of these components together influences the architectural distance between platforms.

**Table 8. Platform APIs and Development Environment**

	PC	PS2	Xbox	GC	Xbox360	PS3	Wii
<b>Prog. Lang.</b>	C/C++	C/C++	C/C++	C/C++	C/C++	C/C++	C/C++
<b>Dev. Env.</b>	VS	ProDG	VS	Codewarrior	VS/XNA	ProDG/VS	Codewarrior
<b>Graphics API</b>	D3D	PS2GL (Proprietary)	D3D	OpenGL	D3D	PSGL	OpenGL
<b>Physics/AI API</b>	DirectX	Proprietary	Direct X	Havok	DirectX	PhysX	PhysX
<b>Sound API</b>	DSL	OpenAL	DSL	OpenAL	DSL	OpenAL	OpenAL

#### 5.6.6. Platform Architecture Distance Measure

The differences between the hardware, OS and development environments and API across the platforms were discussed in detail the previous section. Specifically, at the hardware and OS level differences included the OS type, the CPU type, memory differences, GPU type, and types of input devices/peripherals supported. At the development environment and API level, differences included programming languages supported, the development environment supported, graphics API supported, the Physics API supported and the Sound API supported. These differences in hardware, OS and platform APIs and development environments were then 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:

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

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. Weights of 0.8 and 0.9 were chosen for the API and development environment since the majority of the hardware and OS level differences are abstracted by the platform APIs and hence had less impact on the extent of code rewrite. The choice of weights was further validated through feedback received from game developers as discussed in the previous section.

Two alternative schemes were also used to measure  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$ . In the first measure  $\text{Hardware}_{jk}$  and  $\text{API}_{jk}$  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 –  $\text{Hardware}_{jk}$

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.

Within the hardware and OS level differences, the more nuanced conceptualization was used in measuring the CPU differences between the Xbox360 and PS3, between Xbox360 and Wii, between PS3 and Wii, between GC and Xbox360 and finally, between GC and PS3. This is because while the CPUs of all these platforms was based on IBM's PowerPC architecture, they were based on different threading schemes which in turn influenced the extent of code needed to be rewritten across platforms. For instance, while both the Xbox360 and Wii are based on the PowerPC architecture, the Xbox360 uses the triple core CPU while the Wii uses the single core CPU. Differences in the main memory between PS2 and GC, Xbox and GC, Xbox and Wii, Xbox360 and PS3 and GC and Wii were also assigned weights of 0.5. This was to account for the fact that platforms within certain memory ranges did not encounter significant differences in memory management and performance.

At the API and development environment level, differences in the Graphics API between PS2 and PS3, PS3 and Wii and GC and PS3 were assigned weights of 0.5. For instance, in evaluating the graphics API difference between PS3 and Wii, since both their APIs were based on the OpenGL standards (PSGL for PS3 and a proprietary version of OpenGL for Wii),  $API_{jk}$  for Graphics API differences between PS3 and Wii was assigned a value of 0.5. On the other hand, since the Xbox360 was based on the DirectX standard, the value of  $API_{jk}$  between Xbox360 and PS3 and between Xbox360 and Wii was 1.



Table 9 shows the differences in development environment and APIs across platforms where  $API_{jk}$  has values of either 0 or 1. Table 10 shows the differences in hardware and OS across platforms where  $Hardware_{jk}$  has values of either 0 or 1. Table 11 shows the differences in development environment and APIs where  $API_{jk}$  has values of 0, 0.5 or 1 and Table 12 shows the differences in OS and hardware when  $Hardware_{jk}$  has values of 0, 0.5 or 1. Table 13 then computes the architectural distance across platforms for  $w_{jk} = 0.9$  and 0.1 for API and development environment differences and OS and hardware differences respectively; and  $w_{jk} = 0.8$  and 0.2 for API and development environment differences and OS and hardware differences respectively and  $Hardware_{jk}$  and  $API_{jk} = 0$  or 1. Table 14 computes the architectural distance across platforms for  $w_{jk} = 0.9$  and 0.1 for API and development environment differences and OS and hardware differences respectively; and  $w_{jk} = 0.8$  and 0.2 for API and development environment differences and OS and hardware differences respectively and  $Hardware_{jk}$  and  $API_{jk} = 0, 0.5$  or 1.

Figure 10 visually summarizes the architectural distance between the platforms of interest in this study.

**Table 9. Platform API and Dev. Environment Distance ( $API_{jk} = 0$  or 1)**

	<b>Prog. Lang.</b>	<b>IDE</b>	<b>Graphics API</b>	<b>Physics API</b>	<b>Sound API</b>	<b>Total</b>
<b>PC-PS2</b>	0	1	1	1	1	0.8
<b>PC-Xbox</b>	0	0	0	0	0	0
<b>PC-GC</b>	0	1	1	1	1	0.8
<b>PC-Xbox360</b>	0	0	0	0	0	0
<b>PC-PS3</b>	0	0	1	1	1	0.6
<b>PC-Wii</b>	0	1	1	1	1	0.8
<b>PS2-Xbox</b>	0	1	1	1	1	0.8
<b>PS2-GC</b>	0	1	1	1	0	0.6
<b>PS2-Xbox360</b>	0	1	1	1	1	0.8
<b>PS2-PS3</b>	0	0	1	1	0	0.4
<b>PS2-Wii</b>	0	1	1	1	0	0.6
<b>Xbox-GC</b>	0	1	1	1	1	0.8
<b>Xbox-Xbox360</b>	0	0	0	0	0	0
<b>Xbox-PS3</b>	0	0	1	1	1	0.6
<b>Xbox-Wii</b>	0	1	1	1	1	0.8
<b>Xbox360-PS3</b>	0	0	1	1	1	0.6
<b>Xbox360-Wii</b>	0	1	1	1	1	0.8
<b>PS3-Wii</b>	0	1	0	1	0	0.1
<b>GC-Xbox360</b>	0	1	1	1	1	0.8
<b>GC-PS3</b>	0	1	1	1	0	0.6
<b>GC-Wii</b>	0	0	0	1	0	0.2

**Table 10. Platform OS and Hardware Distance (Hardware<sub>jk</sub> = 0 or 1)**

	OS	CPU	GPU	I/O Devices	Total
PC-PS2	1	1	1	1	1
PC-Xbox	0	0	1	1	0.5
PC-GC	1	1	1	1	1
PC-Xbox360	0	1	0	1	0.5
PC-PS3	1	1	1	1	1
PC-Wii	0	1	0	1	0.5
PS2-Xbox	1	1	1	0	0.75
PS2-GC	1	1	1	0	0.75
PS2-Xbox360	1	1	1	0	0.75
PS2-PS3	1	1	1	0	0.75
PS2-Wii	1	1	1	1	1
Xbox-GC	1	1	1	0	0.75
Xbox-Xbox360	0	1	1	0	0.5
Xbox-PS3	1	1	0	0	0.5
Xbox-Wii	1	1	1	1	1
Xbox360-PS3	1	1	1	0	0.75
Xbox360-Wii	1	0	0	1	0.5
PS3-Wii	1	1	1	1	1
GC-Xbox360	1	0	1	0	0.5
GC-PS3	1	1	1	0	0.75
GC-Wii	0	0	0	1	0.25

**Table 11. Total Platform Architecture Distance (Hardware<sub>jk</sub> and API<sub>jk</sub> = 0 or 1)**

	API/Dev Env	CPU/OS	w <sub>jk</sub> = 90/10	w <sub>jk</sub> = 80/20	w <sub>jk</sub> = 60/40
PC-PS2	0.8	1	0.82	0.84	0.88
PC-Xbox	0	0.5	0.05	0.1	0.2
PC-GC	0.8	1	0.82	0.84	0.88
PC-Xbox360	0	0.5	0.05	0.1	0.2
PC-PS3	0.6	1	0.64	0.68	0.76
PC-Wii	0.8	0.5	0.77	0.74	0.68
PS2-Xbox	0.8	0.75	0.795	0.79	0.78
PS2-GC	0.6	0.75	0.615	0.63	0.66
PS2-Xbox360	0.8	0.75	0.795	0.79	0.78
PS2-PS3	0.4	0.75	0.44	0.47	0.54
PS2-Wii	0.6	1	0.64	0.68	0.76
Xbox-GC	0.8	0.75	0.795	0.79	0.78
Xbox-Xbox360	0	0.5	0.05	0.1	0.2
Xbox-PS3	0.6	0.5	0.59	0.58	0.56
Xbox-Wii	0.8	1	0.82	0.84	0.88
Xbox360-PS3	0.6	0.75	0.615	0.63	0.66
Xbox360-Wii	0.8	0.5	0.77	0.74	0.68
PS3-Wii	0.1	1	0.19	0.28	0.46
GC-Xbox360	0.8	0.5	0.77	0.74	0.68
GC-PS3	0.6	0.75	0.615	0.63	0.66
GC-Wii	0.2	0.25	0.205	0.21	0.22

**Table 12. Platform API and Dev. Environment Distance ( $API_{jk} = 0, 0.5$  or  $1$ )**

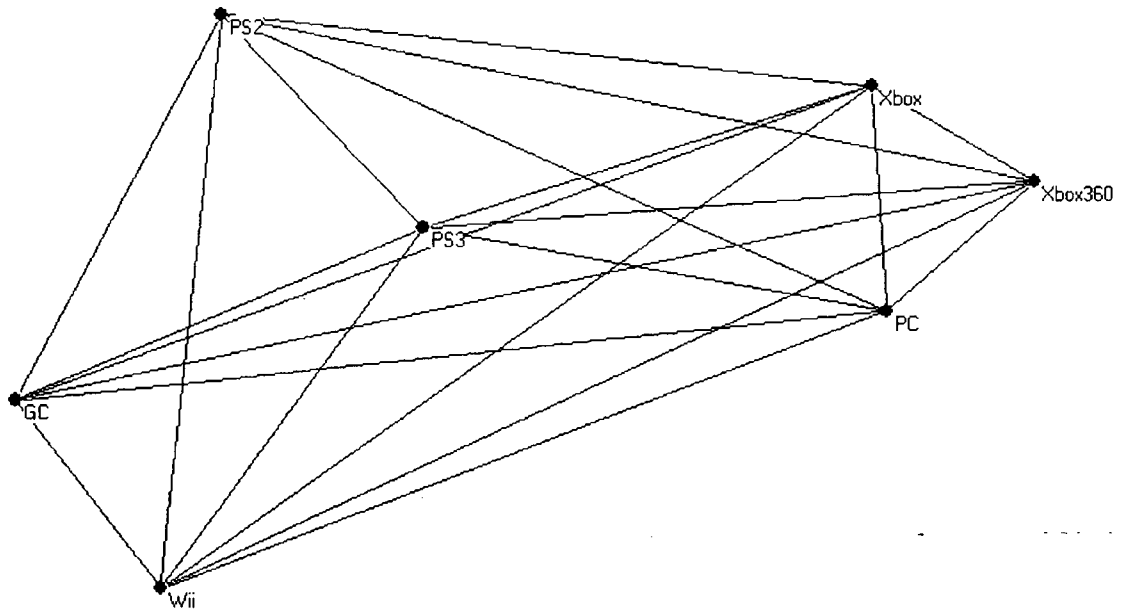
	<b>Prog. Lang.</b>	<b>IDE</b>	<b>Graphics API</b>	<b>Physics API</b>	<b>Sound API</b>	<b>Total</b>
<b>PC-PS2</b>	0	1	1	1	1	0.8
<b>PC-Xbox</b>	0	0	0	0	0	0
<b>PC-GC</b>	0	1	1	1	1	0.8
<b>PC-Xbox360</b>	0	0	0	0	0	0
<b>PC-PS3</b>	0	0	1	1	1	0.6
<b>PC-Wii</b>	0	1	1	1	1	0.8
<b>PS2-Xbox</b>	0	1	1	1	1	0.8
<b>PS2-GC</b>	0	1	1	1	0	0.6
<b>PS2-Xbox360</b>	0	1	1	1	1	0.8
<b>PS2-PS3</b>	0	0.5	0	1	0	0.3
<b>PS2-Wii</b>	0	1	1	1	0	0.6
<b>Xbox-GC</b>	0	1	1	1	1	0.8
<b>Xbox-Xbox360</b>	0	0	0	0	0	0
<b>Xbox-PS3</b>	0	0	1	1	1	0.6
<b>Xbox-Wii</b>	0	1	1	1	1	0.8
<b>Xbox360-PS3</b>	0	0	1	1	1	0.6
<b>Xbox360-Wii</b>	0	1	1	1	1	0.8
<b>PS3-Wii</b>	0	1	0.5	1	0	0.5
<b>GC-Xbox360</b>	0	1	1	1	1	0.8
<b>GC-PS3</b>	0	1	0.5	1	0	0.5
<b>GC-Wii</b>	0	0	0	1	0	0.2

**Table 13. Platform OS and Hardware Distance (Hardware<sub>jk</sub> = 0, 0.5 or 1)**

	OS	CPU	GPU	I/O Devices	Memory	Total
PC-PS2	1	1	1	1	0	0.8
PC-Xbox	0	0	1	1	0	0.4
PC-GC	1	1	1	1	0	0.8
PC-Xbox360	0	1	0	1	0	0.4
PC-PS3	1	1	1	1	0	0.8
PC-Wii	0	1	0	1	0	0.4
PS2-Xbox	1	1	1	0	1	0.8
PS2-GC	1	1	1	0	0.5	0.7
PS2-Xbox360	1	1	1	0	1	0.8
PS2-PS3	1	1	1	0	1	0.8
PS2-Wii	1	1	1	1	1	1
Xbox-GC	1	1	1	0	0.5	0.7
Xbox-Xbox360	0	1	1	0	1	0.6
Xbox-PS3	1	1	0	0	1	0.6
Xbox-Wii	1	1	1	1	0.5	0.9
Xbox360-PS3	1	0.5	1	0	0.5	0.6
Xbox360-Wii	1	0.5	0	1	1	0.7
PS3-Wii	1	0.5	1	1	1	0.9
GC-Xbox360	1	0.5	1	0	1	0.7
GC-PS3	1	0.5	1	0	1	0.7
GC-Wii	0	0	0	1	0.5	0.3

**Table 14. Total Platform Architecture Distance (Hardware<sub>jk</sub> and API<sub>jk</sub> = 0, 0.5 or 1)**

	API/Dev Env	CPU/OS	w <sub>jk</sub> = 90/10	w <sub>jk</sub> = 80/20	w <sub>jk</sub> = 60/40
PC-PS2	0.8	0.7	<b>0.79</b>	<b>0.78</b>	<b>0.76</b>
PC-Xbox	0	0.6	<b>0.06</b>	<b>0.12</b>	<b>0.24</b>
PC-GC	0.8	0.6	<b>0.78</b>	<b>0.76</b>	<b>0.72</b>
PC-Xbox360	0	0.9	<b>0.09</b>	<b>0.18</b>	<b>0.36</b>
PC-PS3	0.6	0.6	<b>0.6</b>	<b>0.6</b>	<b>0.6</b>
PC-Wii	0.8	0.7	<b>0.79</b>	<b>0.78</b>	<b>0.76</b>
PS2-Xbox	0.8	0.9	<b>0.81</b>	<b>0.82</b>	<b>0.84</b>
PS2-GC	0.6	0.7	<b>0.61</b>	<b>0.62</b>	<b>0.64</b>
PS2-Xbox360	0.8	0.7	<b>0.79</b>	<b>0.78</b>	<b>0.76</b>
PS2-PS3	0.3	0.3	<b>0.30</b>	<b>0.30</b>	<b>0.30</b>
PS2-Wii	0.6	0.7	<b>0.61</b>	<b>0.62</b>	<b>0.64</b>
Xbox-GC	0.8	0.7	<b>0.79</b>	<b>0.78</b>	<b>0.76</b>
Xbox-Xbox360	0	0.6	<b>0.06</b>	<b>0.12</b>	<b>0.24</b>
Xbox-PS3	0.6	0.6	<b>0.6</b>	<b>0.6</b>	<b>0.6</b>
Xbox-Wii	0.8	0.9	<b>0.81</b>	<b>0.82</b>	<b>0.84</b>
Xbox360-PS3	0.6	0.6	<b>0.6</b>	<b>0.6</b>	<b>0.6</b>
Xbox360-Wii	0.8	0.7	<b>0.79</b>	<b>0.78</b>	<b>0.76</b>
PS3-Wii	0.5	0.9	<b>0.54</b>	<b>0.58</b>	<b>0.66</b>
GC-Xbox360	0.8	0.7	<b>0.79</b>	<b>0.78</b>	<b>0.76</b>
GC-PS3	0.5	0.7	<b>0.52</b>	<b>0.54</b>	<b>0.58</b>
GC-Wii	0.2	0.3	<b>0.21</b>	<b>0.22</b>	<b>0.24</b>



**Figure 10. Architectural Distance between Platforms**



## VI. RESULTS

### 6.1 Results of Analysis 1

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 15. 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). This is consistent with the explanation that developers that are locked into a single platform technology find it difficult to adapt and migrate to newer platforms and hence 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.

The control variables introduced in the analysis also produce some interesting insights. At the developer level (case specific variables), the impact of the developer age and centrality on the likelihood of porting were analyzed. Developer age as a control focuses on how the age of a developer influences their ability to adapt to technological shifts. The coefficient of developer age in Model 3 was 0.0004, which was positive and significant ( $p < 0.001$ ) for the unique tie nest. The interpretation is that with respect to developers that supported platforms through ported ties, developers that supported platforms through unique ties were 0.0004 times older. This implies that the thesis that developer age increased the likelihood of support through ported ties was not supported. A potential explanation for this could be that as developers grew older they were locked into a single platform technology and unable to adapt to technology shifts. Thus, they

were less likely to migrate to new platforms and consequently were more likely to support extant platforms through unique ties.

The impact of developer centrality on the likelihood of support to platforms through ported ties was also controlled for. The coefficient of developer centrality was 16.76, which was positive but not significant for the unique tie test. The implication is that as there is not much difference in resource positions for developers that support platforms through unique or ported ties.

At the alternative-specific level (platform specific level) the impact of platform age on likelihood of support through ported ties was controlled for. The coefficient of platform age was -0.03, which was negative and significant ( $p < 0.001$ ) implying that developers were more likely to port to older platforms than newer platforms.

At the case specific level, the impact of firm level characteristics of the source platform was also controlled for. In this analysis, Microsoft was set as the base firm and the likelihood of support through unique ties relative to ported ties was evaluated for Sony and Nintendo with respect to Microsoft. The analysis revealed that the choice of the source platform firm influenced the pattern of support extended by a developer. Specifically, relative to Microsoft, developers that launched a game title first on Nintendo or Sony platforms were more likely to support them through unique ties (for Nintendo, the coefficient was 1.34 which was positive and significant ( $p < 0.1$ ) and for Sony, the coefficient was 0.92 which was positive and significant ( $p < 0.1$ ). In other words, developers were most likely to launch unique game titles on Sony and Nintendo relative to Microsoft platforms. This finding highlights the importance of architectural strategies

employed by platform providers to attract developers to their platforms. For instance, all the Microsoft platforms (PC, Xbox and Xbox360) were based on common standards that increased the ease with which developers could port across them. On the other hand, the Sony PS3 was based on a radically different architecture from those of the competing platforms making it difficult for developers to support the PS3 and competing platforms simultaneously.

In order to test the robustness of the analysis, I also re-estimated the models with the alternative measures of platform architecture distance. The results with each of these alternative measures were consistent with the results obtained from Model 3, thereby, underscoring the robustness of the analysis. The results with the alternative architectural distance measures are included in Appendix A.

**Table 15. Results for Analysis 1**

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

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

## **6.2. Results of Analysis 2**

In this section, the results of the nested logit model with separate nests for PC, sixth and seventh generation platforms are presented. The analysis was built in stages through a series of three models. In Model 1, the case specific variables (generation of the source platform) and the alternative specific (platform-specific) control variables

were included. Model 2 added the variable of platform centrality and finally, Model 3 included the platform architecture distance measure. All models were estimated using Stata 10.0. The results discussed below are drawn from Model 3.

The goal of Analysis 2 is to understand the role of architectural shifts in influencing the extent of support to platforms through ported titles. As a result, the analysis focused on explaining how patterns of porting evolved during transition of video game consoles from the sixth to the seventh generation. The dynamics of the patterns of porting are reflected in the results obtained for the case specific variables – the generation of the source platform in this study. The results of the case specific variables are shown in Table 16. These results reflect the patterns of porting with the sixth generation nest as the base and the PC as the base relative to which the generation specific dummies for the sixth and seventh generation consoles were evaluated.

The coefficient for the sixth generation dummy in the PC nest was -0.37 which was negative and weakly significant ( $p < 0.1$ ). The interpretation is that the likelihood of porting from a sixth generation console to another sixth generation console is higher than the likelihood of porting from a sixth generation console to the PC platform. The coefficient for the seventh generation dummy in the PC nest was 0.81 which was positive and significant ( $p < 0.05$ ) which implies that the likelihood of porting from a seventh generation platform to the PC was greater than the likelihood of porting from a seventh generation platform to a sixth generation platform. The coefficient for the sixth generation dummy in the seventh generation nest was 0.62 which was positive and significant ( $p < 0.01$ ) implying that the likelihood of porting from the sixth generation



platform to a seventh generation platform was greater than the likelihood of porting between sixth generation platforms. Finally, the coefficient of for the seventh generation dummy in the seventh generation nest was 1.57 ( $p < 0.001$ ) which implies that the likelihood of porting between seventh generation consoles was greater than the likelihood of porting from the seventh generation to the sixth generation.

The results of the analysis for Analysis 2 broadly reflect the patterns of porting during the transition from the sixth to the seventh generation. The first key implication of the results obtained from the analysis is that the extent of porting increased as video game consoles underwent technological shifts and transitioned to the seventh generation. The second key implication is that in addition to the extent of porting increasing, specifically the extent of porting between the seventh generation consoles and the PC platform increase. 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.

The results for the alternative specific variables (shown in Table 17) also broadly support Hypotheses 2, 3, 4a, 4b and 5 regarding the impact of platforms attributes on the likelihood of support through ported ties. The coefficient for the platform architecture distance was -1.17, which was negative, and significant ( $p < 0.001$ ) thereby, consistent with the prediction that as architecture distance between the source and target platforms decreased, the likelihood of porting increased. The coefficient for platform centrality was -4.92, which was negative and significant ( $p < 0.001$ ). This was consistent with hypothesis 4 that as developers were less likely to port to dominant platforms. The impact of genre

distance (-30.9,  $p < 0.001$ ) and platform porting centrality (2.98,  $p < 0.001$ ) was also significant. Among the control variables, the impact of developer prior ties was also positive and significant (2.67,  $p < 0.001$ ) indicating that developers were most likely to support those platforms with ported ties, which they had supported in the past. Also, the firm specific effects for Nintendo (0.42,  $p < 0.01$ ) was stronger than that of Microsoft while the firm specific effects for Sony were not significantly different from Microsoft. The analysis also controlled for the impact of the launch of new generation platforms on the likelihood of porting to the previous generation. The coefficient for the variable for the launch of newer generation Nintendo platform was not significant, implying that relative to Microsoft, the launch of the Wii did not significantly alter the patterns of porting to the GC platform. The coefficient for the variable for the launch of the newer generation Sony platform was 1.06 ( $p < 0.001$ ), which implied that relative to Microsoft, the launch of the Sony PS3 platform increased the likelihood of porting to the previous generation PS2 platform.

**Table 16. Results of Case Specific (Generation specific) Variables for Analysis 2**

	Model 1	Model 2	Model 3
<i>Generation-specific (6<sup>th</sup> generation as base)</i>			
<i>PC</i>			
<b>Sixth Generation</b>	-2.14*** (0.28)	-0.45 (0.54)	-0.37† (0.22)
<b>Seventh Generation</b>	0.19 (0.35)	1.54** (0.54)	0.81* (0.29)
<i>7<sup>th</sup> Generation</i>			
<b>Sixth Generation</b>	-0.28 (0.28)	-0.07 (0.28)	0.62** (0.26)
<b>Seventh Generation</b>	1.59*** (0.29)	1.69*** (0.29)	1.57*** (0.28)
<b>Log-likelihood</b>	-1345.95	-1338.80	-1310.71
<b>Δ(-2LL)</b>		12.40***	28.09***

N = 5112; \*\*\*p<0.001; \*\*p<0.01, \*p<0.05; †p<0.1

**Table 17. Results of Platform-specific variables for Analysis 2**

	<b>Model 1</b>	<b>Model 2</b>	<b>Model 3</b>
<i>Platform-Specific</i>			
<b>Platform Centrality</b>		-3.63*** (1.00)	-4.92*** (1.07)
<b>Platform Architecture Distance</b>			-1.17*** (0.21)
<b>Platform Genre Distance</b>	15.66** (5.48)	11.85* (5.29)	-30.89** (9.41)
<b>Platform Age</b>	0.01* (0.0064)	0.02** (0.007)	0.008 (0.007)
<b>Platform Porting Centrality</b>	0.79* (0.38)	1.62** (0.50)	2.98*** (0.64)
<b>Sony</b>	-0.35*** (0.08)	-0.02 (0.12)	0.21 (0.14)
<b>Nintendo</b>	-0.28** (0.09)	0.03 (0.18)	0.42* (0.15)
<b>Launch of New Generation (Sony)</b>	0.76*** (0.14)	0.71*** (0.14)	1.06*** (0.18)
<b>Launch of New Generation (Nintendo)</b>	.006 (0.18)	0.03 (0.18)	-0.23 (0.25)
<b>Log-likelihood</b>	-1345.95	-1338.80	-1310.71
<b>Δ(-2LL)</b>		12.40***	28.09***

N = 5112; \*\*\*p<0.001; \*\*p<0.01, \*p<0.05; †p<0.1

### **6.3 Results of Analysis 3**

In this section, the results of the nested logit model with separate nests for the platform firms Microsoft, Sony and Nintendo are presented. The PC, Xbox and Xbox360 platforms are included in the Microsoft nest; the PS2 and PS3 platforms are in the Sony nest and finally, the GC and Wii platforms are included in the Nintendo nest. The analysis was built in stages through a series of three models. In Model 1, the case specific variables (firm developing the source platform) and the alternative specific (platform-specific) control variables were included. Model 2 added the variable of platform centrality and finally, Model 3 included the platform architecture distance measure. All models were estimated using Stata 10.0. The results discussed below are drawn from Model 3.

The goal of analysis 3 was to understand how the architectural strategies of individual platform firms influenced the extent of support to their platforms through ported titles. As a result, the analysis focused on explaining how patterns of porting evolved for each platform as they transitioned their consoles from the sixth to the seventh generation. The dynamics of the patterns of porting are reflected in the results obtained for the case specific variables – the firm launching the source platform in this study. The results of the case specific variables are shown in Table 18. These results reflect the patterns of porting with the Microsoft nest as the base. Microsoft was also the base relative to which the source platform firm specific dummies for Sony and Nintendo were evaluated.

The coefficient for Sony in the Sony nest was -3.41, which was negative and significant ( $p < 0.001$ ). The interpretation is that the likelihood of porting from a Sony platform to another Sony platform is lower than the likelihood of porting from a Sony platform to a Microsoft platform. The coefficient for the Nintendo dummy in the Sony nest was -1.62, which was negative and significant ( $p < 0.001$ ) which implies that the likelihood of porting from a Nintendo platform to a Sony platform was less than the likelihood of porting from a Nintendo platform to a Microsoft platform. The coefficient for the Sony dummy in the Nintendo nest was -1.68, which was negative and significant ( $p < 0.001$ ) implying that the likelihood of porting from a Sony platform to a Nintendo platform was greater than the likelihood of porting from a Sony platform to a Microsoft platform. Finally, the coefficient of for the Nintendo dummy in the Nintendo nest was -2.81 ( $p < 0.001$ ), which implies that the likelihood of porting from a Nintendo platform to another Nintendo platform was lower than the likelihood of porting from a Nintendo platform to a Microsoft platform.

The results of the analysis for Analysis 3 broadly highlight the role of Microsoft in driving the patterns of porting in gaming platforms. The key implication of the results obtained from the analysis is that the likelihood of porting to a Microsoft platform was significantly higher than the likelihood of porting to either a Nintendo or a Sony platform. Even as Sony and Nintendo launched newer generation platforms, their developers were less likely to migrate to the newer generation and support those platforms through ported titles. Instead they were more inclined to port their existing titles to the Microsoft.

The results for the alternative specific variables (shown in Table 19) are also broadly consistent with those obtained from Analysis 1 and Analysis 2. The analysis here allowed testing Hypotheses 2 through 5 pertaining to the impact of platforms attributes on the likelihood of support through ported ties. The coefficient for the platform architecture distance was -4.20, which was negative, and significant ( $p < 0.001$ ) thereby, consistent with the prediction that as architecture distance between the source and target platforms decreased, the likelihood of porting increased. However, the coefficient for platform centrality was not significant indicating that Hypothesis 2 is not supported. The impact of genre distance (1.25) was not significant while platform-porting centrality (1.42,  $p < 0.051$ ) was significant. The impact of platform age (-0.04,  $p < 0.001$ ) was negative and significant. The impact of developer prior ties was also positive and significant (3.54,  $p < 0.001$ ) indicating that developers were most likely to support those platforms with ported ties, which they had supported in the past. Also, the firm specific effects for Nintendo (1.33,  $p < 0.001$ ) and Sony (1.16,  $p < 0.001$ ) was stronger than that of Microsoft. The analysis also controlled for the impact of the launch of new generation platforms on the likelihood of porting to the previous generation. The coefficient for the variable for the launch of newer generation Nintendo platform was not significant, implying that relative to Microsoft, the launch of the Wii did not significantly alter the patterns of porting to the GC platform. The coefficient for the variable for the launch of the newer generation Sony platform was 1.24 ( $p < 0.001$ ), which implied that relative to Microsoft, the launch of the Sony PS3 platform increased the likelihood of porting to the previous generation PS2 platform.

**Table 18. Results of case-specific variables for Analysis 3**

	Model 1	Model 2	Model 3
<i>Firm-specific (Microsoft as base)</i>			
<i>Sony</i>			
<b>Sony</b>	0.11 (0.31)	-0.51*** (0.16)	-3.41*** (0.48)
<b>Nintendo</b>	1.29*** (0.18)	0.76* (0.27)	-1.62*** (0.38)
<i>Nintendo</i>			
<b>Sony</b>	0.96*** (0.21)	0.47* (0.24)	-1.68*** (0.38)
<b>Nintendo</b>	2.09** (0.73)	1.49*** (0.28)	-2.81*** (0.91)
<b>Log-likelihood</b>	-1389.52	-1476.69	-1356.57
<b><math>\Delta(-2LL)</math></b>		0.46	65.44***

N = 5112; \*\*\*p<0.001; \*\*p<0.01, \*p<0.05; †p<0.1



**Table 19. Results for Platform Specific Variables for Analysis 3**

	Model 1	Model 2	Model 3
<i>Platform-Specific</i>			
<b>Platform Centrality</b>		0.42 (0.61)	0.05 (0.55)
<b>Platform Architecture Distance</b>			-4.20*** (0.56)
<b>Platform Genre Distance</b>	-23.72** (6.95)	-26.20** (7.87)	1.25 (7.38)
<b>Platform Age</b>	-0.04*** (0.004)	-0.04*** (0.004)	-0.04*** (0.004)
<b>Platform Porting Centrality</b>	2.51*** (0.72)	2.61*** (0.73)	1.42* (0.56)
<b>Sony</b>	-0.56*** (0.14)	-0.59*** (0.15)	1.16*** (0.26)
<b>Nintendo</b>	-0.88*** (0.18)	-0.88*** (0.18)	1.33*** (0.36)
<b>Launch of New Generation (Sony)</b>	1.13*** (0.25)	1.15*** (0.26)	1.24*** (0.21)
<b>Launch of New Generation (Nintendo)</b>	-0.78 (0.77)	-0.77 (0.76)	-0.75 (0.71)
<b>Log-likelihood</b>	-1389.52	-1389.29	-1356.57
<b>Δ(-2LL)</b>		0.46	65.44***

N = 5112; \*\*\*p<0.001; \*\*p<0.01, \*p<0.05; †p<0.1

## VII. DISCUSSIONS

This dissertation is motivated by the question of how the convergence of disparate platform architectures over time drives the evolution of platform-complementor networks. Academic research on platform-based competition has so far focused primarily on the relationship between the availability of third-party complements and platform success (Gandal, Kende, & Rob, 2000; Nair, Chintagunta, & Dubé, 2004; Clements & Ohashi, 2005). While recent studies within the technology management have examined the actions platform providers and how they can attract third-party complementors to stimulate indirect network effects (Gawer & Cusumano, 2002; Gawer & Henderson, 2007; Eisenmann, 2008; Evans, Hagiu, & Schmalensee, 2008), very few studies have examined the relationship between platform providers and third-party complementors (for example, Venkatraman & Lee, (2004)) and how these interactions evolve with time-varying architectural shifts.

This dissertation builds on the fundamental assertion that the network dynamics of coordination between platform providers and developers are central to platform-based settings. Relationships with complementors, although implicit, provide resources critical to the success of platforms due to inherent mutual dependency between the parties within the system. Platform providers invest significant resources to attract developers to their platform and in turn, third-party developers commit resources to develop software for a platform over time. As a result, developers need to coordinate their choices with the architectural strategies of platform providers and also need to make critical choices about linking with platforms and how to adapt these linkages over time (Venkatraman & Lee,

2004). In addition they also need to make decisions on how to link to these platforms. For instance, when platform providers launch new generations of their technologies, developers need to shift their learning and knowledge along the new technological trajectory (Kash & Rycoft, 2000). Moreover, in addition to adapting to new technological generations launched by the same platform firm, developers also need to adapt to competing platforms launched by other platform providers. Therefore, as new platforms enter the market, in order to support these platforms, the complementor would be likely to incur costs associated with technology specific assets such as middleware and game engines in addition to re-learning the new routines associated with the platform (Afuah & Werner, 2007).

More recently, developers' platform adaptation decisions have begun to transcend traditionally defined platform boundaries. Convergence between previously disparate platforms has occurred as a result of technological shifts such as the launch of common third party standards; the migration of an extant platform firm into a new platform setting and finally, improvements in hardware architecture. This has afforded developers the opportunity to expand their scope and achieve synergies by rapidly adapting to newer platform settings. For instance, the convergence between PC, video game consoles and mobile platforms implies that developers need to continually make linkage decisions that reflect the increasing convergence between these platforms.

The goal of this dissertation has thus been to contribute to our understanding of the evolution of platform-based competition in periods of architectural convergence and explain how networks of coordination between developers and platform providers co-

evolve with architectural shifts. Specifically, the dissertation has sought to address the following questions - how does the heterogeneity in developer attributes influence their ability to adapt to technological shifts? How do developer resource and network positions influence their ability to make the shift to support contemporaneously existing platform architectures through ported ties? How do platform network attributes and architectures contribute to the patterns of platform-complementor network evolution? How do developer attributes and platform architecture strategies co-evolve to alter network dynamics? These questions lie at the intersection of different strands of research—platform-based competition (Shapiro & Varian, 1998); firm capabilities and innovation (Tushman & Anderson, 1986; Klepper, 1996; Sørensen & Toby E. Stuart, 2000) and network perspectives of organizational evolution and growth (Powell, Koput, & Smith-Doerr, 1996; Gulati, Nohria, & Zaheer, 2000). By adopting the perspective of these three distinct streams of research this dissertation recognizes the importance of the interaction between platforms and complementors that is crucial for success in settings of platform-based competition while also recognizing that these networks of relationships need to evolve in the face of technological change.

Each of the questions described previously have been addressed through a set of three analyses. In the first set of analysis in Analysis 1, I attempted to answer the question of how third-party developers respond to architectural shifts in terms of the patterns of support that they extend to competing platforms. The question of how heterogeneity in developer resource and network attributes influence their ability to migrate and support platforms during architectural shifts was specifically addressed. The results of analysis 1

also provided insight into how platform network positions and architectural distance influenced the patterns of support extended to developers. Analysis 2 provided insight into the impact of the architectural transition from the sixth to the seventh generation consoles on patterns of tie formation at the macro-level. Finally, Analysis 3 focused on the patterns of tie formation for the individual platform firms as they transitioned to the seventh generation. The implications of the findings from each of these studies are discussed below.

### **7.1. Developer attributes and adaptation and support to platforms**

Analysis 1 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 developer attributes – age, centrality and 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 relationship between developer age and the likelihood of platform support through ported or unique ties was analyzed. The thesis was that as developers aged, they were more likely to get embedded in their existing routines (Ven, 1986; Ginsberg & Venkatraman, 1992; Klepper, 1996; Sørensen & Stuart, 2000) and hence look for opportunities to reapply their knowledge and code base in new settings. As a result, they

would be more likely to want to port their existing game titles to newer competing platforms. The analysis, however, revealed that developer age positively influenced the likelihood of support through unique ties, thus implying that as developers aged they were less likely to port their existing titles to newer platforms. One possible explanation could be that, as developers grew older they were less likely to be able to adapt and migrate to newer platforms. Given their inability to support multiple platforms simultaneously, their pattern of support extended to platforms would be through unique ties.

The relationship between developer centrality and the pattern of support extended to platforms was also examined. Developer centrality (measured as the proportion of titles released by a developer) served as a proxy of the size or dominance of developers. The thesis was that as their centrality increased, developers would be more likely to support platforms through ported ties. The rationale for this hypothesis was that as organizations get bigger due to their distinctive competencies, they are less willing to embrace new innovations that threaten to alter their established routines and interrelationships (Hannan & Freeman, 1989; Levinthal & March, 1993). Larger, more successful firms continue to look for opportunities to reapply their routines and know-how in novel settings (Starbuck, 1983) that reduces their ability to develop new and unique innovations. However, the results of the empirical analysis in Analysis 1 revealed that there was not any significant difference in terms of dominance between developers that supported platforms through ported ties and those who supported platforms through unique ties. In other words, large developers were equally likely to support platforms

through ported and unique game titles. A potential explanation for this paradox is that larger developers are likely to have greater access to resources to invest in learning new platform technologies and could afford to support multiple platforms with unique sets of game titles.

The single most dominant factor in driving the choice of support to a platform was the extent to which developers' were embedded in 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)) 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 would 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. Nonetheless, the lack of findings in the relationship between developer resource positions (reflected through their age and size) and choice of support coupled with strong support for the relationship between developer network positions (embeddedness) and patterns of tie formation underscores the importance of adopting a network perspective in

platform-based settings. 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. Thus, the impact of platform centrality, platform porting centrality, platform architectural distance and platform genre distance on the choice of platform to support through ported ties was studied. The impact of platform centrality on the likelihood of support through ported ties was positive and significant. This implied that developers supported platforms in superior market positions with ported ties. This was contrary to the prediction that 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, complementors are more likely to migrate to them first and support them through unique titles. The notion that developers will be more likely to migrate to less dominant platforms once the future market viability of these platforms has been determined was rejected. 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.

The salience of the architectural distance between platforms in driving the sequence of launch decisions of developers was also highlighted in the analysis. The notion that the propensity of developers, to form ties with platforms that were architecturally similar to platforms that they have 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). This was especially evident in the patterns of porting between the PC and Xbox360 platforms. Microsoft with the launch of the XNA architecture and the development of the DirectX APIs had created an environment that provided developers with synergies and allowed them to simultaneously support these platforms with minimal reinvestment associated with re-learning and re-writing parts of the game code. Developers were thus more likely to port between Xbox360 and PC rather than between Xbox360 and its seventh generation console counterparts – PS3 and Wii.

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 migrate across platforms and support them through ported ties. The role of converging platform architectures in decreasing the impact of developer embeddedness in influencing their choice patterns was thus, supported through the results.

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.

## **7.2. Platform architecture convergence and platform-complementor network**

### **evolution at the macro level**

In this study, I 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 20 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.

**Table 20. 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	+	-	+

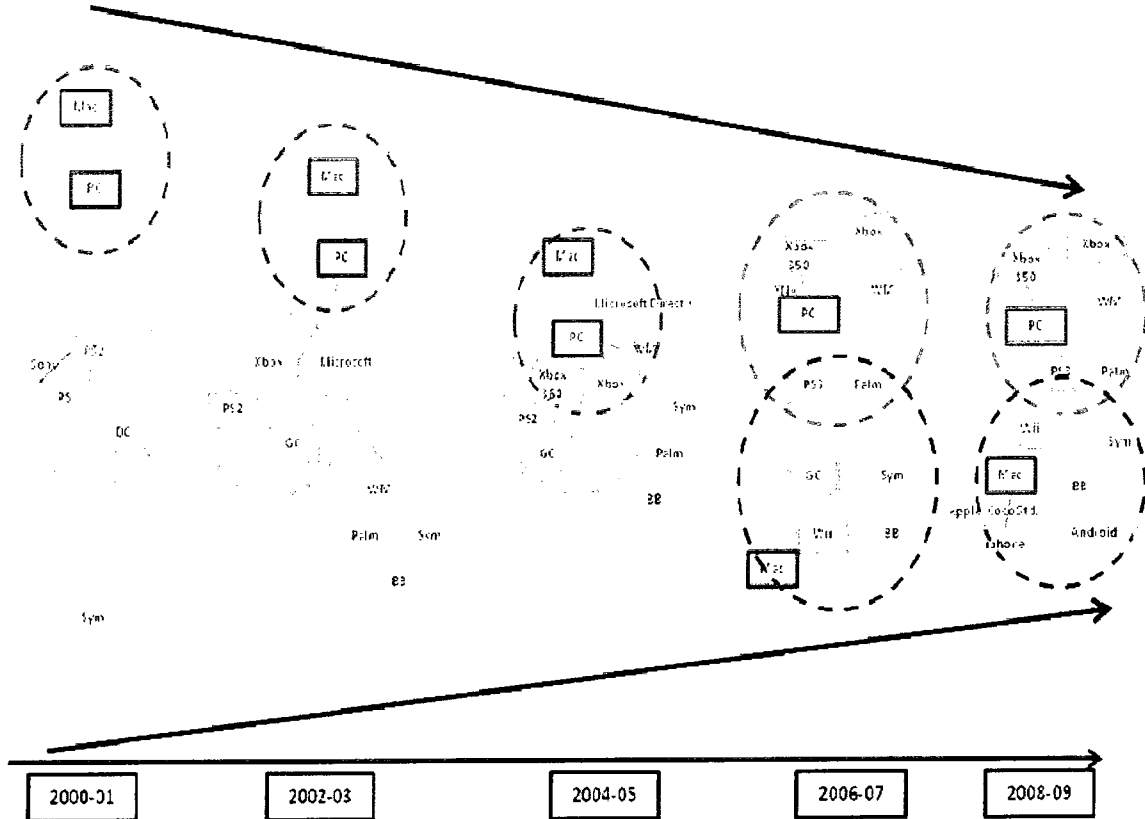
The change in the patterns of tie formation – specifically the overall increase in the level of ported ties could be attributed to the architectural shifts resulting from the transition to the seventh generation consoles. As consoles have improved technologically over time, so has the availability of 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. For instance, a large number

of exclusive PC developers such as Simon & Schuster, Groove Games and Enlight Software migrated to and contemporaneously launched the same game titles across the PC and Xbox360 platforms. 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 dissertation 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. As a result, 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. These patterns of migration and tie formation across previously disparate consoles have been further enhanced by overall improvements in hardware architecture that make it possible for platform firms to enter into new platform markets in the first place. Figure 11 shows the patterns of architectural convergence in the PC, video games and mobile sectors.



**Figure 11. Architectural Convergence in PC, Video Game and Mobile Platforms**

### **7.3. Architectural Shifts and Dynamics of Platform-Complementor Network**

#### **Interactions: Platform Firm Level Effects**

The set of analyses thus far has focused on: one, how developer and platform provider actions co-evolve to influence the patterns of tie formation and two, how the architectural shift from the sixth to the seventh generation consoles influenced the network evolution at the macro level. This dissertation also sought to provide insight into the architectural strategies of individual platform firms (Microsoft, Sony and Nintendo) and how these strategies influenced the patterns of support extended to them by their

ecosystem of third-party game developers.

Table 21 presents the extent of porting activity between each platform firm as they transition their platforms from the sixth to the seventh generation. The extent of titles ported between the platform firms is ranked from 1 to 3 where 1 is the lowest amount of porting and 3 indicates the highest amount of porting. The results reveal a number of interesting insights. One, relative to Microsoft, developers that launched a game title first on Nintendo or Sony platforms were more likely to support them through unique ties. Therefore, Sony and Nintendo platforms had a larger set of exclusive titles relative to Microsoft platforms. Two, relative to Microsoft, the launch of the Wii did not significantly alter the patterns of porting to the GC platform while the launch of the Sony PS3 platform increased the likelihood of porting to the previous generation PS2 platform. Three, the likelihood of porting to a Microsoft platform was significantly higher than the likelihood of porting to either a Nintendo or a Sony platform. Even as Sony and Nintendo launched newer generation platforms, their developers were less likely to migrate to the newer generation and support those platforms through ported titles. Instead they were more inclined to port their existing titles to the Microsoft.

Each of these insights corroborates the impact of individual firm architectural choices in influencing how and why third-party developers supported them. I, now focus on the dynamics of the platform complementor interactions for each platform firm.



**Table 21. Firm Level Patterns of Porting**

		Target Platform Firm		
		Sony	Nintendo	Microsoft
Source Platform Firm	Sony	1	2	3
	Nintendo	2	1	3
	Microsoft	1	2	3

### 7.3.1. Sony and the evolution of its network of complementors

The key findings with respect to the Sony complementor network were that developers that launched titles first on a Sony platform were less likely to port those titles to other competing platforms. Also, as Sony transitioned from PS2 to PS3, the extent of porting across these platforms was lower than the extent of porting between PS2 and Microsoft platforms. Delving into the details of the patterns of evolution of Sony's ties with its complementors provided additional insight into the role of Sony's architectural choices and its market position. The introduction of the PS2 by Sony in October 2000 launched the sixth generation of video game consoles. In the first six months since the launch of the PS2, its existing network of complementors (from the fifth generation Playstation console) engaged in a small percentage (7.6%) of ported ties in the form of cross-generation porting from Playstation while the majority of ties (92.4%) were unique ties in the form of new games. Moreover, there was no overlap in game titles between the PS2 and PC platforms during this time period. With the entry of competing sixth generation platforms - GC and Xbox, PS2 was now supported with 12.2% ported ties and

84.2% unique ties by its existing network of complementors. Interestingly, a majority of the ported ties were with developers such as Electronic Arts that were less embedded and straddled all the existing platforms, which supports the notion that less embedded developers are able to make the transition to newer platforms sooner and support them through ported ties. Additionally, the radically different architecture of the PS2 relative to the PC and other sixth generation platforms along with its proprietary developer SDKs and toolkits helped explain the pattern of tie formation as being primarily unique ties as opposed to ported ties.

As Sony transitioned from PS2 to PS3, the support to the PS3 platform by Sony's existing set of complementors that migrated was through a combination of ported ties and unique ties (38% and 62% respectively). However, the transition for Sony's network of complementors to the PS3 network was much slower with only a few developers (specifically only EA and Activision in addition to in-house developer of Sony) migrating and even then providing only symbolic support to the platform through the launch of only one or two games. On the other hand, PS2 was able to establish a unique network of ties with a subset of small developers like The Game Factory, Valcon, and ValuSoft. While these developers only developed a fewer number of game titles relative to larger ones like EA and Activision, the fact that they provided unique support to PS2 through niche games allowed PS2 to maintain competitive differentiation from Xbox and GC. However, these small developers though highly embedded in the Sony network were unable to migrate to the radically different PS3 architecture. Since the migration from PS2 to PS3 called for radical change in the repertoires and competencies of developers, the focus of

developers was likely to adopt a wait-and-see approach before committing to an innovation that threatened the status quo (Ginsberg & Venkatraman, 1992). The dominant market position of the PS2 in the sixth generation of consoles also potentially contributed the lack of migration of developers to the PS3 platform. This was especially evident from the finding that as Sony transitioned to the PS3, its developers were unwilling to change the status quo and instead increased their support to the PS2 platform through ported ties. This supports recent studies on network inertia (for example, Kim, Oh, & Swaminathan, (2006)) that refers to the inability of certain organizations to change or dissolve its inter-organizational ties due to organizational resistance. In summary, the evolution of the Sony complementor network clearly demonstrates how the architectural choices and the market positions of platforms influence the pattern of tie formation and subsequent network evolution.

### **7.3.2. Nintendo and the evolution of its network of complementors**

The overall patterns of evolution of ties for Nintendo with its complementors were broadly similar to the patterns exhibited by Sony and its complementor ecosystem. Specifically, like Sony's developers, developers that supported the Nintendo platform first were less likely to port these titles to other platforms. Also, the likelihood of porting to a Nintendo platform was significantly lower than the likelihood of porting to Microsoft platforms. To further understand how the architectural shift from the sixth generation GC to the seventh generation Wii influenced network evolution, the dominant patterns of tie formation were observed longitudinally over time. Nintendo entered the sixth generation consoles market with the launch of the GC platform Like the PS2 network, the GC

console also attracted developers who were part of the ecosystem of previous generation consoles developed by Nintendo (100% of its ties were repeat ties with incumbents from the previous generation Nintendo DS). Also, like GC, the Wii networks evolved through ties with the large, dominant complementors such as EA and Activision. However despite the fact that the large developers migrated to the Wii platform, its network also evolved primarily through unique ties (80% respectively).

The patterns of network evolution between Nintendo and its complementors reiterate the role of the platform firm's architectural strategies in driving network dynamics. The inability of developers to port their titles from the GC or the Wii to other competing platforms especially, the Wii was based on a fundamentally different architecture than competing platforms such as Xbox360, PS3 and PC. Its memory management capabilities and its unique motion control sensors made it a radically different platform for developers to support. Moreover, the game genres and hence the user base supported by Wii was radically different from that of PS3 and Xbox360 which potentially could explain why developers were unwilling to port games launched first on the Wii. Again, the relative difference in extent of porting between GC and Wii and between the Microsoft platforms was due to the fact that unlike Microsoft, Nintendo did not provide common development toolkits that supported simultaneous development on the Wii and GC platforms.

### **7.3.3. Microsoft and the evolution of its network of complementors**

The entry of Microsoft into the video game console space with the launch of the Xbox played a key role in altering the dynamics of network evolution. Game developers

that launched games on a Microsoft platform first were most likely to port those titles to competing platforms. Moreover, the analysis in Study 3 also revealed that in addition to porting from Microsoft platforms, developers were also most likely to port to Microsoft platforms. Further, the extent of porting between PC, Xbox and Xbox360 was significantly higher than the extent of porting between PS2 and PS3 and between GC and Wii. Together these findings demonstrate that the primary pattern of network evolution for Microsoft was through ported ties. A longitudinal analysis of the evolution of the patterns of tie formation broadly supported these findings. As Microsoft entered the sixth generation video game consoles with Xbox, it sought to form ties with the largest and most dominant developers like Electronic Arts and Konami during the early stages of its entry. Moreover, Xbox was supported primarily by (68.5%) ported ties. As the sixth generation consoles matured and the seventh generation was launched with the introduction of the Xbox360, 71% of its ties were with incumbents who were part of the Xbox network. A large percentage of its ecosystem of developers was able to make this transition due to the architectural similarity (both in terms of hardware and development tools provided to developers) between the PC, Xbox and Xbox360 platforms. Interestingly 42% of all Xbox360's ported ties were with the previous generation Xbox platform.

Additionally, after the launch of the Xbox 360 console, Microsoft was also able to bring in another set of developers from its ecosystem in the PC network to counter the threat posed by the introduction of Sony PS3 and Nintendo Wii. The ability of developers to migrate from the PC network to the video game network was made easier with 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 finding asserts the role of software architecture in driving the dynamics of network evolution by enabling network migration not only within and across technological generations but also across related yet disparate platform settings.

The migration of complementors from the PC network to the video game network demonstrates both the role of technological characteristics and network characteristics in determining the structure of the platform-complementor network (Funk 2008). Theories of technological evolution have emphasized the role of skills, capabilities and routines in determining a firm's ability to successfully transition to a new technological order (Anderson & Tushman, 1990; Henderson & Clark, 1990). Our findings further extend this line of research by demonstrating that existing know-how in a different but related platform setting can be effectively applied to new settings. Moreover, as predicted by network research (Granovetter, 1985; Gulati, 1995a; Gulati, 1999) that embeddedness plays a strong role in determining future network partners, it was found that Microsoft relied on its old network partners (in the PC industry) to develop unique game titles across its Xbox platform. The fact that these complementors had worked with Microsoft before meant that they would be able to rapidly develop games for the Xbox platform without having to focus on the development of trust that is typically the focus of new relationships (Gulati, 1995a; Uzzi, 1997).

## **7.4. Consequences of Architectural Shifts for Third-Party Developers: A Case**

### **Analysis of Electronic Arts and Valve Software**

Architectural shifts that drive convergence have important consequences for third-party developers that support platforms. They present an opportunity for developers to extend their reach and support multiple platforms that transcend platform boundaries simultaneously. Moreover, in order to compete effectively, developers need to adapt to newer platform settings and realign their knowledge and routines to the shifting platform architectures. It is a key contention of this dissertation that there is considerable heterogeneity in how, when and why developers adapt to these architectural shifts. It was argued that ability of developers to migrate to newer platforms depends to a large extent on developer attributes including their age and size and also on their prior experience with certain platform architectures. Developer attributes in addition to influencing their ability to migrate also co-evolve with platform architectures to influence the patterns of support extended to platforms. 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. For instance, developers are more likely to launch game titles on platforms that are more similar to platforms they have supported in the past. They are also more likely to port their existing applications to newer platforms rather than launch exclusive titles across 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. Therefore, in this section, I analyze in detail the strategic actions of two complementors – Electronic Arts and Valve Software over time to understand how they respond to architectural shifts. There is significant heterogeneity between these firms in terms of the attributes, resource positions and relative network positions, thereby allowing us to specifically understand how each of these firms responded to architectural shifts in terms of the support that they extended to platforms.

#### **7.4.1. Responses to Architectural Shifts by Electronic Arts and Valve Software**

Electronic Arts is a large publisher, distributor and developer of video games that was founded in 1982. While it started out as a home computing games firm primarily focused on developing games for the PC, by the 1990s it had begun to support video game consoles. Its transition to consoles was largely enabled due to its acquisition of several successful developers (Kohler, 2008). By the early 2000s, Electronic Arts had become one the largest video game companies developing and publishing successful games such as *Need for Speed*, *Medal of Honor* and the *Madden NFL Series* in almost every game genre.

By the launch of the sixth generation of video game consoles, Electronic Arts had established itself successfully as a console and PC game developer. Its significant growth allowed it to invest significant resources in developing its own proprietary game engines



to assist in cross-platform game development. Indeed, a large part of the success of Electronic Arts has been attributed to its strategy of platform-agnostic development.

Table 22a demonstrates growth of Electronic Arts from November 2000-November 2008 – the time period of focus in this dissertation. Specifically, the evolution of the number of game titles, the average embeddedness and the porting patterns of Electronic Arts since the launch of the sixth generation video game consoles in November 2000 is shown. The numbers clearly demonstrate some interesting patterns. As Electronic Arts made the shift from supporting the sixth to seventh generation consoles, it was not significantly influenced by the entry of Microsoft with its Xbox platform and also by the launch of the XNA architecture. This is demonstrated by its fairly consistent embeddedness between November 2003 and November 2005 (35%, 35% and 34% in November 2003, November 2004 and November 2005 respectively). This indicated that its extent of cross-platform support was not significantly altered by the architectural shift towards convergence. However, the extent of porting across platforms increased from November 2004 to November 2005 from 70% to 82% indicating its “follow-the-trend” response to competitive pressures to launch cross-platform games.

**Table 22a. Evolution of Patterns of Support for Electronic Arts**

	Nov-00	Nov-01	Nov-02	Nov-03	Nov-04	Nov-05	Nov-06	Nov-07	Nov-08
Number of titles launched	40	49	78	111	136	164	191	222	259
Embeddedness	0.59	0.46	0.34	0.35	0.35	0.34	0.32	0.26	0.23
Proportion of titles ported	0.13	0.55	0.72	0.73	0.71	0.82	0.86	0.90	0.88

Valve software is a video game development and publishing company that was founded in 1996. It's most famous game is the *Half Life* launched in 1998. Over the time period of this study, Valve Software continued to remain a fairly small and niche developer (developing games mostly in the first person shooter genre) focused primarily on PC gaming.

Table 22b demonstrates the growth of Valve Software between November 2000 and November 2008 – the time period of focus in this dissertation. Specifically, the evolution of the number of game titles, the average embeddedness and the porting patterns of Valve Software since the launch of the sixth generation video game consoles in November 2000 is shown. The numbers clearly demonstrate some interesting patterns that contrast with those of Electronic Arts. As Valve Software made the shift from supporting the sixth to seventh generation consoles, there was a significant decrease in its embeddedness and most of its game titles launched for the PC were ported to the Xbox and Xbox360 platforms. Specifically, it went from being 100% embedded in the PC platform to being only 33% embedded by November 2008. Also, note the significant change in the embeddedness and extent of porting before and after the launch of the Xbox platform in 2004 and Xbox360 and XNA architecture in 2005. It was 80% embedded in November but only 56% and 48% embedded in November 2004 and November 2005 respectively. Moreover, the extent of porting in November 2003 was only 11% in 2003 but went up to 28% and 33% in November 2004 and 2005 respectively. Therefore, unlike Electronic Arts whose level of embeddedness and extent of porting remained fairly consistent during this time period, the response to the

architectural shifts and consequent evolution of patterns of support were significantly different for Valve Software. This clearly indicates that it was influenced more by the architectural shifts - the entry of Microsoft with its Xbox platform and also the launch of the XNA architecture – than Electronic Arts.

**Table 22b. Evolution of Patterns of Support for Valve Software**

	Nov-00	Nov-01	Nov-02	Nov-03	Nov-04	Nov-05	Nov-06	Nov-07	Nov-08
Number of titles launched	6	8	8	9	14	18	19	36	38
Embedded-ness	1	0.78	0.78	0.80	0.56	0.49	0.51	0.33	0.34
Proportion of titles ported	0	0.13	0.13	0.11	0.29	0.33	0.32	0.5	0.5

This is consistent with the contention of this dissertation that there is significant heterogeneity in how and why third-party developers respond to architectural shifts. Developers that were less embedded in a single platform were less influenced by exogenous architectural shifts initiated by platform providers. In this case the launch of the XNA platform did not significantly change the extent of cross-platform support and porting by Electronic Art between PC and consoles. A potential explanation for this could be the availability of in-house game engines that it had significant experience using to realign their knowledge, routines and code base as newer platform generations were launched. On the other hand smaller, more embedded developers that were locked-in to a single platform technology and did not have the resources to invest in expensive

middleware were unable to make the transition across radically different platform architectures. However, as platform architectures shifted towards convergence and architectural distance across platforms decreased even smaller, more embedded third-party developers were able to leverage their existing knowledge and code base in new settings. This clearly demonstrates that the role of architectural shifts in moderating the relationship between developer embeddedness and patterns of support was less pronounced for larger, more influential and less embedded developers and more critical for smaller, more embedded developers.

## VIII. CONCLUSIONS

Platform-based settings such as video games, packaged software and mobile telephony are increasingly characterized by highly dynamic and uncertain environments. Platform owners face the increasingly difficult challenge of leveraging their core platform architectures, knowledge assets and complementor networks to create competitive distinctiveness in new platform settings. Third-party developers, on the other hand, face the challenge of having to rapidly adapt their knowledge and routines to competing platforms simultaneously (Venkatraman & Lee, 2004). This challenge is exacerbated by the fact that platform technologies are evolving very rapidly and previously disparate platforms are converging in terms of functionality compelling developers to expand the scope of their support across extant platforms.

As a consequence, architectural strategies are continuously structured by platform providers that allow them to translate their existing standards and routines in new settings and hence, leverage their current ecosystem of complementors. Third-party complementors also are continuously seeking opportunities to re-use and adapt their existing portfolio of applications across disparate platform settings. The impact of the architectural shifts that are driving convergence across platforms on the dynamics of platform-complementor interactions, thus, cannot be underestimated and needs to be examined to better understand the competitive outcomes that accrue to both platform providers and third-party developers who support these platforms.

It is, therefore, a contention of this dissertation that a more holistic examination of the dynamics of platform-complementor interactions during periods of architectural change is needed. A better understanding of the evolution of patterns of linkage between platform owners and complementors during architectural shifts invariably leads to a better understanding of the dynamics of platform-based competition and success outcomes for both parties.

### **8.1. Contributions**

This dissertation has 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. While strategic management research has adopted some of the core ideas of platform-based competition in concepts such as platform leadership (Gawer & Cusumano, 2002; Gawer & Henderson, 2007), additional research on strategic actions and performance drivers under dynamic evolution of platform-based competition is needed that recognizes that competing systems not only exist contemporaneously but also evolve to take advantage of newer generations of technology. The general specification of platform-based competition (i.e., hardware – software coordination) has triggered several strands of research. Some research has focused on the characteristics of platforms and/or those firms that offer them (e.g., Gawer & Cusumano, (2002)) while others have focused on complementors supporting the platforms (Afuah & Bahram, 1995; Afuah, 2000). However, empirical research is characterized by (a) predominance of cross-sectional

studies and (b) studies mostly focused on one actor—either the hardware or the software firm but not on their interactions. This dissertation contended that the research can be informed by a more holistic, dynamic approach that recognizes the co-evolution of the roles of hardware platform and software components. For this purpose, I built on the general research stream of network evolution (Gulati & Gargiulo, 1999; Powell, White, Koput, & Owen-Smith, 2005) as well as on specific attempts to understand the impact of technological change on network evolution (Rosenkopf & Padula, 2008).

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. Specifically, the PC and video game consoles are becoming increasingly convergent and substitutable as gaming platforms. For instance, the launch of cross-industry standards such as the XNA platform by Microsoft has resulted in increased overlap in developers and games between PC and video game consoles. Also since the launches of video game consoles clearly fall into generations based on technology improvements, looking at platform-complementor evolutions across the PC and two distinct video game generations enables us to understand the impact of technology changes on complementor choices and hence, platform-complementor network evolution.

Another important contribution of this dissertation is the adoption of the complementor perspective to understand how complementor attributes influence why and how they choose to link to platforms. In so doing, this dissertation has focused attention

on an important but relatively neglected area of work - a small, albeit important body of studies on platform-based competition has examined the impact of linkage decisions from the perspective of third-party complementors (example, Afuah & Werner, (2007); Venkatraman & Lee, (2004). However, these studies have: 1.) focused on a single setting 2.) focused only on how platform attributes determine the choice of support. By focusing exclusively on platform and platform provider attributes in driving complementor choice, these studies have not attempted to reconcile the fact that there is heterogeneity in complementor attributes and experience and hence their ability to leverage the resources provided by platform sponsors and support multiple contemporaneously existing platforms. This dissertation, thus, extends this body of research by using the firm capabilities and network perspectives as conceptual anchors to understand how developer specific attributes like the relative embeddedness in platforms influences their choice of support.

By embracing the network perspective in platform settings, this dissertation underscores that complementor 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 results of the study also offer further evidence of the value of adopting a network perspective. Specifically, the lack of findings in the relationship between developer resource positions (reflected through their age and size) and choice of support coupled with strong support for the relationship between developer network positions (embeddedness) and patterns of tie formation underscores the importance of adopting a network perspective in platform-based settings.



This research also contributes to the broader stream of network research by expanding the scope and application of network concepts in a novel setting such as the platform-complementor ecosystem. The concept of embeddedness has been adopted extensively in network research (for instance, (Granovetter, 1985; Uzzi, 1996; Gulati, 1999; Venkatraman & Lee, 2004; Powell, White, Koput, & Owen-Smith, 2005)) 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). A key 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 would 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.

Another key contribution of this dissertation 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. By developing a more nuanced conceptualization of the mode of attachment, this dissertation has sought to focus attention on how developer attributes influence the extent to which developers' are likely to support platforms through ported or unique ties.

Additionally, this dissertation has also 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. Within the literature on management and organization, there is general acceptance that organizational actions are embedded in networks of relationships ((Granovetter, 1985; Gulati, 1999) with studies focused on how prior network structures drive formation of new ties. A related stream of research has in turn focused on the role of 'weak ties' on network partner choice. 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 allows us to discern the role of architectural compatibility between platforms and the role of design operators like porting and the availability of middleware on the preferential attachment decisions of complementors.

To this end, the development of the architectural distance measure used in this dissertation is an important first step towards understanding the co-evolution between architectural convergence and platform-complementor network evolution. To the knowledge of this dissertation, 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.

Empirically, this dissertation also makes modest contributions. This study goes beyond describing the macro-structure of the network to modeling network evolution from a decision-making perspective for each developer in the network. By adopting the nested logit model to understand why a developer chooses to link to a platform through unique or ported ties and consequently which platform it supports through ported ties, the model explicates the logic of network dynamics as an aggregate of individual choices (Salancik, 1995; Venkatraman & Lee, 2004). While the value of models of choice (Greve, 1999; Venkatraman & Lee, 2004; Powell, White, Koput, & Jason Owen-Smith, 2005) such as the conditional logit model have been demonstrated in settings where decision makers have to choose from available alternatives, the value of the nested logit model in its ability to model a sequence of decisions has not yet been demonstrated in strategic management research. This dissertation thus contributes methodologically to the strategic management literature by introducing the nested logit model that has been used extensively in transportation and marketing research.

Finally, this dissertation 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 dissertation 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 was validated by the result that as platform firms like Microsoft transferred their architectural standards from the PC platform to the video game sector, they expanded the scope of network overlap between the PC and video game platforms by enabling even their most technologically embedded developers to migrate quickly to the new platform setting.

## **8.2. Limitations and Future Work**

This dissertation is a valuable first step introduced in discerning the dynamics of platform-complementor network evolution during periods of architectural shifts. The results of this dissertation 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, netbooks and e-book readers. However, due to data limitations this dissertation could not expand its scope to include other platform settings. Therefore, a useful extension to this paper could be to test the hypotheses in a new setting such as mobile telephony where platforms like Windows Mobile, Symbian, Apple iPhone and Google Android are competing for third-party applications to run on their smart phones. This would allow us to test the generalizability of the results obtained within the context of the PC and video game sectors. Moreover, this dissertation has only focused on one genre of complements – the third-party games. However, platform such as PCs and mobile support an extensive array of applications beyond games. Examining how third-party developers of different genres respond to architectural shifts could also provide additional insight into platform-complementor network dynamics during periods of convergence.

This 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 dissertation. 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.

Another important limitation of this study is that it focuses exclusively on the evolution of the linkage patterns between platform providers and third-party developers without explicating the impact of these linkage patterns in driving indirect network effects and hence performance. As a result, in addition to studying the ability of game developers to port games across platforms, a possible extension to our study could be to study the impact of porting on both game developer and platform dominance. In addition, it would be interesting to analyze the differential impact of patterns of linkage on platform dominance across larger developers as compared smaller or less significant developers.

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. Finer-grained approaches that elaborate on the metric proposed in this dissertation would go further to link concepts of platform architecture to

strategies under network-based competition. They will also go a long way towards examining how platform providers can specifically influence and induce different complementors to make their games perform at a higher level of performance in comparison to alternative platforms that may be available or forthcoming.

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 dissertation nonetheless contributes to a better understanding of the dynamics of competition in platform-based settings.

## IX. APPENDICES

### 9.1. Appendix A: Results for Analysis 1 with Alternative Weighting Schemes

**Table 23. Results of Sensitivity Analysis for Analysis 1 with Alternative Architecture Distance Weights**

	Model 1 (AD = 90/10, 0 and 1)	Model 2 (AD = 80/20, 0 and 1)	Model 3 (AD = 80/20, 0/0.5/1)	Model 4 (AD = 60/40, 0/0.5/1)
<i>Platform-Specific</i>				
<b>Platform Centrality</b>	23.11*** (2.35)	22.08*** (2.33)	24.86*** (2.47)	27.61*** (2.71)
<b>Architecture Distance</b>	-37.81*** (2.90)	-37.32*** (2.90)	-41.92*** (3.15)	-44.62*** (3.36)
<b>Platform Age</b>	-0.03* (0.01)	-0.03* (0.01)	-0.04*** (0.01)	-0.06*** (0.01)
<b>Platform Porting Centrality</b>	61.07*** (3.94)	61.90*** (4.03)	64.44*** (4.41)	68.56*** (4.84)
<b>Platform Genre Distance</b>	-73.66*** (6.67)	-74.26*** (6.43)	-77.74*** (6.74)	-84.12*** (7.03)
<b>Architecture Distance*Dev Embeddedness</b>	84.88*** (6.72)	89.37*** (7.11)	86.92*** (6.94)	87.56*** (7.36)
<i>Developer-Specific: Ported ties base</i>				
<b>Developer Age</b>	0.0004*** (0.0001)	0.0006*** (0.0001)	0.0003* (0.0001)	0.0002† (0.0001)
<b>Developer Centrality</b>	20.78 (18.84)	19.50 (19.25)	28.41 (19.08)	11.63 (20.71)
<b>Developer Embeddedness</b>	56.65*** (4.26)	61.12*** (4.60)	56.31*** (4.34)	56.92*** (4.66)
<b>Sony</b>	-0.28 (0.50)	-0.11 (0.50)	6.26*** (0.51)	4.77*** (0.52)
<b>Nintendo</b>	2.22*** (0.57)	2.52*** (0.56)	1.32* (0.66)	1.39* (0.69)
<b>Log-likelihood</b>	-1943.48	-1941.92	-1911.44	-1883.48

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



## BIBLIOGRAPHY

Afuah, A., & Werner, R. 2007. *Technological change and the negative effects of embeddedness*. Paper presented at Druid Summer Conference Proceedings, Copenhagen.

Afuah, A. 2000. How much do your "co-opetitors'" capabilities matter in the face of technological change? *Strategic Management Journal*, 21(3, Special Issue: Strategic Networks): 387-404.

Afuah, A. N., & Bahram, N. 1995. The hypercube of innovation. *Research Policy*, 24(1): 51-76.

Ahuja, G. 2000a. Collaboration networks, structural holes, and innovation: A longitudinal study. *Administrative Science Quarterly*, 45(3): 425-455.

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.

Aldrich, H., & Auster, E. R. 1986. Even dwarfs started small: Liabilities of age and size and their strategic implications. *Research in Organizational Behavior*, 8: 165.

Amit, R., & Schoemaker, P. J. H. 1993. Strategic assets and organizational rent. *Strategic Management Journal*, 14(1): 33-46.

Anderson, P., & Tushman, M. L. 1990. Technological discontinuities and dominant designs: A cyclical model of technological change. *Administrative Science Quarterly*, 35(4): 604-633.

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

Bae, J., & Gargiulo, M. 2004. Partner substitutability, alliance network structure, and firm profitability in the telecommunications industry. *Academy of Management Journal*, 47(6): 843-859.

Baker, W. E. 1990. Market networks and corporate behavior. *The American Journal of Sociology*, 96(3): 589-625.

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.

Barnett, W. P. 1990. The organizational ecology of a technological system. *Administrative Science Quarterly*, 35(1, Special Issue: Technology, Organizations, and Innovation): 31-60.

Barney, J. B. 1986. Types of competition and the theory of strategy: Toward an integrative framework. *The Academy of Management Review*, 11(4): 791-800.

Barney, J. B. 1996. The resource-based theory of the firm. *Organization Science*, 7(5): 469.

Barron, D. N., West, E., & Hannan, M. T. 1994. A time to grow and a time to die: Growth and mortality of credit unions in new york city, 1914-1990. *The American Journal of Sociology*, 100(2): 381-421.

Baum, J. A. C., Calabrese, T., & Silverman, B. S. 2000. Don't go it alone: Alliance network composition and startups' performance in Canadian biotechnology. *Strategic Management Journal*, 21(3, Special Issue: Strategic Networks): 267-294.

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

Baum, J. A. C., Rowley, T. J., Shipilov, A. V., & Chuang, Y. 2005. Dancing with strangers: Aspiration performance and the search for underwriting syndicate partners. *Administrative Science Quarterly*, 50(4): 536-575.

Beckman, C. M., & Haunschild, P. R. 2002. Network learning: The effects of partners' heterogeneity of experience on corporate acquisitions. *Administrative Science Quarterly*, 47(1): 92-124.

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.

Ben-Akiva, M., & Lerman, S. R. 1985. *Discrete choice analysis: Theory and application to travel demand*. Cambridge, MA: MIT Press.

Bhat, C. R. 1997. Covariance heterogeneity in nested logit models: Econometric structure and application to intercity travel. *Transportation Research Part B*:

*Methodological*, 31(1): 11-21.

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

Burt, R., & Knez, M. 1995. Kinds of third-party effects on trust. *Rationality and Society*, 7(3): 255-292.

Burt, R. S. 1997. The contingent value of social capital. *Administrative Science Quarterly*, 42(2): 339-365.

Chib, S., Seetharaman, P. B., & Strijnev, A. 2004. Model of brand choice with a no-purchase option calibrated to scanner-panel data. *Journal of Marketing Research (JMR)*, 41(2): 184-196.

Chintagunta, P. K. 1993. Investigating purchase incidence, brand choice and purchase quantity decisions of households. *Marketing Science*, 12(2): 184-208.

Chintagunta, P. K., & Vilcassim, N. J. 1998. Empirical implications of unobserved household heterogeneity for manufacturer and retailer pricing. *Journal of Retailing and Consumer Services*, 5(1): 15-24.

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.

Clayton M. Christensen, Suárez, F. F., & Utterback, J. M. 1998. Strategies for survival in fast-changing industries. *Management Science*, 44(12, Part 2 of 2): S207-S220.

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.

Cohen, D. 1981. On holy wars and a plea for peace. *IEEE Computer Society*, 14(10): 48-54.

Coleman, J. S. 1988. Social capital in the creation of human capital. *The American Journal of Sociology*, 94(Supplement: Organizations and Institutions: Sociological and Economic Approaches to the Analysis of Social Structure): S95-S120.

- Conner, K. R., & Prahalad, C. K. 1996. A resource-based theory of the firm: Knowledge versus opportunism. *Organization Science*, 7(5): 477-501.
- Cool, K., & Schendel, D. 1988. Performance differences among strategic group members. *Strategic Management Journal*, 9(3): 207-223.
- 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.
- de Nooy, W., Mrvar, A., & Batagelj, J. 2005. *Exploratory social network analysis with pajek* (Volume 27 ed.) Cambridge University Press.
- Deeds, D. L., & Hill, C. W. L. 1999. An examination of opportunistic action within research alliances: Evidence from the biotechnology. *Journal of Business Venturing*, 14(2): 141.
- 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.
- Dyer, J. H., & Singh, H. 1998. The relational view: Cooperative strategy and sources of interorganizational competitive advantage. *The Academy of Management Review*, 23(4): 660-679.
- Echols, A., & Tsai, W. 2005. Niche and performance: The moderating role of network embeddedness. *Strategic Management Journal*, 26(3): 219-238.
- Eisenhardt, K. M., & Martin, J. A. 2000. Dynamic capabilities: What are they? *Strategic Management Journal*, 21(10): 1105.
- 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., Hagi, A., & Schmalensee, R. 2008. *Invisible engines how software platforms drive innovation and transform industries*. Boston: MIT Press.

Gal, V., Le Prado, C., Natkin, S., & Vega, L. 2004. *Writing for video games* (Working Paper ed.). CEDRIC.

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.

Garud, R., Jain, S., & Kumaraswamy, A. 2002. Institutional entrepreneurship in the sponsorship of common technological standards: The case of Sun Microsystems and java. *Academy of Management Journal*, 45(1): 196-214.

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

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.

Greve, H. R. 1999. The effect of core change on performance: Inertia and regression toward the mean. *Administrative Science Quarterly*, 44(3): 590-614.

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.

Gulati, R., Nohria, N., & Zaheer, A. 2000. Strategic networks. *Strategic Management Journal*, 21(3, Special Issue: Strategic Networks): 203-215.

Haaijer, R., Kamakura, W., & Wedel, M. 2001. The 'no-choice' alternative in conjoint choice experiments. *International Journal of Market Research*, 43(1): 93-106.

Hakuta, M., & Ohminami, M. 1997. A study of software portability evaluation. *Journal of Systems & Software*, 38(2): 145.

Hannan, M. T., & Freeman, J. 1989. Organizations and social structure. In *Organizational ecology*: 3-27. Cambridge: Harvard U. Press.

Hannan, M. T. 1998. Rethinking age dependence in organizational mortality: Logical formalizations. *The American Journal of Sociology*, 104(1): 126-164.

Hannan, M. T., & Freeman, J. 1984. Structural inertia and organizational change. *American Sociological Review*, 49(2): 149-164.

Hardwidge, B. The problem with porting games. <http://www.bit-tech.net/gaming/2009/08/18/the-problem-with-porting-games/1> January 15, 2010.

Helfat, C., S. Finkelstein, W. Mitchell, M.A. Peteraf, H. Singh, D.J. Teece, & S.G. Winter. 2007. *Dynamic capabilities*. Malden: Blackwell.

Henderson, R. 1993. Underinvestment and incompetence as responses to radical innovation: Evidence from the photolithographic alignment equipment industry. *The Rand Journal of Economics*, 24(2): 248-270.

Henderson, R. M., & Clark, K. B. 1990. Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*, 35(1, Special Issue: Technology, Organizations, and Innovation): 9-30.

Henderson, R., & Cockburn, I. 1994. Measuring competence? exploring firm effects in pharmaceutical research. *Strategic Management Journal*, 15: 63-84.

Henderson, R., & Cockburn, I. 1996. Scale, scope, and spillovers: The determinants of research productivity in drug discovery. *RAND Journal of Economics*, 27(1): 32-59.

Hoang, H., & Rothaermel, F. T. 2005. The effect of general and partner-specific alliance experience on joint R&D project performance. *Academy of Management Journal*, 48(2): 332-345.

Iansiti, M., & Clark, K. B. 1994. Integration and dynamic capability: Evidence from product development in automobiles and mainframe computers. *Industrial & Corporate Change*, 3(3): 557-605.

Kamakura, W. A., Kim, B., & Lee, J. 1996. Modeling preference and structural heterogeneity in consumer choice. *Marketing Science*, 15(2): 152-172.

Kash, D. E., & Rycroft, R. W. 2000. Patterns of innovating complex technologies: A framework for adaptive network strategies. *Research Policy*, 29(7): 819.

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

Khanna, T., Gulati, R., & Nohria, N. 1998. The dynamics of learning alliances: Competition, cooperation, and relative scope. *Strategic Management Journal*, 19(3): 193-210.

Klepper, S. 1996. Entry, exit, growth, and innovation over the product life cycle. *The American Economic Review*, 86(3): 562-583.

Klepper, S., & Simons, K. L. 2000. The making of an oligopoly: Firm survival and technological change in the evolution of the U.S. tire industry. *Journal of Political Economy*, 108(4): 728-760.

Knoke, D. 1990. Networks of political action: Toward theory construction. *Social Forces*, 68(4): 1041-1063.

Kogut, B., Walker, G., & Kim, D. 1995. Cooperation and entry induction as an extension of technological rivalry. *Research Policy*, 24(1): 77-95.

Kohler, C. EA's CEO: How I learned to acquire developers and not \*\*\* them up. <http://www.wired.com/gamelifelife/2008/02/riccitiello/>. January 15, 2010.

Koppelman, F. S., & Chieh-Hua Wen. 1998. Alternative nested logit models: Structure, properties and estimation. *Transportation Research: Part B*, 32B(5): 289.

Lee, D., & Mendelson, H. 2008. Divide and conquer: Competing with free technology under network effects. *Production & Operations Management*, 17(1): 12-28.

Levinthal, D. A. 1991. Organizational adaptation and environmental selection-interrelated processes of change. *Organization Science*, 2(1, Special Issue: Organizational Learning: Papers in Honor of (and by) James G. March): 140-145.

Levinthal, D. A., & March, J. G. 1993. The myopia of learning. *Strategic Management Journal*, 14(, Special Issue: Organizations, Decision Making and Strategy): 95-112.

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 Journal*, 45(6): 1104-1119.

Lippman, S. A., McCardle, K. F., & Rumelt, R. P. 1991. Heterogeneity under competition. *Economic Inquiry*, 29(4): 774.

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

McEvily, B., & Marcus, A. 2005. Embedded ties and the acquisition of competitive capabilities. *Strategic Management Journal*, 26(11): 1033-1055.

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 & Economics*, 2(1): 23-58.



Nelson, R. R., & Winter, S. G. 1982. The schumpeterian tradeoff revisited. *American Economic Review*, 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.

Nohria, N., & Garcia-Pont, C. 1991. Global strategic linkages and industry structure. *Strategic Management Journal*, 12(, Special Issue: Global Strategy): 105-124.

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

Oxley, J. E. 1997. Appropriability hazards and governance in strategic alliances: A transaction cost approach. *Journal of Law, Economics, and Organization*, 13(2): 387-409.

Palmer, D. A., Jennings, P. D., & Zhou, X. 1993. Late adoption of the multidivisional form by large U.S. corporations: Institutional, political, and economic accounts. *Administrative Science Quarterly*, 38(1): 100-131.

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

Penrose, E. G. 1959. *The theory of the growth of the firm*. New York: Wiley.

Peteraf, M. A. 1993. The cornerstones of competitive advantage: A resource-based view. *Strategic Management Journal*, 14(3): 179-191.

Podolny, J. 1993. A role-based ecology of technological change. *American Journal of Sociology*, 100(5): 1224-1260.

Podolny, J. M. 1994. Market uncertainty and the social character of economic exchange. *Administrative Science Quarterly*, 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.

Podolny, J. M., & Page, K. L. 1998. Network forms of organization. *Annual Review of Sociology*, 24: 57-76.

Podolny, J. M., Stuart, T.E. & Hannan, M. T. 1996. Networks, knowledge, and niches: Competition in the worldwide semiconductor industry, 1984-1991. *The American Journal of Sociology*, 102(3): 659-689.

Poh-Lin Yeoh, & Roth, K. 1999. An empirical analysis of sustained advantage in the U.S. pharmaceutical industry: Impact of firm.. *Strategic Management Journal*, 20(7): 637-653.

Powell, W. W., Koput, K. W., & Smith-Doerr, L. 1996. Interorganizational collaboration and the locus of innovation: Networks of learning in biotechnology. *Administrative Science Quarterly*, 41(1): 116-145.

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.

Prahalad, C. K., & Hamel, G. 1990. The core competence of the corporation. *Harvard Business Review*, 68(3): 79-91.

Ranger-Moore, J. 1997. Bigger may be better, but is older wiser? Organizational age and size in the New York life insurance industry. (Cover story). *American Sociological Review*, 62(6): 903-920.

Reimer, J. Cross-platform game development and the next generation of consoles. <http://arstechnica.com/old/content/2005/11/crossplatform.ars/> January 15, 2010.

Rosenkopf, L., & Padula, G. 2008. Investigating the microstructure of network evolution: Alliance formation in the mobile communications industry. *Organization Science*, 19(5): 669-687.

Rowley, T., Behrens, D., & Krackhardt, D. 2000. Redundant governance structures: An analysis of structural and relational embeddedness in the. *Strategic Management Journal*, 21(3): 369.

Rumelt, R. P. 1991. How much does industry matter? *Strategic Management Journal*, 12(3): 167-185.

Salancik, G. R. 1995. Review: WANTED: A good network theory of organization; structural holes: The social structure of competition. *Administrative Science Quarterly*, 40(2): 345-349.

Schumpeter, J. A. 1942. *Capitalism, socialism and democracy*. New York: Harper.

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

Shimpi, A. L. Hardware behind the consoles - part I: Microsoft's xbox. <http://www.anandtech.com/show/853/2> January 15, 2010.

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.

Smith-Doerr, L., Manev, I. M., & Rizova, P. 2004. The meaning of success: Network position and the social construction of project outcomes in an R&D lab. *Journal of Engineering & Technology Management*, 21(1): 51.

Sørensen, J. B., & Toby E. Stuart. 2000. Aging, obsolescence, and organizational innovation. *Administrative Science Quarterly*, 45(1): 81-112.

Sriram, S., Chintagunta, P. K., & Neelamegham, R. 2006. Effects of brand preference, product attributes, and marketing mix variables in technology product markets. *Marketing Science*, 25(5): 440-456.

Starbuck, W. H. 1983. Organizations as action generators. *American Sociological Review*, 48(1): 91-102.

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.

Stokes, J. The PlayStation2 vs. the PC: A system level comparison of two 3D platforms. <http://arstechnica.com/old/content/2000/04/ps2vspc.ars> January 25, 2010.

Stuart, T.E. 2000. 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.

Sun, B., Neslin, S. A., & Srinivasan, K. 2003. Measuring the impact of promotions on brand switching when consumers are forward looking. *Journal of Marketing Research*, 40(4): 389-405.

Tai-Young Kim, Oh, H., & Swaminathan, A. 2006. Framing interorganizational network change: A network inertia perspective. *Academy of Management Review*, 31(3): 704-720.

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 Journal*, 51(2): 381-397.

Teece, D. J. 2007. Explicating dynamic capabilities: The nature and microfoundations of (sustainable) enterprise performance. *Strategic Management Journal*, 28(13): 1319-1350.

Teece, D. J., Pisano, G., & Shuen, A. 1997. Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7): 509-533.

Train, K. E. 2003. *Discrete choice methods with simulation*. Cambridge University Press.

Train, K. 1980. A structured logit model of auto ownership and mode choice. *The Review of Economic Studies*, 47(2): 357-370.

Tushman, M. L., & Anderson, P. 1986. Technological discontinuities and organizational environments. *Administrative Science Quarterly*, 31(3): 439-465.

Tushman, M. L., & Romanelli, E. 1985. Organizational evolution: A metamorphosis model of convergence and reorientation. *Research in Organizational Behavior*, 7: 171.

Utterback, J. M., & Suárez, F. F. 1993. Innovation, competition, and industry structure. *Research Policy*, 22(1): 1-21.

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.

Uzzi, B., & Gillespie, J. J. 2002. Knowledge spillover in corporate financing networks: Embeddedness and the firm's debt performance. *Strategic Management Journal*, 23(7): 595.

Ven, A. H. V. d. 1986. Central problems in the management of innovation. *Management Science*, 32(5, Organization Design): 590-607.

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.

Wernerfelt, B. 1984. A resource-based view of the firm. *Strategic Management Journal*, 5(2): 171-180.

Wernerfelt, B., & Montgomery, C. A. 1988. Tobin's q and the importance of focus in firm performance. *American Economic Review*, 78(1): 246.

West, J., & Dedrick, J. 2000. Innovation and control in standards architectures: The rise and fall of Japan's PC-98. *Information Systems Research*, 11(2): 197.

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

Westphal, J. D., Gulati, R., & Shortell, S. M. 1997. Customization or conformity? an institutional and network perspective on the content and consequences of TQM adoption. *Administrative Science Quarterly*, 42(2): 366-394.

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.

Zott, C. 2003. Dynamic capabilities and the emergence of intraindustry differential firm performance: Insights from a simulation study. *Strategic Management Journal*, 24(2): 97-125.

## CURRICULUM VITAE

### Arati Srinivasan

Strategy and Innovation Department  
Boston University – School of Management  
595 Commonwealth Avenue  
Boston, MA 02215  
arati@bu.edu

30 CambridgePark Drive, Apt. 5132  
Cambridge, MA 02140  
(404)-808-4734

#### RESEARCH AGENDA

Research interests include a network centric view of business strategy in technology settings such as the video game and mobile sectors. I am specifically interested in adopting a network lens to understand how platform-complementor interactions evolve dynamically over time. My research further emphasizes the use of network analysis tools and measures to understand how collaborations with third party developers result in platform dominance. Other research interests also include technology evolution, standards and dominant designs.

#### TEACHING INTERESTS

Courses in Strategy and Innovation Management  
Electives in Technology and Innovation Management, IT Strategies in Networked Markets

#### EDUCATION

**Boston University**, School of Management, Boston MA  
*DBA, Strategy and Innovation*, September 2010

**Georgia Institute of Technology**, College of Management, Atlanta GA  
*MBA, Information Systems*, June 2004

**Georgia Institute of Technology**, School of Textile Engineering, Atlanta GA  
*MS, Textile Engineering*, June 2001

**Anna University**, Chennai, India  
*BS, Textile Engineering*, May 1999

#### DISSERTATION

**Title:** The Dynamics of Platform-based Competition during Periods of Architectural Shifts

**Dissertation Committee:** N. Venkatraman (Chair), Fernando Suarez, George Wyner, Shuba Srinivasan

**Abstract:** Success in platform-based settings calls for mutual dependency between platform providers and third-party complementors. Consequently, complementors need to continually refine their routines to align with the shifting architectures of platforms. However, prior research on platform-based competition has focused mostly on the actions of platform providers

with minimal work examining the perspective of complementors. Moreover, most extant research has adopted a static or cross-sectional perspective that does not recognize the role of architectural shifts in driving platform-complementor network dynamics. Yet there is considerable heterogeneity in the resource positions of third-party complementors and hence in when, how and why they respond to architectural shifts and support certain platforms.

This dissertation is, therefore, a first step towards recognizing the time-varying changes in platform-based competition during periods of architectural shifts with a focus on complementors who support and shape the characteristics of network structure between platform providers and complementors. This study also brings to the forefront the differences in the type of support provided by complementors, namely: ported ties vs. unique ties and that this type of support influences the dynamics of platform-based competition. This dissertation, thus, specifically seeks to address the following questions: How do third-party complementor firm specific attributes (age and dominance) influence their likelihood of support to platform through ported ties? Do complementor network positions (embeddedness in platform-complementor networks) further influence their likelihood of support to platforms? Finally, how do platform characteristics (platform dominance and platform relatedness) coupled with complementor attributes influence the developers' choice of platform to support? To answer these and related questions, I study the platform-complementor network in the PC and video game industry as the sixth generation technology of video game consoles transitions to the seventh generation from January 2000-December 2008. This setting allows us to understand the role of architectural shifts that have driven convergence between PC and video game platforms such as the introduction of cross-boundary standards (such as DirectX Graphics API) and middleware (for example, XNA Games Studio) in influencing related shifts in platform-based competition. The expected contributions of this dissertation are to offer insights into the dynamics of platform-based competition by focusing on complementor attributes and by distinguishing between the types of support provided by complementors to platforms. Further, by adopting a longitudinal perspective and studying the dynamics of platform-based competition during periods of architectural shifts, this study provides insight into time-varying changes in platform-complementor interactions.

## **JOURNAL PAPERS**

2010, "Indirect Network Effects and Platform Dominance in the Video Game Industry: A Network Perspective," *IEEE Transactions on Engineering Management*, Forthcoming (with N. Venkatraman)

2009, "Investigating the Dynamics of System-Based Competition in the Video Game Sector," *Academy of Management Best Paper Proceedings* (with N. Venkatraman)

**CONFERENCE  
PRESENTATIONS**

2010, ““First Mover Advantages in Hyper-Dynamic Environments: A Study of the iPhone Platform and its Ecosystem of Complementors,” Academy of Management, Montreal, Forthcoming (with Fernando Suarez)

2009, “Investigating the Dynamics of System-Based Competition in the Video Game Sector,” Academy of Management, Chicago (with N. Venkatraman)

2008, “The Role of Indirect Effects in Explaining Platform Dominance in the Video Game Industry (2002-2006): A Network Perspective,” International Conference on Information Systems, Paris (with N. Venkatraman)

**CASE STUDIES**

2009, “Symbian, Google and Apple in the Mobile Space,” Harvard Business School Case (with Fernando Suarez and Benjamin Edelman)

**IN PROGRESS**

2010, “The Role of Environmental Dynamics in Driving First Mover Advantages in the iPhone Ecosystem” (with Fernando Suarez)

**TEACHING  
EXPERIENCE**

Teaching Assistant, Information and Technology Strategies in a Networked Economy (IS 710), with Prof. Fernando Suarez, Spring 2008. Responsibilities included teaching a couple of sessions on Information Technology Networks.

Teaching Assistant, Information Systems Summer Intensives, with Prof. George Wyner, Summer 2006, Summer 2007.

**PROFESSIONAL  
EXPERIENCE**

**Research Assistant, CATEA, Atlanta, GA 2003-2004**

- Developed a Training Database to enable information gathering across all trainings conducted by CATEA projects.
- Analyzed and documented the adoption trends of the new database and the factors that resulted in resistance to adoption of new technologies.
- Helped develop measures to determine the outcomes of the training conducted by the Institute.
- Assisted in the usability study of the AssistiveTech.net website – a web resource for people with disabilities.

**Software Developer, Vedas Systems Solutions, Atlanta, GA 2001-2002**

- Led the efforts of a team of five to implement a companywide intranet to support the internal business processes.
- Worked closely with client team to develop a set of requirements to develop a “Shipping” application to facilitate the package shipping process by completely automating it.
- Worked with three developers in India to upgrade a software package to facilitate the acquisition of titration data in real-time from the titration instrument.



<b>SKILL SET</b>	Multivariate (regression, factor analysis), cross-sectional and time series models Statistical Packages (SPSS, STATA) Social Network Analysis (Pajek, UCINET, SONIA, Visone) Computer Programming (VBScript, Perl)
<b>DISTINCTIONS AND AWARDS</b>	Finalist, Best Paper Award in Technology & Innovation Division, Academy of Management 2009 Member, Beta Gamma Sigma Society Boston University Doctoral Fellowship 2004-Present Merit-based GRA, Georgia Institute of Technology 1999-2001, 2002-2004 Qureshi Award for Academic Excellence, Anna University 1997-1999
<b>WORK STATUS</b>	U.S. Citizen