The Impact of Design Moves on Platform Adoption: The Case of Microsoft Windows OS

Xiang Liu Chi-Hyon Lee Bala Iyer
School of Management, Boston University
{xiangl,chi-hyon,bala}@bu.edu

Abstract

How soon will firms form alliances with a platform provider? What factors impact the decision and the speed with which firms first move into partnerships with a platform provider such as Microsoft? This study uses the theories of design rules and network effects as its theoretical anchors to explore the timing of the alliance decision with Microsoft. We develop a model that incorporates platform design changes and its impact on the decision of firms to form an alliance with Microsoft. In particular, we analyze two design moves that a platform provider can make – augmenting and inversion. We empirically test the impact of each design move using a sample spanning the years 1989 to 1994. Study results provide an initial design theoretic based examination of platform design changes and its impact on alliance formation.

1. Introduction

There exists a significant body of research on organizational technology adoption [19, 20, 27, 28, 30]. Platform adoption – one branch of this stream of research – is a focus of current research [11, 13]. A platform is defined as "a bundle of standard components around which buyers and sellers coordinate efforts" [6]. In this paper we focus on software platforms. For example, the Microsoft Windows operating system (OS) is a software platform because it can create value to customers by attracting complementors or software companies that design and deliver software products in line with the platform architecture [15].

In addition to making a technical decision to launch their products on the Windows OS, many software firms also form alliances with Microsoft. Given that independent software vendors (ISVs) can develop products for Microsoft by adopting its Windows platform, one interesting question is – why do ISVs form alliances with Microsoft? This may be because an alliance represents a level of commitment over and above adoption of a platform [42]. Alliance partners exchange knowledge, private information, and other strategic resources. An ISV's alliance with a platform manufacturer requires that the ISV realign business resources, i.e., innovate with the focal partner (Microsoft). We will expand on this point in the following section.

A significant body of research explores factors motivating firms to enter into strategic alliances (e.g., [4, 7, 16, 25, 33]). However, there have been relatively few empirical studies undertaken to unearth what factors impact the probability of or the speed at which firms form alliances. One notable exception is Stuart's paper, in which the author examined the effect of a firm's technological network position in the market on its propensity to enter into a strategic alliance [39]. Rather than focusing on a positional explanation of the alliances, we analyze this ubiquitous inter-organizational phenomenon from the ISV's perspective. We focus on the US prepackaged software industry (SIC 7372) – and within it, alliances formed amongst all of the prepackaged software firms and Microsoft concentrating on the Windows OS. We examine how different OS platform design strategies adopted by a platform provider impact the probability of ISVs forming alliances with the platform manufacturer.

One contribution of this paper is that we focus on a relatively young but highly distinctive industry – software – as our research context. The software industry is a classic example of system-based competition – no single company has all the resources to deliver what the end customers seek in terms of value [34]. The software industry is characterized by two notable features: software platforms and network effects. Parties such as software developers, hardware developers, and users are considered to be part of the platform's "ecosystem", for which the platform must

deliver significant value to sustain its own growth. In order to keep the ecosystem "healthy and sustainable", it is important, as well as necessary, for the ecosystem to be openly connected across different members and be actively managed so that the platform is embraced and accepted [14].

The second interesting feature of this sector, actually highly correlated with the "software platform", is "network effects", - a concept rooted in the economics literature. Network effects has been defined as a change in the benefit, or surplus, that an agent derives from a good when the number of other agents consuming the same kind of good changes [23]. The essence of network effects in the software industry is felt because a software platform like Windows is able to accumulate the value by drawing other companies within the ecosystem to interact with itself on its core architecture. In effect, these two interesting features – software platforms and network effects – make our study on alliances new and distinctive.

One of the objectives of this article is to introduce new concepts and constructs concerning modularity and design moves in order to build a new conceptual model of platform adoption. This is one of the most important differentiators of our study from previous ones. More specifically, we do not focus on those attribute-based explanations of the speed of formation of alliances with Microsoft – i.e., endogenous variables of individual firms which have been examined in previous studies. Instead, we examine exogenous variables or design moves enacted between different versions of Windows OS. We call them exogenous variables because they are not a characteristic of ISVs, but rather used to describe dynamic changes of technological components in each version of the Windows platform enacted by the platform manufacturer. The theory of design rules proposed by Baldwin and Clark has developed six design operators which can generate all possible evolutionary paths for the system structure when applied at various points and in different combinations [3]. By applying this theory, we operationalize the various design moves made by Microsoft from one Windows OS version to another. Furthermore, we explore which design operator used by Microsoft, to evolve the platform, influences an ISV's decisions to form an alliance with Microsoft. We believe that this is the first study that employs design operators as major factors in explaining alliances formation.

In the remainder of the paper, we develop a conceptual model of factors affecting how soon software companies form alliances with Microsoft for products that run on the Windows OS and empirically test the model on data gathered from a six-year period – 1989 to 1994. Later years were omitted because by that the Windows OS has become the dominant software platform. More details are given in the data section.

2. Research Model and Hypotheses

2.1. Network effects and ISVs' alliances with Microsoft

Much of the research exploring factors that drive platform adoption have focused on network effects [8, 10, 19, 21, 22, 35]. In the software industry, network effects means that the value of software product is a function of its adoption by customers and complementors [1, 6, 34]. According to economics theory on networks effects, ISVs will develop software products for Windows OS faster as the platform's network effects increase. Larger network effects is associated with platform dominance, all other aspects of the environment for decision making being equal [20].

Because alliances represent higher levels of commitment over and above technology adoptions [42], the same logic suggests that network effects are also an important factor driving the speed with which an ISV forms alliances with Microsoft. When no one firm has the technological resources to deliver products for the end customers by itself, firms access complementary assets through contractual means such as alliances [2]. Research on network effects suggests that the value of a technology may not only rise with the number of other users employing the same technology [19], but is also closely linked to the availability of complementary goods [8, 12]. For example, in 1993 Microsoft and SAP formed an alliance to jointly develop, manufacture, and market applications and products for the client/server environment. Through the alliance, SAP ported its client-server-based R/3 System of applications software to the Windows NT OS. The alliance was designed to provide customers with an efficient exchange of data and higher quality information management between enterprise systems and end-user tools. On the one hand, through porting R/3 System application software to the Windows NT OS, SAP provides a complementary product to the Windows OS, which is a complementor network effect; on the other hand, SAP products could be sold to customers who already adopted Windows OS, thus the consumers benefit from increased network size, which is a direct network effect.

In summary, an ISV, forming alliances with platform providers sooner rather than later, may accrue increasing returns to adoption that allow its products to be improved faster than competitors' products. Pressures for compatibility may further induce more

ISVs to form alliances with Microsoft around a single technology platform – the Windows OS. ISVs sponsoring technologies that are incompatible with Microsoft OS may be locked out of the market. We thus hypothesize that:

H1: Ceteris paribus, an increase in the underlying platform's network effects will decrease the time for an ISV to form an alliance with the platform provider.

Are network effects the only major factor to impact the probability and the speed with which an ISV forms alliances with Microsoft? Do design moves of a platform provider influence the speed of alliance formation beyond that explained by network effects? If yes, then what design strategies used by a platform provider matter? We will explore these questions using design rules theory. Two other hypotheses will be generated to build the conceptual model.

2.2 Design moves of the Windows platform and ISVs' alliances with Microsoft

To reiterate, one of the important distinctions between this study and prior studies on alliances is that we include the design moves a platform provider applies to the platform as factors that predict the speed with which an ISV forms alliances with the platform provider. In this section, we will draw on modular systems theory and design rules theory to build our case.

In *The Sciences of the Artificial*, Simon posits the concept of "nearly decomposable systems", in which the interactions among subsystems could be negligible compared to those interactions within the individual subsystems [36]. The major point of this theory is that a hierarchically nested system consists of components, which can be decomposed into finer components until the components are "elementary particles". Based on Simon's work [36], Baldwin and Clark [3] define modularity as "a particular design structure, in which parameters and tasks are interdependent within units (modules) and independent across them." Therefore, modularity is a systems concept based on relationships among structures, which is a continuum describing the degree to which a system's components can be separated and recombined [32]. According to Baldwin and Clark, if the structure has the form of a nested hierarchy, i.e. is built on units that are highly interconnected in themselves, but largely independent of other units, then the system is modular. Since this concept applies to any entity, one can also apply it to the Windows OS.

An increase in modularity exponentially increases the number of possible configurations achievable from a given set of inputs, greatly increasing the design flexibility of a system [32]. Therefore, modular design structures are expected to allow greater flexibility and rapid innovations. Operators, defined as "actions that change existing structures into new structures in well-defined ways." (p. 129), are used to describe the changes that can be performed on a modular system. The six modular operators are splitting, substituting, augmenting, excluding, inverting and porting. Baldwin and Clark [3] state that these operators, when applied at various points and in different combinations, can generate all possible evolutionary paths for the structure of a system. In this study, we focus on the technological changes between one Microsoft Windows OS version and another and employ the modular operators to analyze each move made to the Windows platform.

As indicated earlier, we are interested in the question of which modular operators will influence the time taken by ISVs to form alliances with Microsoft on the Windows OS. As far as the ISVs are concerned, during the process of execution of those six design operators in the market place, only two of them have an impact on the speed with which software vendors form alliances with Microsoft.

Among the six operators, four of them are excluded in this study: splitting, excluding, porting, and substituting. They are excluded from this study because the four operators, from the ISV's perspective of the OS, are either opaque or rare. **Splitting** takes a single-level design with interdependent parameters and converts it into a hierarchical design with a core set of independent modules… If a module is split, only its internal design structure will change" (p. 133). In the case of an OS platform vendor, splitting may be performed to support internal development effort but may not be visible to an ISV. The same logic can be applied to the operator **substituting**. This operator uses an enhanced module to replace the old one. In other words, although the OS modules may be substituted, each successive generation provides backward compatibility and is opaque with most of the major software applications developed for previous generations. Backward compatibility here refers to new versions of Windows OS that can successfully offer functionalities from earlier versions. Thus customers can upgrade without having to replace their entire libraries of software applications [31]. **Excluding** "denotes leaving a module or functionality out" (p.135, [3]) of the OS. It is rare for Windows OS to include excluding design moves because OS vendors provide backward compatibility. If there are frequent

exclusions in each Windows OS releases, then the rule of backward compatibility will be broken. "**Porting** occurs when a hidden module 'breaks loose' and is able to function in more than one system, under different sets of design rules" (p. 140, [3]). In this paper we specifically focus on one OS: Windows OS. Therefore, the design move porting is beyond the scope of what we are discussing about and not relevant to our study. Based on the reasons provided thus far, in this study, only two operators – augmenting and inversion – out of six are included in our conceptual model.

"**Augmenting** means adding a module." (p. 135, [3]) By augmenting, the user [Microsoft] adds a module to give the system some new type of functionality (p. 136, [3]). In our case, augmenting means that the Windows platform adds new functionality. Since each new version of Windows OS always adds new functionalities compared to the previous version, the operator augmenting is always used in each version release. From the perspective of an ISV, in order to adapt to the technological changes occurring to the underlying platform and reduce uncertainties, they need to realign their internal resources to utilize those new modules within the platform. The more new functionalities are added to the platform, the more time ISVs have to take to determine how to interact with each new module within the platform. In other words, OS complexity increases whenever a new module is augmented to the platform and thus more time is needed by the ISV to realign its internal resources. We thus hypothesize that:

H2: Ceteris paribus, an increase in the number of augmentations to the underlying platform will increase the time for an ISV to form an alliance with the platform provider.

"The operator **inversion** describes the action of taking previously hidden information and 'moving it up' the design hierarchy so that it is visible to a group of modules" (p.138, [3]). An inversion design move makes the inverted-module, previously invisible, visible to one or more modules. In this study inverting means that some module of the Windows OS is made visible to a group of other modules. In an earlier version, the inverted module would have been integrated and opaque to the ISV. For instance, Windows OS simplifies the developer's task by providing an intermediate "abstraction" layer called APIs [24]. Madnick argued that APIs (Application Program Interfaces), which are interfaces "exposed" (made available for applications to use) and "documented" (so that developers know how to use them), give software developers more control and flexibility to manage or develop their applications. More specifically, when a firm obtains complementary assets through contractual means, it shares knowledge about the component interface. This exchange is easier for ISVs if the interface specifications between components and complementary products are codified [29]. Codified knowledge is easily transmitted through designs and specifications [26, 40, 41]. The desire to reduce the costs with its partners through exchange of codified knowledge provides an incentive for the software firms to pursue standards through alliances. Based on the above analysis, we hypothesize that:

H3: Ceteris paribus, an increase in the number of inversions to the underlying platform will decrease the time for an ISV to form an alliance with the platform provider.

The conceptual model is shown as follows.

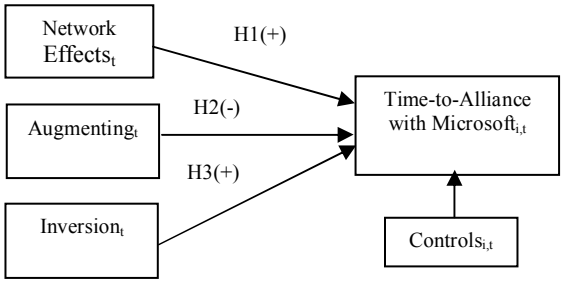

Figure 1. The conceptual model

3. Methods

3.1. Research settings

We use the packaged software industry as our research context. The software industry is prototypical of new modes of competition, where vertical competition or co-opetition is prevalent [5].

3.2. Data

Our analysis is based on the data collected by SDC (Securities Data Corporation). This database contains all alliances that software companies formed with Microsoft across the years 1989-2003. For the purpose of this study, we only focus on the alliances concerning Windows OS. An ISV can form an alliance with Microsoft more than once on different versions of Windows OS at different times. For each software company, we only look at the first alliance it formed with Microsoft and the version of OS it is based on at that time. The Stata program automatically drops data

after year 1994 because there is no significant difference for each predictors of heterogeneity in hazard. It is supported by the fact that at that time Microsoft has already become the dominant software platform leader. Therefore, the predictors of ISVs' timing to form alliances with Microsoft, which we discussed in the conceptual model, will be irrelevant. Based on the data analysis, the resulting database includes a subset of the database that only includes each software firm's "first time alliance" observations covering the period 1989-1994. All observations are aggregated yearly.

3.3. Construct Operationalization

Dependent variable.

Alliance with Microsoft$_{i,t}$. We operationalize the dependent variable as a binary variable. We code Alliance $_{i,t}$ = 1 if ISV i formed an alliance with Micrsoft for the first time in year t and Alliance $_{i,t}$ = 0 otherwise.

Independent Variables.

Network effects$_t$. We use Microsoft's annual sales on 16-bit, 32-bit, and 64-bit Windows OSs as a proxy measure of network effects. We obtained Microsoft's annual Windows OS revenues from International Data Corporation's proprietary database. We thank Dan Vesset and Henry Morris for access to the data. Revenues are denoted in millions and deflated in constant 1994 dollars.

Augmenting$_t$ and Inverting$_t$. We collected descriptions of Windows OS desktop products from mainly three sources: Microsoft, Lexis-Nexis, and wikipedia websites. Then for each Windows OS version in year t, we coded and totaled the number of augments and inversions to the Windows OS. To ensure the reliability of our coding results, two persons coded the data independently. After finishing the coding process, the results were compared by thorough discussions between two coders until an agreement was reached. The full coding scheme and results can be found in Appendices A and B.

Control Variables.

We include a set of variables to control for possible confounding effects.

Absorptive capacity$_{i,t}$. We include absorptive capacity$_{i,t}$ as the ratio of R&D expenses of software vendor i in year t divided by its sales in the same year. Based on Cohen and Levinthal [9], a firm's absorptive capacity is critical to its innovative capabilities. Since within the software ecosystem ISVs choose to develop new complementary products for Windows OS through alliances, and a firm's absorptive capacity positively influence a firm's innovative capabilities; then there is an association between an organization's absorptive capacity and the likelihood for them to form alliances with Microsoft. We obtained R&D expenses from the ComputStat database.

Age$_{i,t}$ and **Size$_{i,t}$.** The ISV age and size are included as control variables. Many studies have examined the impact of firm age and size on innovative behavior and alliance formation. In Sorensen and Stuarts' article [38], based on the theory of "liability of newness", they proposed that organizational age will be positively associated with the rate of innovation. Also in their study, they use total number of employees in biotechnology as time-varying measure of firm size. We obtained ISV age and size from the CompuStat database. These years were log transformed to remove skewness.

Year$_{i,t}$. Finally, we include a set of time indicators. The set of dummy variables takes on values that identify the particular time period of an observation and are used in the statistical estimations.

We summarize the constructs and the operationalizations in the following table.

Table 1. Constructs and operationalizations

Construct		Operationalization	Predicted Signs
Network Effects (NE)		Total Windows OS products sales	+
Augmenting (AUG)		The number of augmenting moves within one version of Windows OS.	-
Inverting (Invert)		The number of inverting moves within one version of Windows OS	+
Control Variables	Absorp -tive Capacity(AC)	Firm's R&D intensity (=R&D expenditure /Sales)	N/P
	Age	Years since first incorporation	N/P
	Size	Total number of employees	N/P
	Year	Dummy variables for year 1989 to 1994	N/P

N/P: No Prediction

3.4. Descriptive statistics

Table 2 is a summary of the descriptive statistics of the ISVs in the sample.

Table 2 Descriptive statistics of the sample and matrix of zero-order correlations

	Age	Size	AC	NE	Aug	Invert
Age	1.000					
Size	0.143	1.000				
AC	0.123	-0.153	1.000			
NE	0.063	-0.122	0.272	1.000		
Aug	-0.049	-0.148	0.100	0.177	1.000	
Invert	-0.106	-0.078	0.138	0.411	0.213	1.000
Mean	2.520	0.877	-1.656	9.649	0.226	0.037
S.D.	1.025	0.631	0.933	0.958	0.444	0.189

There are altogether 184 observations in our sample, which represents the number of unique ISVs times the number of years in each firm's life time.

3.5 Model Specification and Testing

We tested our hypotheses using a discrete time-to-event model of the following form:

$$\ln\left(\frac{h_{i,t}}{1-h_{i,t}}\right) = \tau_{i,t} + \alpha NE_t + \beta Augment_t + \gamma Invert_t + \lambda_{i,t}\varphi$$

where, $h_{i,t}$ is the hazard probability that ISV i will form an alliance with Microsoft on the Windows OS in year t. $\tau_{i,t}$ are year indicator variables, $Augment_t$ and $Invert_t$ represent the number of augmentations and inversions within each version of Windows OS release in year t, and NE_t represents Windows OS annual sales in year t. The vector $\lambda_{i,t}$ contains for firm i the control covariates and coefficient vector φ: firm absorptive capacity, firm age and firm size.

In the discrete time-to-event model, a positive coefficient associated with the predictor suggests an increase in the probability of the event i.e., alliance occurring. An increase in the probability thus suggests a decrease in the time associated with the event [37]. Similarly, a negative coefficient suggests a decrease in the probability of the event occurring thus implying an increase in the time associated with the event.

We estimated the models using Stata 8.2 and utilized robust estimates of variance [17, 43, 44] because a company's covariates are repeatedly measured for every year until the firm's alliance with Microsoft or loss due to end of study and censoring.

4. Results of Hypotheses Testing

Table 3 is a summary of the Stata estimation of the model. We estimated three models by sequentially adding sets of variables. Model 1 contains only time predictors. Model 2 builds on Model 1 but includes control variable such as firm age and firm size. Model 3 builds on Model 2 by including the main effect parameters such as augmenting, inverting, and network effects. All models are individually significant and the addition of covariates across the models significantly increases the overall fit of the hazard model. Based on the known asymptotic distributional properties of –2 times the log-likelihood statistic (-2LL), we can compare the goodness of fit of the original and extended model. For instance, from the row labeled –2LL in Table 3, we see that the model chi-square goodness-of-fit statistic has decreased from 181.778 to 154.108 on the addition of the main effects — network effects, augmentation, and inversion — a difference of 27.67 for the loss of 3 degree of freedom ($p<0.05$).

Table 3. Estimation of models of alliances

Predictor	Model 1		Model 2		Model 3	
	Coef.	Robust S.E	Coef.	Robust S.E	Coef.	Robust S.E
1989	-3.199*	0.728	-3.837*	0.785	-4.686*	0.714
1990	-1.126*	0.335	-1.736*	0.440	-2.357*	0.533
1991	-0.105	0.328	-0.717	0.426	-1.113*	0.545
1992	-0.619	0.473	-1.134*	0.550	- 1.746*	0.691
1993	-1.204	0.665	-1.801*	0.728	-2.156*	0.899
1994	-0.405	0.652	-1.020	0.699	-1.603*	0.952
Age			0.046	0.025	0.031	0.040
Size			0.186*	0.078	0.216*	0.085
AC			0.357*	0.131	0.053	0.197
NE					4.18^E-5*	0.000
Aug					-0.044	0.208
Invert					1.474*	0.671
LL	-94.941		-91.889		-77.054	
-2LL	189.882		181.778		154.108	
Change in -2LL(df)			8.104(3)		27.67(3)	
p			0.044		0.000	

*<0.05, Coef: coefficient, SE: standard error

H1 hypothesizes a negative relationship between network effects and alliance formation time with Microsoft. The coefficient is positive and significant (α = 4.18^E-5, $p<0.001$). As Windows OS network effects increases, the likelihood of alliance formation increases and thus time to formation decreases as hypothesized. H1 is supported.

H2 hypothesizes a positive relationship between the number of augmentation and time to alliance formation. The sign of the parameter is negative as predicted but

not significant. (β =-0.044, p>0.10). H2 is not supported. One possible explanation for the lack of significance can be that platform providers preferentially offer training programs to some ISVs to help them utilize new OS functionalities. However, not all ISVs may participate in the programs. Those ISVs that participate will be able to manage the additional complexity. Additional research is warranted.

H3 hypothesizes a negative relationship between the number of inversions and time to alliance formation. The coefficient for the number of Windows OS inversions is positive and significant (γ =1.474, p<0.05). A positive coefficient denotes an increase in the likelihood of an alliance thus denoting a decrease in the time to alliance formation. H3 is supported.

5. Discussion

In this study we analyze the design moves applied by a platform provider to a platform to understand the timing of ISV alliances with the platform vendor. In particular, we gathered data on firms forming alliances with Microsoft for the Windows operating systems from 1989 to 1994. Rather than focusing on the motivations for alliances, we are interested in what factors will impact the time for ISVs to form alliance with Microsoft. In other words, the main research question is what factors influence the rate of platform adoption in the prepackaged software industry.

Study findings confirm that platform network effects positively impact ISV alliance formation with the platform. Larger network effects will attract more complementors (ISVs) to form alliance with the platform manufacturer and decrease the time for this decision. Although this result is not new, it nonetheless adds to the growing body of findings.

More importantly, a key finding of this study is that the design moves to the platform enacted by a platform manufacturer (provider) play a critical role in the timing of an ISV's decision to form an alliance with the platform manufacturer. Platform design strategies matter.

Unlike previous studies on the platform adoption, we take a perspective that a platform is not static but dynamic. The platform manufacturer could change the internal architecture of the platform to influence complementors' decisions whether to form alliances with the platform vendor or not. The empirical result of this study, to some extent, suggests that in order to speed up adoption, the platform manufacturer should utilize "inversion" design operators more by making more modules visible to ISVs.

Although further research is warranted, inverting OS modules can increase the "openness" of the OS. Linux is an example of an open OS. As such, Linux vendors, such as Redhat and Suse, may enjoy some of the positive alliance formation effects exhibited by Windows OS and Microsoft.

6. Directions for future research

A limitation of this study is that we only focus on one platform: Microsoft Windows OS. The next step could be to extend the current research and study multiple platforms, leading to more constructs in the conceptual model, such as platform competencies or preferential linkages.

Second, although our research question is centered around the kinds of factors that influence a software companies' speed of decision to form alliances with Microsoft on Windows OS, we do not focus on the concept of platforms and its value appropriation capability. Recently the concept of architectural control points (ACP) has drawn a lot of attention[18]. An ACP is a design attribute that can be used by a stakeholder to manage the flow of information or control between connected components of a modular system. APIs (application program interfaces) for Windows discussed under "inversion" could be regarded as architectural control points for Windows. A related question would be — what are the interrelationships between design operators of platforms and ACPs? Can certain design operators improve or impede management of ACP?

Third, in this study we explore the impact of two design operators (augmenting and inversion) as the main factors in the conceptual models. Other factors, such as a software firms' technical interests or focus, platform characteristics etc., may improve the fitness of the model. Another limitation is the use of annual sales as a proxy measure for network effects. A more accurate measure of Windows OS would be shipped units.

Fourth, as stated in the data section, the Stata program automatically drops data after year 1994 because there is no significant difference for each predictors of heterogeneity in hazard. The next interesting step could be to analyze the case of multiple events per ISV i.e., repeated event design.

Finally, the research setting for this article is the packaged software industry. The term platform here refers to the operating systems. One extension could be changing the research setting to supply chain management and examine the business platform.

7. Summary

In summary, the contribution of this study lies in using a new theoretical foundation— design rules theory—to explore the timing of an alliance with a platform vendor. We believe this to be one of the first studies to operationalize design operators and apply them in an empirical setting. The results of this study provide a different theoretical perspective for strategic and organizational scholars to closely examine alliance formations.

Acknowledgement

I thank the Boston University School of Management's Institute for Leading in a Dynamic Economy (BUILDE) for providing support for completion of this study. I gratefully acknowledge the advice and support of Professor John Henderson through the whole course of this project. I also thank Professor George Wyner for his insightful comments on this paper. Thanks also go to Professor Stephanie Watts for commenting on earlier drafts of the article.

References

[1] Arthur, W.B. Competing Technologies, Increasing Returns, and Lock-in by Historical Events. *Economic Journal*, Blackwell Publishing Limited, 1989, 116.

[2] Axelrod, R., Mitchell, W., Thomas, R.E., Bennet, D.S. and Bruderer, E. Coalition Formation in Standard-Setting Alliances. *Management Science*, *41* (9). 1493-1507.

[3] Baldwin, C.Y. and Clark, K.B. *Design rules: the power of modularity*. The MIT Press, Cambridge, MA, 2000.

[4] Barley, S., Freeman, J. and Hybels, R.C. Strategic alliances in commercial biotechnology. in Eccles, R. ed. *Networks and organizations: Structure, form and action*, Harvard Business School Press, Boston, 1992, 311-347.

[5] Bresnahan, T., New modes of competition: Implications for the future structure of the computer industry. in *Prepared for the Progress and Freedom Foundation Conference on Competition, Convergence and the Microsoft Monopoly.*, (Mimeo, 1998).

[6] Bresnahan, T. and Greenstein, S. Technological competition and the structure of the computer industry. *The Journal of Industrial Economics*, *47* (1). 1-40.

[7] Burgers, W.P., Hill, C. and Kim, W. A theory of global strategic alliances: The case of the global auto industry. *Strategic Management Journal*, *14* (6). 419-432.

[8] Choi, J. Network externalities, compatibility choice, and planned obsolescence. *Journal of Industrial Economics*, *42*. 167-182.

[9] Cohen, W.M. and Levinthal, D.A. Absorptive Capacity: A New Perspective on Learning and Innovation. *Administrative Science Quarterly*, *35*. 128-152.

[10] Cottrell, T. Software variety and hardware value: A case study of complementary network externalities in the microcomputer software industry. *Journal of Engineering and Technology Management*, *15*. 309-339.

[11] Dana Jr., J.D. Remark on "Appropriateness and Impact of Platform-Based Product Development". *Management Science*, INFORMS: Institute for Operations Research, 2003, 1264-1267.

[12] Farrell, J. and Saloner, G. Standardization, Compatibility, and Innovation. *The RAND Journal of Economics*, *16* (1). 70-83.

[13] Fichman, R.G. Real Options and IT Platform Adoption: Implications for Theory and Practice. *Information Systems Research*, INFORMS: Institute for Operations Research, 2004, 132-154.

[14] Garud, R. and Kumaraswamy, A. Changing competitive dynamics in network industries: An exploration of Sun Microsystems' open systems strategy. *Strategic Managment Journal*, *14*. 351-369.

[15] Gawer, A. and Cusumano, M.A. *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation.* Harvard Business School Press, Cambridge, MA, 2002.

[16] Gulati, R. Social structure and alliance formation:.

[17] Huber, P.J., The behavior of maximum likelihood estimates under non-standard conditions. in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, (Berkeley, 1967), University of California Press, 221-233.

[18] Iyer, B. and Dreyfus, D. Architectural control and emergent architecture: a network perspective, Boston University Working Paper, 2005.

[19] Katz, M.L. and Shapiro, C. Technology Adoption in the Presence of Network Externalities. *Journal of Political Economy*, *94* (4). 822.

[20] Kauffman, R.J., McAndrews, J. and Wang, Y.-M. Opening The 'Black Box' of Network Externalities in Network Adoption. In*formation Systems Research,* 11 *(*1). 61-82.

[21] Khazam, J. and Mowery, D. The commercialization of RISC: Strategies for the creation of dominant designs. Re*search Policy,* 23. 89-103.

[22] Kristiansen, E.G. R&D in the presence of network externalities: Timing and compatibility. Ra*nd Journal of Economics,* 29. 531-548.

[23] Liebowitz, S.J. and Margolis, S.E. Network Externalities (Effects) En*try in The New Palgrave's Dictionary of Economics and the Law, MacMillan, 1998,* 1998.

[24] Madnick, S. Direct testimony, 2002.

[25] Nohria, N. and Garcia-Pont, C. Global strategic linkages and industry structure. St*rategic Managment Journal,* Su*mmer Special Issue (*12). 105-124.

[26] Polanyi, M. The *Tacit Dimension*. Routledge and Kegan Paul, London, 1966.

[27]Ranger-Moore, J. 1997. Bigger may be better, but is older wiser? *American Sociological Review*, *62*. 903-920

[28] Rajagopalan, S., Singh, M.R. and Morton, T.E. Capacity expansion and replacement in growing markets with uncertain technological breakthroughs. Ma*nagement Science,* INFORMS: Institute for Operations Research, 1998, 12.

[29] Rogers, E.M. New Product Adoption and Diffusion. Jo*urnal of Consumer Research,* 2 *(*March). 290-301.

[30] Sahay, A. and Riley, D. The Role of Resource Access, Market Considerations, and the Nature of Innovation in Pursuit of Standards in the New Product Development Process. The *Journal of Product Innovation Management,* 20. 338-355.

[31] Schilling, M. Technological Lock Out: An Integrative Model of the Economic and Strategic Factors Driving Technology Success and Failure. Ac*ademy of Management Review,* 23 *(*2). 267-284.
[32] Schilling, M.A. Technology Leapfrogging: Lessons from the U.S. Video Game Console Industry. Ca*lifornia Management Review,* 45 *(*3). 6-32.
[33] Schilling, M.A. Toward a general modular systems theory and its application to interfirm product modularity. Ac*ademy of management review,* 25 *(*2). 312-334.
[34] Shan, W., Walker, G. and Kogut, B. Interfirm cooperation and startup innovation in the bio-technology industry. St*rategic Management Journal,* 15. 387-394.
[35] Shapiro, C. and Varian, H.R. In*formation Rules: A strategic guide to the network economy*. Harvard Business School Press, Cambridge, MA, 1998.
[36] Shurmer, M. An investigation into sources of network externalities in the packaged PC software market. In*formation Economics and Policy,* 5. 231-252.
[37] Simon, H.A. The *sciences of the artificial*. MIT Press, Cambridge, Massachusetts, 1962.
[38] Sorensen, J.B. and Stuart, T.E. Aging, Obsolescence, and Organizational Innovation. Ad*ministrative Science Quarterly,* 45. 81-112.
[39] Stuart, T.E. Network Positions and Propensities to Collaborate: An Investigation of Strategic Alliance Formation in a High-technology Industry. Ad*ministrative Science Quarterly,* Administrative Science Quarterly, 1998, 668.
[40] Teece, D. Competition, Co-operation, and Innovation, Organizational Arrangements for Regimes of Rapid Technological Progress. Jo*urnal of Economic Behavior and Organization,* 18 *(*1). 1-25.
[41] Teece, D. Profiting from Technological Innovation, Implications for Integration, Collaboration, Licensing, and Public Policy. Re*search Policy,* 15. 285-305.
[42] Venkatraman, N., Loh, L. and Koh, J. The Adoption of Corporate Governance Mechanisms: A Test of Competing Diffusion Models. Ma*nagement Science,* 40 *(*4). 496-507.
[43] White, H. Maximum likelihood estimation of misspecified models. Ec*onometrica,* 50. 1-25.
[44] Winter, H. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. Ec*onometrica,* 48. 817-830.

APPENDIX A: WINDOWS PRODUCTS RELEASE CODING SCHEME

INTRODUCTION

This document describes the coding procedure used to analyze each version of a Windows operating systems release. The announcements and descriptions of Windows OS releases are collected from three major sources: Microsoft Homepage (http://www.microsoft.com/windows/), http://en.wikipedia.org/wiki/microsoft_windows, and Lexis-Nexis (http://web.lexis-nexis.com).

The goal of coding is to identify the design operators involved in each new version of the operating systems. For each version more than one operator could be used and one operator could also be used more than once. If this case, we coded and recorded all operators and the number of times they were used.

MODULAR OPERATORS

Modular operators considered in this research are:

1. Augmenting:

"Augmenting means adding a module. (p.135)" Augmenting only occurs to already modularized design. By augmenting, the user adds a module to give the system some new functionality (p.136). Augmenting is the operator underlying almost each operating systems release in this study. Through augmenting, the OS can gain new software functionality. Keywords/typical phrases that could suggest augmenting include, but are not limited to: include new features/support/functionalities into Windows OS; add a new module into Windows OS.

2. Inversion:

"The operator *inversion* describes the action of taking previously hidden information and 'moving it up' the design hierarchy so that it is visible to a group of modules." (p.138) Inversion changes the hierarchy of a module. It makes the *inverted*-module visible by more modules (at a lower level). In other words, the inverted-module now can serve multiple modules.

Inversion takes place when a module within the Windows OS is made available to multiple software systems. The following scenarios are examples of inversion:

- Software development tool-kits that can be used, either by self or by customers, to develop new systems or upgrade existing systems.
- A software module that can be integrated into more than one software system. For example, Windows 95 integrated a 32-bit TCP/IP stack for built-in Internet support that made easier for users to install hardware and software.

Keywords/typical phrases that could suggest inversion include, but are not limited to: integrating into (Windows); tool-kits (such as developing tools, software libraries, not client tools in C/S environment); API (application programming interface)

APPENDIX B
WINDOWS DESKTOP OS DESIGN MOVES

Date	Version	Augment	Invert
11/1985	Windows 1.0	11	1
12/9/1987	Windows 2.0	3	
12/9/1987	Windows/386 2.1	1	1
06/28/1988	Windows/286 2.1	2	
05/22/1990	Windows 3.0	6	
03/18/1992	Windows 3.1	2	
07/1993	Windows NT Advanced Server 3.1	5	
08/1993	Windows NT 3.1	6	
11/11/1993	Windows for Workgroups 3.11	3	
1993	Windows NT Workstation 3.5	2	
1994	Windows NT Server 3.5	2	
06/1995	Windows NT Server 3.51	2	
08/24/1995	Windows 95	2	
04/1996	Windows 3.2		
07/29/1996	Windows NT Workstation 4.0	1	
07/1996	Windows NT Server 4.0	5	
1997	Windows NT Server 4.0, Enterprise Edition	2	
06/25/1998	Windows 98	6	1
09/1998	Windows NT Server 4.0, Terminal Server Edition	7	1
06/10/1999	Windows 98 Second Edition	1	
02/17/2000	Windows 2000 Professional	6	1
09/14/2000	Windows Millennium Edition (Windows ME)	1	1
10/25/2001	Windows XP		2
12/31/2001	Windows XP Professional	6	
12/31/2001	Windows XP Home Edition		
2001	Windows XP 64-bit Edition		
10/2002	Windows XP Media Center Edition	1	
04/2003	Windows Server 2003	1	1
2003	64-bit Operating Systems		