CONTROL IN APP PLATFORMS: THE INTEGRATION-DIFFERENTIATION PARADOX

Research-in-Progress

Chris Maurer
University of Georgia
316A Brooks Hall, Athens, GA 30602
cmaurer@uga.edu

Amrit Tiwana
University of Georgia
305 Brooks Hall, Athens, GA 30602
tiwana@uga.edu

Abstract

Despite violations of several common assumptions regarding the use of control, platform ecosystems rely heavily on such mechanisms. This calls into question whether controls serve purposes other than just the alignment of their participants' interests, as the extant literature suggests. We develop a model that explains how *and* why *control mechanisms influence performance of applications in a mobile computing platform ecosystem. Our model uncovers an apparent paradox in that controls influence performance because they promote knowledge integration but simultaneously inhibit differentiation. Knowledge is embedded within controls, which allows developers to leverage it in developing high-quality applications. However, leveraging the same knowledge stock inhibits the extent to which developers can differentiate their work from others. Taking into consideration endogenous instruments of control, we will test our model using data collected from Research-in-Motion's Blackberry AppWorld. Our results will extend the current understanding of the purpose and use of controls in platforms.*

Keywords: Platform Ecosystems, Software Development, IS Control

Introduction

The 'app economy', or generally the market driven by the sale of mobile computing devices and the apps or other digital goods that are consumed on such devices, generated an estimated $20 billion in revenues in 2011 with expectations of over $75 billion in revenues by the year 2015 (Rubinson Partners, Inc. 2011). The core infrastructure supporting the app economy is a collection of software-based platforms, defined as "the extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they interoperate" (Tiwana et al. 2010). These platforms create an overall ecosystem in which numerous participants, including the platform owner (such as RIM, Google, or Apple), app developers and end users, all transact with one another in complex ways. The astounding growth of these platform ecosystems offers IS researchers unique and unprecedented research and theory development opportunities.

Software-based platform ecosystems have significantly blurred conventional firm boundaries by incorporating third-party app developers into innovation activities that were historically done in-house or contracted to outside partners. The premise is that a large pool of outside complementors with diverse expertise and market-like motivations will out-innovate a smaller scale in-house department. The challenge however is that the work of such complementors (hereafter simply app developers) must be integrated into the platform without sacrificing differentiation in their work. Of particular importance is the notion of organizational control in which one party (the controller) enforces some form of monitoring and evaluation on another party (the controllee). The platform owner serves as the controller and the app developer the controlee in a platform ecosystem. It is worth noting that in a platform setting, an app developer can be an individual human being or a group of individuals working together as an organization. For the purposes of this paper, we use the term app developer to refer to an organization that produces a complementary app, even though some are comprised of only a single person. Controls therefore exert an influence on all individuals working on app development activities, even if they are part of a larger organization.

While previous research has established a link between control and performance of IS development projects (Gopal and Gosain 2010; Tiwana and Keil 2007, 2009) thus suggesting that control may improve performance in a platform, four key assumptions in the traditional controls literature are violated to some degree in platforms. First, unlike traditional settings where control is used as a mechanism for aligning the interest of two divergent parties (Ouchi 1979), the interests of the platform owner and app developer are not necessarily misaligned. This is because both parties are highly motivated to sell apps in the marketplace. Second, the platform ecosystem resembles a classic market more than it does a principal-agent relationship, a situation in which overt control beyond price mechanisms is rarely needed (Ouchi 1979, 1980). Third, unlike traditional systems development projects with a defined start and end point where requirements are compiled, budgets are set, and timelines are enforced (Tiwana and Keil 2007, 2009), formal requirements, delegation by a principal, and a definitive end-point are typically absent in platform app development activities. Finally, unlike dyadic client-developer relationships studied in the IS controls literature, the sheer scale of the number of app developers can make traditional control mechanisms prohibitively costly (e.g., Apple's iOS platform has 600,000 app developers working on individual projects that the platform owner cannot readily keep up with). These violations of commonly held assumptions for the effective use of control cast some doubt on the interest-alignment motivation behind using control mechanisms and the purposes they serve in a platform.

Although the conditions associated with the use of control mechanisms are absent, a variety of control mechanisms are widely observed in platforms. For example, platform owners (such as RIM and Apple) review all applications to ensure they meet certain criteria before making them available in the public marketplace. Further, platform owners issue development guidelines and provide standardized development & testing environments in the form of software development kits to control the manner in which development work is performed by app developers. The presence of these control mechanisms raises the question of whether controls serve other coordinative functions beyond just the divergent-interest-alignment explanation in existing controls research. In summary, existing theory provides limited insight into the manner in which control influences the market performance of applications in a platform ecosystem. Further, we know little regarding the explanatory mechanisms that describe why control exerts an influence on application performance in the marketplace. Thus, we are guided by the following research question:

How *and* why *do different control mechanisms influence the performance of applications in a platform ecosystem?*

Theoretical Development

Control refers to any attempt by one party (the controller) to influence or motivate another party (the controllee) to act in accordance with the controller's goals and/or objectives (Ouchi 1979). Researchers have broken control into two main categories, each with several types of control: formal control (process, output and input control) and informal control (clan and self control). *Process* control is *the extent to which the application developer follows pre-specified methods and procedures for developing updates to an application* (Kirsch et al. 2002). *Input* control is defined as *the extent to which the application developer positions updates to its application to have a higher likelihood of being accepted by the platform owner for distribution on the platform. Clan* control can be defined as *the extent to which the application developer adapts its behaviors in an effort to embrace the set of beliefs, norms, and values the platform owner attempts to foster among its developer community (*Ouchi 1979*). Self* control is *the extent to which the application developer autonomously defines goals and executes procedures to accomplish those personal goals (*Henderson and Lee 1992*).* Output control is omitted from our model because we are interested in controls enforced by the platform owner. End users in the marketplace enforce output control using financial rewards and penalties to developers based on their published applications.

Note that the aforementioned definitions focus on the notion of *realized*, not *attempted*, control. Traditional control studies focus on attempted control since they measure the extent to which a controller takes certain actions. The notion of realized control, or "*the extent to which the controller is able to successfully exercise a given control mechanism*" (Tiwana and Keil 2009), has been mentioned in passing without any further theoretical development or empirical investigation. All of our definitions of control focus on the manner in which a controllee abides by an attempted control mechanism and the behavioral changes that result because of this. We believe this is an important distinction because not every attempt at enforcing a control mechanism is successful in achieving its intended goal. Indeed, Ouchi (1978) states that "let us note that, whether the control process is based on behavior or on output, it is always behavior that is the ultimate objective of feedback and change" (p. 175). This quote refers to the fact that the key consideration of a control mechanism is not whether it is merely communicated, but whether it results in some behavioral change desired by the controller. The platform context offers an unusually appropriate setting to theoretically develop the notion of realized control because one set of control mechanisms is attempted across a wide population of controlleees and it is impossible to actively monitor all control mechanisms. Since the platform owner cannot fully monitor the actions and behaviors of every single app developer in the platform ecosystem, we expect there to be variance in the extent to which control mechanisms are realized. Further, developers do not randomly select control mechanisms to abide by – certain factors influence the extent to which they abide by various control mechanisms. We therefore identify and evaluate these factors as endogenous instruments in our proposed model.

Existing measures of performance in control research have included ISD performance (Tiwana and Keil 2009), alliance performance (Tiwana and Keil 2007), and project efficiency/quality (Gopal and Gosain 2010). All measures focus on the controller's evaluation of the work performed by the controllee but we are ultimately interested in how the marketplace evaluates an application. Therefore, our dependent variable is relative market performance, defined as *the marketplace's evaluation of an application's quality relative to other similar applications offered in the same platform ecosystem.* We argue that application developers join a platform in hopes of achieving high levels of performance for their applications. In joining a platform, an application developer is able to draw on expertise beyond its own existing stock of knowledge however strictly relying on the provided stock of knowledge inhibits the ability to be individualistic and differentiate one's app from others. This apparent paradox between integrating knowledge provided by the platform owner and differentiating an application from competitors is discussed next.

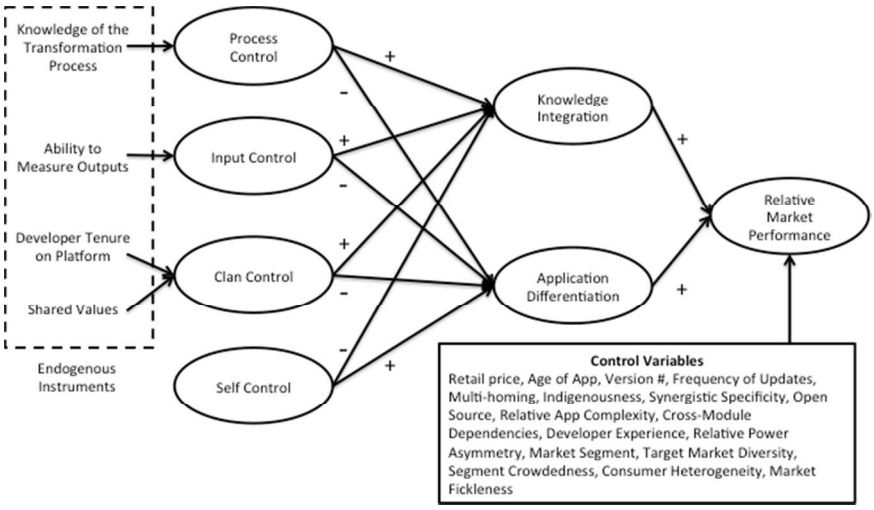

Figure 1: Proposed Research Model

Knowledge Integration

We define *knowledge integration* as the extent to which the application developer coordinates its existing stock of knowledge with that of the platform owner in the development of an application. The concept of knowledge integration focuses on two primary components, coordination and application of specialized knowledge (Mitchell 2006; Tiwana and McLean 2005) – both of which are emphasized in our definition.

In Grant's (1996a) knowledge-based view of the firm, it is argued that knowledge is the most strategically important resource a firm possesses. However, it is imperative that the firm effectively coordinates and leverages its stock of knowledge in order to achieve competitive advantage. As demonstrated in the study of system development (Faraj and Sproull 2000) and enterprise application integration (Mitchell 2006), effectively integrating knowledge results in greater performance than relying only on a sole source of knowledge thereby suggesting that performance of an application will be increased when developers effectively integrate knowledge from multiple sources.

One potential source for external knowledge is a control mechanism enforced by the platform owner. Control mechanisms are not mindlessly derived – they are based on previous experience and knowledge of the work that is to be performed. Control mechanisms can serve as the mechanism by which knowledge is transferred from the controller to the controllee. This is true for both explicit (or easily codified knowledge) and tacit (or experiential-based) knowledge, however explicit knowledge is far easier to transfer (Grant 1996b). Explicit knowledge can be transferred via simple communication in the form of written procedures, rules, or specifications. Tacit knowledge, on the other hand, requires immersion and observation in order to be effectively transferred (Nonaka 1994). Formal control mechanisms, therefore, serve as a mechanism for the controller to communicate explicit knowledge to the controllee through written instruction and rules. Alternatively, informal control mechanisms, such as clan control, involve immersion and fostering relational capital, which provides a mechanism for the transfer of tacit knowledge. The transfer of knowledge is not sufficient for gains in performance however (Tiwana and McLean 2005), thus we do not expect to see a direct effect of control on performance. Control will only influence performance because this knowledge is effectively integrated into the development of an application.

Since most platform owners also develop applications for their own platforms, they have substantial experience in the development process of taking an idea and translating it into an application. This experience results in codified knowledge that is embedded within process control mechanisms. When developers abide by process control mechanisms, it gives them the opportunity to integrate the procedural knowledge embedded in control, thus allowing the developer to leverage the best practices laid forth by the platform owner. In doing so, performance of the application that is developed should improve.

Hypothesis 1A: Greater use of process control will enhance the relative market performance of an application because *it increases the degree to which knowledge is integrated by the application's developer.*

For input control, platform owners embed knowledge concerning the characteristics of an acceptable high-quality application. By providing criteria that will be assessed in the application review process as well as justifications for why applications are rejected, the platform owner codifies knowledge concerning the characteristics of a high-quality application. With repeated interactions, the developer gains a better understanding of the platform owner's expectations of a high-quality app. This will result in the development of applications that are more successful.

Hypothesis 1B: Greater use of input control will enhance the relative market performance of an application because *it increases the degree to which knowledge is integrated by the application's developer*

In terms of clan control, tacit knowledge is deeply embedded within the norms and routines of the clan (here, producer-participants in the platform's ecosystem). Members of the clan have fully internalized these shared values and therefore have access to the tacit knowledge provided by other members of the clan (Ouchi 1980). Integrating the tacit knowledge embedded in the clan control mechanisms during the development process will improve the performance of the resultant application.

Hypothesis 1C: Greater use of clan control will enhance the relative market performance of an application because it increases the degree to which knowledge is integrated by the application's developer.

Since self control often consists of evaluation criteria that rewards autonomy and self-management of the controllee (Kirsch et al. 2002), high levels of self control require the controllee to act independently in governing its own actions. Within our context, a high degree of self control results in the developer believing it can develop a high-quality application without interference or monitoring on the part of the platform owner. As such, the developer eschews the opportunity to gain knowledge from the platform owner. Therefore, the market performance of an application developed under a high degree of self control will be purely the result of the capabilities and knowledge stores of the application developer. Since there is a limited amount of knowledge transfer between the platform owner and the developer, there can only be a minimal amount of knowledge integration that occurs and performance should suffer.

Hypothesis 1D: Greater use of self control will reduce the relative market performance of an application because it decreases the degree to which knowledge is integrated by the application's developer.

Application Differentiation

We define *application differentiation as the extent to which an application and subsequent versions of the application can be distinguished on actual or perceived characteristics vis-à-vis direct competitors by the marketplace.* One of the earliest conceptualizations of product differentiation comes from Shaw (1912) in which the concept is described as a means by which the producer stimulates demand for its product by changing certain characteristics of the product to distinguish it from the other stock commodities. Later conceptualizations have expanded the definition to include characteristics for differentiation to be either real or imagined (Chamberlin 1962).

The marketplace for applications in a platform setting is typically crowded (Tiwana et al. 2010). If all apps had product attributes that are effectively equal, then the app marketplace would resemble a commodity market where each producer would claim equal share of the market (Dickson and Ginter 1987). In such commodity markets, price becomes the most critical factor in differentiating one product from another (Dickson and Ginter 1987). However, application developers often strive to produce applications that are differentiated on factors other than price, especially since there is a floor at which price can no longer be lowered (developers cannot pay users to download their applications). When an application is sufficiently differentiated from other similar commoditized products on factors other than price, it will increase the demand for the application so long as the differentiated product meets the customer's needs (Chamberlin 1962). In other words, consumers in the marketplace will be more willing

to purchase applications that are appropriately differentiated from close competitors on attributes that consumers value.

As was demonstrated in the prepackaged software industry, software vendors can improve their market share by purposely reconfiguring resource allocations and the functionality of their products to differentiate their products and capitalize on complementarities requested by members in the marketplace (Lee et al. 2010). The prepackaged software industry closely resembles the characteristics of applications competing within a platform ecosystem. In both settings, we see extreme competition characterized by software developers rapidly responding to changing user needs. There are also network effects (Katz and Shapiro 1994) present in both settings because the number of users of a particular system or application can influence the extent to which potential users wish to utilize the same system. We can therefore conclude that application developers who are cognizant of the shifting competitive landscape and who appropriately improve their applications to differentiate them from imitators and other competitors will earn a greater share of the market and increased market performance. Further, the effects of such differentiation can increase significantly in magnitude in the presence of network effects because gaining traction within a user base can quickly increase the demand from other potential users (Lee et al. 2010).

Whereas our first set of hypotheses discusses how control mechanisms enable knowledge integration, our second set of hypotheses discusses an alternate perspective. Increasing the extent to which knowledge is integrated often makes it difficult to properly differentiate one's app when the sources of knowledge being integrated are the same across competitors. This paradox is similar to the difficulties organizations face in maximizing the antagonistic states of integration and differentiation found in Lawrence & Lorsch's (1967) seminal article.

When rules and regulations are highly formalized, parties performing work are less likely to experiment (March 1958). Furthermore, when a party is subject to increased monitoring in the process of getting work accomplished, it will consider the monitoring and measurement methods of the controller to be obtrusive thus resulting in a lower level of commitment to getting the job done (Ouchi 1979). Such monitoring can also lead the developer to focus more on complying with the stated procedural rules than on thinking outside the box in order to innovate thereby decreasing the extent to which the resultant application is differentiated from competitors.

> *Hypothesis 2A: Greater use of process control will reduce the relative market performance of an application* because *it decreases the extent to which the application is differentiated from close competitors.*

The application review process (input control) in a platform is meant to ensure applications abide by certain standards and requirements laid forth by the platform owner. In short, the platform owner reserves the right to reject any application that appears to violate the rules of the platform. Hence, applications that attempt to implement creative and innovative uses of the platform in hopes of differentiating their applications are at a higher risk of being rejected by the platform owner. It then appears as though the use of input control actually dissuades developers from utilizing the platform's functionality in creative and novel ways, which is necessary to properly differentiate one's application from competitors (Dickson and Ginter 1987).

> *Hypothesis 2B: Greater use of input control will reduce the relative market performance of an application* because *it decreases the extent to which the application is differentiated from close competitors*

Clans survive because of a strong commitment to shared values and high levels of goal congruence (Ouchi 1980). Empirical evidence suggests that employees in an organization resembling a clan have higher levels of attachment to the organization thus resulting in behaviors that would not violate the set of values and beliefs shared by members of the clan (Ouchi and Johnson 1978). Clans also thrive in an environment where there is great ambiguity in attempting to measure individual performance (Ouchi 1980). This is primarily due to the fact that individuals within the clan are not seeking to outshine others in the clan: everybody is working toward a collective goal. Further, the clan does not reward opportunistic behaviors on the part of its members (Ouchi 1980) and therefore there is little incentive to try to gain advantage over another application developer.

Hypothesis 2C: Greater use of clan control will reduce the relative market performance of an application because it decreases the extent to which the application is differentiated from close competitors

Despite not having access to the knowledge embedded in explicit forms of control, there could be benefits associated with the use of self control. As Tiwana & Keil (2009) noted, self control increases performance in internally developed IS projects because it allows the development team the autonomy to creatively solve problems so long as the controller trusts the development team to work towards a collective goal. If every app developer is drawing on the same set of guidelines and rules for developing applications, it is less likely that there is variance in the quality of applications produced and distributed in the platform ecosystem. By allowing self control, platform owners grant the application developers the freedom to innovate freely.

Hypothesis 2D: Greater use of self control will enhance the relative market performance of an application because it increases the extent to which the application is differentiated from close competitors

Endogeneity of Realized Control

Realized control is endogenous in that app developers choose the degree to which they abide by controls issued by the platform owner, a nuance that that has previously been overlooked in IS controls research (e.g., Rustagi et al. 2008; Tiwana and Keil 2009). Failure to correct for endogeneity can result in biased estimates and incorrect conclusions (Shaver 1998). We use four variables as instrumental variables in the model to correct for endogeneity. First, the perception of the platform owner's ability to evaluate submitted applications (e.g. ability to measure outputs) influences input control because it has been shown that a controller who cannot adequately assess the end result of a work task cannot properly enforce this type of control (Kirsch et al. 2002). Knowledge of the transformation (or more specifically, development) process (Ouchi 1979) positively influences process control because the controller must understand the work processes in order to properly monitor and evaluate them. Clans have high social requirements and therefore take a long time to form, thus *developer tenure* on the platform influences the realization of clan control. Further, clan control is influenced by the presence of *shared values*, because shared values are a necessary requirement for clans to form (Ouchi 1979).

Rival Explanations of Relative Market Performance

We are cognizant of the fact that many other factors can influence the performance of applications in a platform marketplace. It is true that most developers multi-home (Armstrong 2006), or participate in development activities across multiple platforms. The fact that developers enter into a platform ecosystem with varying levels of experience and knowledge requires us to consider numerous rival explanations of market performance. Rival explanations can be attributed to characteristics of 1) the application, 2) the developer, 3) the marketplace, and 4) the application's category. A full list of our control variables is provided in Figure 1.

Methodology

To test the proposed research model, we will employ a cross-sectional survey to collect primary data. The unit of analysis for this study is the software application. The context of this study is Research In Motion's (RIM) BlackBerry platform and it was chosen for two reasons. First, the BlackBerry platform is not licensed for use on hardware that is produced by vendors other than Research In Motion. The fact that the platform and applications can only run on RIM devices allows us to control for potential confounding effects that result from hardware incompatibilities. Second, the control structure employed in the BlackBerry platform also align closely with that of other competing platforms thus allowing our findings to be generalizable to the broader population of platforms. The sampling frame is a stratified random sample of 1,500 apps in BlackBerry's app marketplace (App World), excluding e-book and themes categories (as they emphasize content rather than functionality). We also eliminated those applications that have received less than 5 user-submitted reviews since user reviews are a component of our dependent variable and we need an adequate number of reviews to ensure robustness of this measure.

This resulted in a total of 8,024 unique applications and 3,479 unique developers. Each developer appears only once in our sample population.

Measures. We intend to follow rigorous scale development and validation procedures as outlined by Babbie (1990). Where possible, existing scales will be utilized or adapted. We recognize that a survey methodology measures the respondent's perception of a specific phenomenon. To this end, we will construct our measures in an attempt to focus on the actual behaviors that app developers exhibit in response to the different forms of control. This approach should allow us to capture the extent to which a control was realized, and not simply a developer's perception of the efficacy of the control mechanism. Table 1 provides a summary of our measures.

Table 1: Construct Measures

Construct	Source of measurement
Dependent Variable	
Average App Rating, Average rating X # of ratings	z-scores of values gathered from Blackberry App World
# of downloads	z-scores of values provided by survey respondent
Types of Control	
Process, Clan, Self Control	Adapted from Kirsch et al. (2002)
Input Control	Newly Developed Scale
Explanatory Mechanisms	
Knowledge Integration	Adapted from Mitchell (2006)
Application Differentiation	Adapted from Chaudhuri & Holbrook (2001)
Endogenous Instruments	
Ability to Measure Outputs, Knowledge of the Transformation Process	Adapted from Kirsch et al. (2002)
Shared Values	Newly developed scale
Developer Tenure	Single item measure administered in survey

Data Collection and Analysis. A pilot test will be conducted with a small group of Blackberry developers to further refine the items. Once primary data are collected, the measurement model will be evaluated and psychometric properties will be assessed through confirmatory factor analysis. Two-stage, instrumental variables hierarchical regression will be used to test the hypotheses for two reasons. First, hierarchical regression allows us to account for rival explanations of relative market performance by including our control variables as the first step in the hierarchical regression. Finally, it allows accounting for endogeneity of realized control in the model (Heckman 1979) as well as mediation (Sobel 1982).

Conclusion

Our intended theoretical contribution is two-fold. First, we extend our current understanding of control to include purposes beyond just aligning interests of two parties. To do so, we situate control in a platform ecosystem, thus allowing us to examine nuances of control, such as realized control, that have never been empirically examined. Another unique theoretical contribution lies in the theoretical development of the idea that all control mechanisms carry certain costs and that these costs may change the direction and/or strength of the influence control has on performance. Our acknowledgement that realized control is at least partly influenced by endogenous factors also provides both a theoretical and methodological contribution to the existing stream of research on control. We therefore believe that this study contributes new insights into the micro-processes of how and why control mechanisms shape performance in competitive markets for specialized software complements.

References

Armstrong, M. 2006. "Competition in two-sided markets," *RAND Journal of Economics (RAND Journal of Economics)* (37:3), pp. 668–691.

Babbie, E. R. 1990. *Survey Research Methods, Second Edition*, (2nd ed,)Wadsworth Publishing.

Chamberlin, E. 1962. *The theory of monopolistic competition; a re-orientation of the theory of value.*, Harvard economic studies, (8th ed.,)Harvard University Press,.

Chaudhuri, A., and Holbrook, M. B. 2001. "The Chain of Effects from Brand Trust and Brand Affect to Brand Performance: The Role of Brand Loyalty," *Journal of Marketing* (65:2), pp. 81–93.

Dickson, P. R., and Ginter, J. L. 1987. "Market Segmentation, Product Differentiation, and Marketing Strategy," *Journal of Marketing* (51:2), pp. 1–10.

Faraj, S., and Sproull, L. 2000. "Coordinating Expertise in Software Development Teams," *Management Science* (46:12), pp. 1554.

Gopal, A., and Gosain, S. 2010. "The Role of Organizational Controls and Boundary Spanning in Software Development Outsourcing: Implications for Project Performance.," *Information Systems Research* (21:4), pp. 960–982.

Grant, R. 1996a. "Toward a Knowledge-Based Theory of the Firm," *Strategic Management Journal* (17), pp. 109–122.

Grant, R. 1996b. "Prospering in dynamically-competitive environments: Organizational capability as knowledge integration," *Organization Science* (7:4), pp. 375–387.

Heckman, J. J. 1979. "Sample Selection Bias as a Specification Error," *Econometrica* (47:1), pp. 153–161.

Henderson, J. C., and Lee, S. 1992. "Managing I/S Design Teams: A Control Theories Perspective," *Management Science* (38:6), pp. 757–777.

Katz, M. L., and Shapiro, C. 1994. "Systems Competition and Network Effects," *The Journal of Economic Perspectives* (8:2), pp. 93–115.

Kirsch, L. J., Sambamurthy, V., Dong-Gil Ko, and Purvis, R. L. 2002. "Controlling Information Systems Development Projects: The View from the Client," *Management Science* (48:4), pp. 484–498.

Lawrence, P., and Lorsch, J. 1967. "Differentiation and Integration in Complex Organizations," *Administrative Science Quarterly* (12:1), pp. 1–47.

Lee, C.-H., Venkatraman, N., Tanriverdi, H., and Iyer, B. 2010. "Complementarity-based hypercompetition in the software industry: Theory and empirical test, 1990-2002," *Strategic Management Journal* (31:13), pp. 1431–1456.

March, J. G. 1958. *Organizations*, New York: Wiley.

Mitchell, V. L. 2006. "Knowledge Integration and Information Technology Project Performance," *MIS Quarterly* (30:4), pp. 919–939.

Nonaka, I. 1994. "A Dynamic Theory of Organizational Knowledge Creation," *Organization Science* (5:1), pp. 14–37.

Ouchi, W. G. 1978. "The Transmission of Control Through Organizational Hierarchy.," *Academy of Management Journal* (21:2), pp. 173–192.

Ouchi, W. G. 1979. "A Conceptual Framework for the Design of Organizational Control Mechanisms," *Management Science* (25:9), pp. 833–848.

Ouchi, W. G. 1980. "Markets, Bureaucracies, and Clans.," *Administrative Science Quarterly* (25:1), pp. 129–141.

Ouchi, W. G., and Johnson, J. B. 1978. "Types of Organizational Control and Their Relationship to Emotional Well Being," *Administrative Science Quarterly* (23:2), pp. 293–317.

Rubinson Partners, Inc. 2011. *How Big is the US App-Economy? Estimates and Forecasts 2011-2015*, .

Rustagi, S., King, W. R., and Kirsch, L. J. 2008. "Predictors of Formal Control Usage in IT Outsourcing Partnerships.," *Information Systems Research* (19:2), pp. 126–143.

Shaver, J. M. 1998. "Accounting for endogeneity when assessing strategy performance: Does entry mode choice affect..," *Management Science* (44:4), pp. 571–585.

Shaw, A. W. 1912. "Some Problems in Market Distribution," *Quarterly Journal of Economics* (26:4), pp. 703–765.

Sobel, M. E. 1982. "Asymptotic Confidence Intervals for Indirect Effects in Structural Equation Models," *Sociological Methodology* (13), pp. 290–312.

Tiwana, A., and Keil, M. 2007. "Does peripheral knowledge complement control? An empirical test in technology outsourcing alliances," *Strategic Management Journal* (28:6), pp. 623–634.

Tiwana, A., and Keil, M. 2009. "Control in Internal and Outsourced Software Projects.," *Journal of Management Information Systems* (26:3), pp. 9–44.

Tiwana, A., Konsynski, B., and Bush, A. A. 2010. "Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics.," *Information Systems Research* (21:4), pp. 675–687.

Tiwana, A., and McLean, E. R. 2005. "Expertise Integration and Creativity in Information Systems Development," *Journal of Management Information Systems* (22:1), pp. 13–43.