MIS Quarterly

SPECIAL ISSUE: DIGITAL BUSINESS STRATEGY

DESIGN CAPITAL AND DESIGN MOVES: THE LOGIC OF DIGITAL BUSINESS STRATEGY[1]

C. Jason Woodard
School of Information Systems, Singapore Management University,
Singapore 178902 SINGAPORE {jwoodard@smu.edu.sg}

Narayan Ramasubbu
Katz Graduate School of Business, University of Pittsburgh,
Pittsburgh, PA 15260 U.S.A. {narayanr@pitt.edu}

F. Ted Tschang
Lee Kong Chian School of Business, Singapore Management University,
Singapore 178899 SINGAPORE {tedt@smu.edu.sg}

V. Sambamurthy
Eli Broad College of Business, Michigan State University,
East Lansing, MI 48824 U.S.A. {sambamurthy@bus.msu.edu}

As information technology becomes integral to the products and services in a growing range of industries, there has been a corresponding surge of interest in understanding how firms can effectively formulate and execute digital business strategies. This fusion of IT within the business environment gives rise to a strategic tension between investing in digital artifacts for long-term value creation and exploiting them for short-term value appropriation. Further, relentless innovation and competitive pressures dictate that firms continually adapt these artifacts to changing market and technological conditions, but sustained profitability requires scalable architectures that can serve a large customer base and stable interfaces that support integration across a diverse ecosystem of complementary offerings. The study of digital business strategy needs new concepts and methods to examine how these forces are managed in pursuit of competitive advantage. We conceptualize the logic of digital business strategy in terms of two constructs: design capital (i.e., the cumulative stock of designs owned or controlled by a firm) and design moves (i.e., the discrete strategic actions that enlarge, reduce, or modify a firm's stock of designs). We also identify two salient dimensions of design capital, namely, option value and technical debt. Using embedded case studies of four firms, we develop a rich conceptual model and testable propositions to lay out a design-based logic of digital business strategy. This logic highlights the interplay between design moves and design capital in the context of digital business strategy and contributes to a growing body of insights that link the design of digital artifacts to competitive strategy and firm-level performance.

Keywords: Design capital, design moves, digital options, technical debt, IT architecture

[1] Anandhi Bharadwaj, Omar A. El Sawy, Paul A. Pavlou, and N. Venkatraman served as the senior editors for this special issue and were responsible for accepting this paper.

Introduction

Advances in the functionality of information technologies and transformations in the way they are being fused with products, services, and business processes are challenging conventional wisdom about the design and execution of competitive strategy. Three distinct types of logic have guided the prevailing managerial insights. The logic of *positioning* argues that managers should choose a profitable position in their industry and execute their firm's competitive strategy through cost leadership, differentiation, or market segmentation (Porter 1980). Within a chosen position, this logic recommends that firms align their activities and value chains for superior execution and delivery of a compelling value proposition to customers (Porter 2001). The classic conceptual arguments for the strategic role of IT in activities such as pricing (Beath and Ives 1986) and customer relationship management (Ives and Learmonth 1984; Porter and Millar 1985) reflect the logic of positioning. In contrast, the logic of *leverage* argues that firms can sustain their competitive advantage through the possession of rare, valuable, and inimitable resources and capabilities (Barney 1991); managers should therefore direct their attention toward resource-picking and capability-building processes as mechanisms for executing an effective competitive strategy (Makadok 2001). Information systems researchers have examined the complementarities between IT and business capabilities and resources (Melville et al. 2004) to establish mechanisms by which investments in IT could catalyze the development of business capabilities and measurably impact firm performance (e.g., Banker et al. 2006; Mithas et al. 2011; Ray et al. 2005). Finally, the logic of *opportunism* argues that managers must devote their attention to continuous innovation and competitive maneuvering in order to achieve sustained profitability (D'Aveni et al. 2010). According to this logic, competitive advantages are fleeting; therefore, successful firms must play the role of arbitrageurs, detecting windows of opportunity and executing competitive actions to seize them (Helfat and Raubitschek 2000). The role of IT in facilitating these competitive actions has been examined in prior conceptual developments (e.g., Piccoli and Ives 2005; Sambamurthy et al. 2003) and recent empirical studies (e.g., Lu and Ramamurthy 2011; Pavlou and El Sawy 2010).

While the need to align IT and business strategy has been a dominant theme in this work, El Sawy (2003) makes the case for a new perspective, which he terms "the fusion view of IS," that sees information systems as embedded in, and integral to, the product and service offerings of the firm. The emergence of new business models (Osterwalder and Pigneur 2010) and the transformation of entire industries by IT (Dhar and 2007) are prompting calls for a new logic of competitive strategy that recognizes the fused nature of IT and its central role in product development and service delivery. Observations of the competitive conduct of prominent technology firms such as Google, Apple, and Microsoft (e.g., Cusumano 2010)—as well as firms in industries that are undergoing strategic transformations such as financial services, hospitality, and entertainment—suggest that additional insights are needed to explain and predict strategic behavior in fused environments.

In this context, we define a digital business strategy as a pattern of deliberate competitive actions undertaken by a firm as it competes by offering digitally enabled products or services. Although the logic of positioning, leverage, and opportunism continue to provide a robust umbrella for generating insights about competitive strategy, recent conceptual frameworks, such as the digital ecodynamics perspective (El Sawy et al. 2010) and the complex adaptive business systems perspective (Tanriverdi et al. 2010), have begun to draw attention to the ways in which the logic of digital business strategy might be distinctive. In the same spirit, Yoo et al. (2010) call for a deeper examination of the logic of digital business strategy when they state:

> IS scholars need to question and complement their received models of aligning IT to business strategy, identifying core IT resources, and managing IT as a standardized commodity.…We need new strategic frameworks that are aimed at deliberately harnessing the unique capabilities of digital technology that are embedded into products to gain competitive advantage (p. 730).

They argue that digital innovation—the process of leveraging digital artifacts to transform existing physical products or create new ones—offers a powerful lens for developing such frameworks. The concept of digital innovation draws attention to the ways in which firms recombine, reconfigure, or design new digital artifacts in response to competitors' actions or windows of market opportunity.

Yoo et al. highlight the special importance of layered modular architectures in shaping digital innovation. These architectures coevolve with the governance choices of their stakeholders and are influenced by forces of technological and market turbulence beyond their control (Tiwana et al. 2010). On one hand, they must remain flexible and allow rapid adaptation to changes in technology and consumer preferences (El Sawy et al. 2010; Tanriverdi et al. 2010). On the other hand, they must achieve sufficient scale and stability to allow firms to extract economic rents from a large customer base over an extended period of time (Adner and Kapoor 2010; Boudreau 2010; West 2003). Moreover, the success of architectures that are championed by a single firm such as Apple or Google, or a small number of firms in partnership

such as Microsoft and Intel, often depends on the voluntary participation of other firms to create a vibrant ecosystem of complementary and competing offerings (Iansiti and Levien 2004). These firms must strike a balance between investing in an architecture for long-term value creation and exploiting it for short-term value appropriation.

Recent anecdotes about the success and failure of digital products and services testify to the strategic salience of this dilemma. For example, the initial success and eventual downfall of Myspace have been attributed in part to its early choice of an easy-to-use but "simplistic" software development platform, which limited its ability to sustain growth in the face of competitive pressure from Facebook (Gillette 2011). In the mobile device industry, architectural bottlenecks in the incumbent platforms of RIM (BlackBerry) and Nokia (Symbian) hampered their ability to compete with newer entrants such as Apple's iOS and Google's Android (ben-Aaron 2011, Blandford 2011; Mace 2010). In an extreme case, HP's TouchPad tablet was withdrawn from the market after only seven weeks. Members of the development team conceded that the product's core software, WebOS, suffered from architectural flaws that may have doomed the effort from the start (Chen 2012).

The goal of our research is to expand the emerging theoretical understanding about the way firms formulate and execute digital business strategies. Specifically, we study the design-based competitive actions through which firms develop new digital artifacts, transform their digital architectures over time, and influence their competitive environments. We draw upon the rich and growing literature on strategic product and system design (Baldwin and Clark 2000; Garud et al. 2003; LaMantia et al. 2008) and the dynamics of competitive actions (Smith et al. 2001), as well as case studies of four digital businesses. Our emergent theory connects the design decisions of firms to their role in influencing higher-level strategic capabilities. In the next section, we present our conceptual model and key constructs. After describing our empirical methodology we present our case data. We proceed to synthesize our case findings into a set of testable propositions, and then discuss the implications of our approach for developing a research program on digital business strategy.

Conceptual Development

The Locus of Digital Business Strategy: Design Capital and Design Moves

We conceptualize the logic of digital business strategy in terms of two key constructs: *design capital* and *design moves*. We define design capital as the cumulative stock of designs owned or controlled by a firm. In the context of digital business strategy, the most important elements of a firm's design capital are typically designs for digital artifacts, such as software components and their associated interfaces and data structures. These designs are sometimes collectively called the firm's digital architecture.[2] While some of these designs might have a direct financial value (e.g., a patented algorithm or a copyrighted user interface), the strategic value of a firm's design capital lies mainly in the fact that it enables the firm to innovate through new and improved designs. Design moves are discrete strategic actions that enlarge, reduce, or modify a firm's stock of designs—for example, developing a new product or service, improving an existing component in a layered modular architecture, or reconfiguring the architecture itself (Henderson and Clark 1990). The state of a firm's design capital both enables and constrains the design moves available to the firm at a given time.

Design Capital as a Digital Options Platform

Like other kinds of capital stocks (e.g., property, plant, and equipment), design capital is an economic factor of production. Unlike physical goods, designs are information goods and thus intrinsically non-rival (i.e., a firm can license or give away its designs without losing them). Moreover, the value of design capital may be highly firm-specific, since a particular set of designs might require complementary assets or capabilities to monetize them successfully (Teece 1986). Hence, design capital can be a source of competitive advantage—or disadvantage—similar to other firm-specific resources. In addition to designs for customer-facing system components (e.g., user interfaces) and architectural design elements (e.g., programming interfaces or design rules), design capital encompasses internal systems and processes that enable business capabilities (e.g., the ability to stream content, manage identity information, or transact payments securely).

Designs for digital artifacts can be highly complex, making it difficult to characterize a firm's design capital in a parsimonious way. Building on the prior literature on modular designs (Baldwin and Clark 2000) and an emerging literature in software engineering (Brown et al. 2010), we focus on two key dimensions that directly enable or constrain the competitive actions taken by firms in executing a digital business strategy*: option value* and *technical debt*.

[2]Our definition of design capital extends prior conceptualizations such as digitized process capital and knowledge capital (Sambamurthy et al. 2003), and embraces the concept of a layered modular architecture, which is central to digital innovation activities that fuse various components of a digital product or service offering such as content, network, devices, operating systems, user interfaces, and data (Yoo et al. 2010).

Baldwin and Clark (2006) argue that "new designs are fundamentally *options* with associated economic *option value*" (p. 181). The option value of a design reflects both the value of the products or services in which it is directly realized (which may be uncertain during the design process) and the value of the alternative designs that it makes possible (which might include variations or subsequent improvements on the initial design). Option value is closely related to the concept of generativity, defined by Zittrain (2006) as "a technology's overall capacity to produce unprompted change, driven by large, varied, and uncoordinated audiences" (p. 1980). Yoo et al. (2010) propose that "generativity in a layered modular architecture is accomplished through loose couplings across layers whereby innovations can spring up independently at any layer, leading to cascading effects on other layers" (p. 728). At the firm level, option value is a measure of the breadth of opportunities afforded by the firm's design capital. Option value is enhanced by architectures that enable designers to combine components within or across layers, cultivate or attract external partners (e.g., by providing application programming interfaces or software development kits), and launch innovative digital offerings.

Technical debt refers to the expected cost or effort entailed in exercising the options embedded in a firm's design capital (Baldwin and MacCormack 2011). Designers accumulate technical debt as systems evolve and create obligations that must be "repaid" in order to make changes to a system (Brown et al. 2010; Cunningham 1992). Such obligations could be associated with technical redesign, component upgrading, or wholesale replacement of an architecture or layer to implement a desired functionality. Software engineering research proposes that technical debt is a natural by-product of the design process and can be modeled using observed design decisions taken by the system designers (Sullivan et al. 1999; Sullivan et al. 2001). Technical debt is incurred for many reasons; while sometimes reckless or inadvertent, it may be prudent and deliberate (Fowler 2009). In particular, shortages of resources, time, or talent frequently lead designers intentionally to optimize for short-term goals at the expense of making a product or system easy to maintain and evolve. According to Brown et al. (2010, p. 47), "Like financial debt, sometimes technical debt can be necessary. One can continue paying interest" in the form of increased costs of code maintenance and development, "or pay down the principal by re-architecting and refactoring to reduce future interest payments." Recent software engineering studies have begun to quantify these costs empirically (e.g., Guo and Seaman 2011; Nugroho et al. 2011).

Baldwin and MacCormack (2011) argue that option value and technical debt are akin to financial assets and liabilities, and define the net present value of a modular architecture as the difference between the two. While a mathematical analysis of these relationships is beyond the scope of our work, we propose the analogous idea that the value of a firm's design capital is enhanced in the presence of high option value and low technical debt, and diminished by the opposite. These two dimensions of design capital thus provide a way to reason about the ability of a digital business to exploit the opportunities available to it.

The Design Capital Map

Figure 1 represents our two-dimensional conceptualization of design capital as a matrix of four different states. This matrix, or *design capital map*, provides an intuitive way to indicate the state of a firm's design capital at a point in time, and to illustrate the effects of design moves as transitions from one state to another.

A typical new product life cycle begins in quadrant I, with a low level of technical debt, but few options. In this state, firms are constrained by the technical potential of their existing designs, but exploiting this potential is relatively easy due to the absence of "legacy" code or compatibility requirements. In this *option-constrained* state, the limiting factor in enhancing the value of the firm's design capital is the lack of design options embedded in the current architecture. This could be remedied by continued investments in product development with the aim of expanding the firm's portfolio of design options. Conversely, quadrant III represents design capital with high technical debt and high option value. In this state, firms have a rich stock of designs with high potential for future development. However, their ability to exercise the options embedded in these designs is limited by the cost of retiring the technical debt that is attached to them. In such a *debt-constrained* state, significant investments may be needed to clean up a code base or address incompatibilities among existing modules before new development can proceed. Debt-constrained firms are often inhibited in their cycle time for new product releases or the ability to rapidly add functionality in response to market demand.

Quadrant II describes a state of *low-quality* design capital characterized by low option value and high technical debt. In this state, firms suffer from the worst of both worlds because their designs afford few options to exploit market opportunities, and even those options are of limited net value due to the high cost of exercising them. Finally, quadrant IV represents a state of *high-quality* design capital characterized by high option value and low technical debt. Firms in this state enjoy the opportunity to seize a wide range of market oppor-

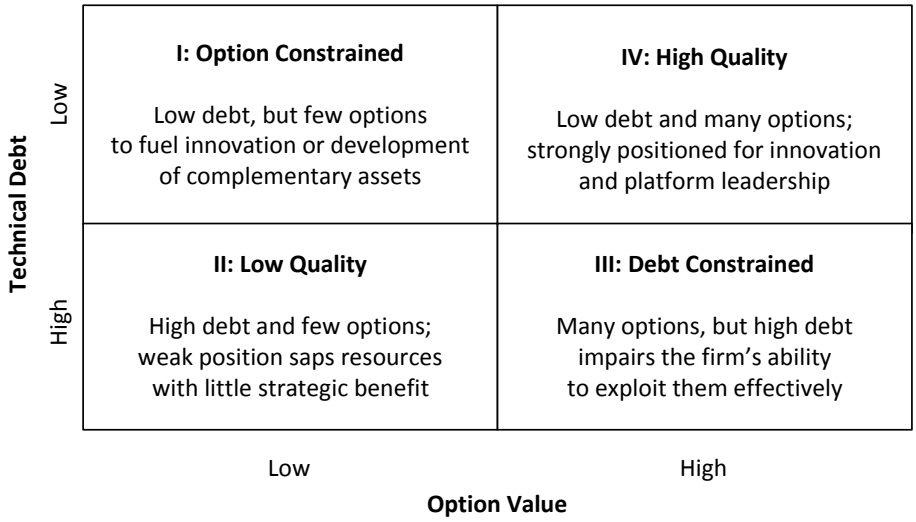

Figure 1. Design Capital Map

tunities and respond to their competitors' actions with speed and scale. These firms are typically well positioned to influence their broader ecosystems by using architectural control as a source of competitive advantage (Woodard 2008).

Design Moves and Their Dual Relationship to Design Capital

Design moves are discrete strategic actions that change the structure or function of a digital artifact (product or system). "Discrete" means that each action can be identified separately from others and arranged in a temporal sequence or nested hierarchically as part of a larger design move. "Strategic" means that the actions are taken with the intent of obtaining competitive advantage. We focus on actions taken by firms, although one could also consider design moves by individuals, governments, standards organizations, open-source communities, or other agents that engage in strategic design. The iterated application of design moves drives the evolution of individual designs and changes the stock of designs owned or controlled by a firm. Thus, design capital can be viewed as the cumulative result of design moves enacted over time.

Our use of the design move as a unit of analysis is informed by the work of Pentland (1992), who developed the concept of an organizing move to explain the behavior of technical support specialists in responding to customer calls. His work built on Goffman's (1981) use of moves to analyze discourse in face-to-face interactions. These studies grappled with the need to reason about actions that are deeply embedded in a situational context whose boundaries are hard to define in advance, and whose effects both enable and constrain subsequent actions. We contend that these conditions apply equally to the design of complex artifacts, which are core to firms that engage in digital business strategy.[3]

Just as characterizing design capital presents difficult theoretical challenges, there could be many ways to describe and classify design moves. For conceptual parsimony, we characterize design moves based on their effects on a firm's design capital, using the same two dimensions as the design capital map (Figure 1). A design move can be represented as a vector on the map, indicating the extent to which the move increases or decreases the option value of a firm's designs and increases or decreases the firm's technical debt.

[3]The concept of a move was also invoked by Donald Schön (1983) in his seminal study on reflective practice.

> According to Schön, designing proceeds as "a reflective conversation with the situation," an interactive process based on posing a problem frame by frame and exploring its implications in "moves" that investigate the arising solution possibilities. A designer, he argued, is faced with a situation of complexity. "Because of this complexity, the designer's moves tend, happily or unhappily, to produce consequences other than those intended. When this happens, the designer may take into account of the unintended changes he has made in the situation by forming new appreciations and understandings and by making new moves" (Cross 2011, p. 23).

To increase option value, a design move must either create new design options or increase the value of existing options. We define a design option as the right but not the obligation to make a design move in the future. Modularity multiplies design options because it allows multiple design experiments to be executed in parallel, each of which may be evaluated independently (Baldwin and Clark 2000). Design moves that change the architecture of a modular system are of special strategic significance because they can reshape both short-term opportunities for capturing economic value and the system's long-term path of design evolution (Baldwin and Clark 1997; Garud and Kumaraswamy 1995; Morris and Ferguson 1993).

Although it may seem counterintuitive that a firm would voluntarily reduce the option value of its design capital, product and system designers frequently encounter costs associated with keeping one's options open. For example, consider the additional effort involved in designing a graphical user interface to support multiple languages. At minimum, text strings need to be externalized (i.e., separated from the code); full international support requires a way to handle both multi-byte characters (e.g., for Chinese) and bi-directional text (e.g., for Arabic). One could imagine an ambitious programmer designing these features into a new product from the start, but if it turned out that most of the product's initial sales were in regions where externalized strings were sufficient, it might save time and confer competitive advantage (e.g., through speed to market) to forgo multi-byte and bi-directional support in subsequent versions.

The same example can be used to illustrate the relationship between design moves and technical debt. Consider, as an alternative to dropping full international language support, the possibility of simply "hard-coding" English icons and menus for a new feature that is slated to be released in an upcoming version of the product targeted exclusively at the U.S. market. Making this design move would not require abandoning the option to support additional languages in the future (since the existing internationalization code would not be removed), but it would raise the cost of doing so because additional effort would be required to undo the hard-coding and reimplement the icons and menus properly. This contingent cost is the technical debt that would be created by the move. The additional effort to undo the hard-coding (perhaps negligible, but larger if the original programmer has left the firm) is analogous to interest on the debt. Just as firms and consumers rationally take on financial debt when they face short-term constraints or opportunities to leverage their investments, designers face situations in which design moves that increase technical debt are perfectly appropriate. Eventually, however, the debt must be retired, or design evolution will grind to a halt. This can be achieved either through debt-reducing design moves or moves that reduce option value by abandoning options to which debt is attached.

The Dynamics of Design Moves

As a given design move increases or decreases option value and/or technical debt, it may shift a firm's design capital from one to another of the states shown in Figure 1. For example, consider a firm with low-quality design capital (quadrant II). If the firm's designers make debt-reducing design moves, the firm might transition upward to the option-constrained state (quadrant I). Alternatively, an option-creating move would shift the firm's position to the right, toward the debt-constrained state (quadrant III). A complex design move that simultaneously reduces debt and creates new options would help the firm move toward the high-quality state (quadrant IV).

Our conceptual model proposes that digital businesses enact design moves with the goal of managing the levels of option value and technical debt associated with the firm's design capital. At the same time, consistent with the conceptualizations of moves by Pentland and by Goffman, a firm's choice of design moves is typically both enabled and constrained by the prevailing characteristics of its design capital, as well as by the firm's overall strategy and the resources available to the designers. The remainder of the paper explores this duality between design moves and design capital through case studies of four digital businesses.

Research Methodology

Data

We conducted an embedded multiple-case study (Yin 2009) of four firms operating in different industries: enterprise software, wireless test and measurement equipment, mobile applications, and communication services. We gained access to these firms as a result of prior research engagements, and observed their evolution over a multi-year period between 2002 and 2010. We visited each firm for periods of two weeks at a time over an observation period of two to five years, during which interviews and participant observations were conducted with the executive leadership, project and product managers, and engineering staff. Follow-up interviews were conducted in the subsequent years. In each case, we collected quantitative, observational, and interview data on design decisions relating to applications, products, platforms, and services; the outcomes that resulted from those decisions; and a variety of context-specific factors such as the

influence of prior business outcomes, organization and project level conditions, architectural dependencies and affordances, and broader changes in the industry environment. For the enterprise software and test equipment firms, we also directly observed multiple versions of software source code together with associated project management and product release data.

Case Analysis

Our analysis of each case site began with the identification of the design moves involved. Then, using the design move as a unit of analysis, we examined how the prevailing design capital and other contextual factors influenced the sequence of design moves over time, as well as the effects of each design move on the firm's design capital. Finally, we conducted cross-case analyses to develop propositions about the conceptualized dualities between design capital and design moves.

Tracing Design Moves. We traced the design moves by identifying and cataloging the design changes in the products we studied at our case firms, and organizing these changes into sequences based on temporal and causal precedence. A design move can thus be viewed as a logical grouping of sequential design changes.[4] We established temporal precedence using product release data, source code version data, and the descriptions of case informants. To establish causal linkages between design changes, we paid careful attention to whether a design change provided a technological affordance or an architectural quality on which a subsequent design change depended. Establishing causal precedence in this way helped us to empirically trace the dual relationship between design moves and design capital by clearly establishing both the state of the firm's design capital at the beginning of the move and the impact of the move in terms of a transition to another state. Further, the temporal and causal sequencing of moves helped us to distinguish the "optional" nature of certain moves from simple path dependence (Adner and Levinthal 2004). A move was viewed as the exercise of a design option if it was not a necessary consequence of any prior move.

To characterize the state of design capital before and after each move, we coded the relative levels of option value and technical debt at these points in time, iterating between first-order analysis—giving voice in the interpretation of case events to the people who actually experienced them (Van Maanen 1988)—and second-order analysis of design changes observed using product- and system-level data. We did not employ quantitative measures of option value or technical debt; developing and applying such measures remains an open area for future work.

Understanding the Strategic Context. After plotting the sequence of design moves within each case, we systematically examined their strategic intent, actual outcomes, and the environmental and organizational considerations that influenced them. Multiple authors independently coded the strategic context associated with each design move using the case narratives we developed jointly, and resolved discrepancies in the coding during two separate peer-review sessions. In two cases, we had to completely rely on first-order interpretive data for our coding, whereas, in the other cases, we were able to corroborate the interpretive data with detailed quantitative metrics drawn from project management and product release records.

Resource munificence—that is, the availability of critical resources needed to operate within an environment (Castrogiovanni 1991; Dess and Beard 1984; Staw and Szwajkowski 1975)—emerged early in our analysis as a salient aspect of the strategic context for a firm's design moves. Designers working under low munificence (resource scarcity) typically faced long working hours and challenging expectations, while those in situations of high munificence (resource abundance) typically enjoyed more organizational slack. Other strategic contingencies emerged later (e.g., the level of technical capability in the organization responsible for a design move). When this occurred, we retraced the relevant design moves and coded the new contingency variable.

Developing Propositions. In the final phase of our analysis, we formally conceptualized an emergent theory by conducting a cross-case analysis on the design moves to develop testable propositions. Following a standard process of theory development using case data (Eisenhardt and Graebner 2007), we adopted an iterative approach, repeatedly comparing our emergent theory with the case evidence and prior theory. Eventually, our interpretations converged to a common conceptualization of design moves and their relationship to option value and technical debt, which in turn led to the propositions that are presented in the paper.

Reliability and Validity

We took several measures to ensure the reliability of our case data, the validity of our empirical constructs, and the external and internal validity of our analysis. We sought to improve reliability by (1) organizing case records for each firm in the same way, (2) using multiple observers to take notes during

[4]Design changes tend to follow patterns. Baldwin and Clark (2000) identified six patterns that occur in the design of modular systems. In the Appendix we discuss the relationship between design moves and these patterns, which they formalize as modular operators.

conversations with case informants, (3) reconciling any discrepancies through discussions among observers and/or follow-up clarification by informants, and (4) conducting periodic peer reviews of the interpretations and conclusions that emerged from our analysis. Construct validity was enhanced by using multiple sources of evidence gathered from different informants, and by establishing a chain of evidence from both the first-order interpretive data and the second-order archival data. Although we selected our cases by convenience, the sample and our case analysis procedures do not pose a serious threat to external or internal validity. The cases were similar in that they all concerned digital businesses—that is, digital artifacts were core to each firm's business operations and ecosystem relationships. Our sample also exhibits variation in the types of design moves that were enacted and the antecedents we studied, which facilitated the exploration of differences in the dynamics of formulating and executing digital business strategies across the four cases.

Case Descriptions and Findings

Softsys

Softsys is a multinational software product development and services company established in 1989 with worldwide revenues of about US$50 million in 2011-12. The company is part of a US$800 million business conglomerate with diversified operations in the manufacturing and alternative energy sectors. Table 1 summarizes the set of design moves enacted by Softsys between 1989 and 2010, along with the conditions under which these moves were made and their impact on the firm's design capital. In addition, Figure 2(a) illustrates these moves visually using a design capital map (see Figure 1).

The goal of the first set of design moves enacted between 1989 and 1993 (labeled 1A, 1B, and 1C) was to rapidly transform a single-client manufacturing resource planning (MRP) application into a mass-customized enterprise software suite with a sustainable installed base beyond the sister companies within the conglomerate. Since a mass-customized software product caters to the needs of heterogeneous clients, a key feature was the ability to configure the product to satisfy the needs of individual clients. Move 1A entailed developing a systematic configuration mechanism that pulled together scattered configuration information, and creating a configuration engine module—a design option that product developers could use in the future to help clients configure the MRP application easily. This move sought to enhance the option value of the firm's design capital, as shown in Figure 2(a). Move 1B built on the option created by 1A and entailed replacing the previous configuration engine with an enhanced version that helped designers to track and store clients' configuration information in a centrally controlled database. As Softsys clients configured and deployed the product, the company collected a large repository of system configuration data and business rules implemented by its clients in the manufacturing sector.

Move 1C leveraged the new configuration repository to help Softsys product implementation teams match end-user requirements with the configuration information found in the repository and substitute default system configuration settings with the closest matches found in the repository. The configuration settings suggested by this solution were often far from ideal due to poorly specified end-user requirements, leading to more expensive post-implementation system changes. Thus, move 1C increased the technical debt of systems implemented at client sites. However, the move was seen as a crucial step in reducing the initial cost of implementation at a time when the firm's available resources for product development were diminishing and managers were under intense pressure to expand the product's installed base. Collectively, this set of design moves gave the Softsys product implementation team the capability to configure the product easily and reduce implementation time and costs without the use of expensive third-party functional consultants. Moreover, Softsys was also able to offset most of its system configuration-related technical debt (i.e., the cost of post-implementation changes) through annual maintenance contracts with clients. Thus, the first set of Softsys design moves built high-quality design capital that the firm used to position itself as a low-cost enterprise software vendor and launch competitive actions such as disintermediating the role of functional consultants in the small and medium business segment of the enterprise software industry.

In the mid-1990s, Softsys further leveraged its design capital to mine "best practices" from its configuration repositories and offer consulting and system integration services for small and medium businesses looking to ride the enterprise resource planning (ERP) wave. As Softsys's system integration services grew rapidly, tensions began to build between the product development and services divisions of the firm. Service personnel facing heterogeneous customer needs wanted immediate, client-specific solutions whereas the product development team aimed for mass customized functionality with a structured product release schedule. Moreover, as the services team enrolled high-value customers who demanded immediate fixes, the product development team was constantly fire-fighting, leaving little room for well-designed and forward looking product functionality. Design moves 2A, 2B, and 2C were enacted in this environment with an aim to balance the tensions between the services and product development Softsys teams. These design moves created and developed "custom extensions," which were extensions to

Table 1. Softsys Design Moves

Design Move	Strategic Intent	Design Actions	State of Design Capital	Strategic Contingencies	Impact of Design Move
1A	Simplify product configuration, disintermediate functional consultants	Expose configuration settings, create configuration scripts, collate configuration information into a new configuration engine module	Option constrained	Resource abundance	Create options
1B		Create a configuration database and add new logging functionality to the configuration engine	High quality	Resource abundance	Create options
1C		Enable consultants to search configuration database to alter default configuration settings during implementation at the client site	High quality	Resource scarcity*	Increase debt
2A	Pursue dual low-cost product plus value-added services strategy	Expose function calls for modification by clients, create tool set to append user-written arguments to function calls	High quality	Resource scarcity*	Increase debt
2B		Implement customer exit points that allow clients to add their own source code to create new functionality	High quality	Resource scarcity*	Increase debt
2C		Develop a new tool set to help customers manage their exit-point portfolios	Debt constrained	Resource scarcity*	Increase debt
3A	Achieve flexibility without losing efficiency; build business partner ecosystem	Remove all customer-written functionality and port product's source code to a component-based architecture	Debt constrained	Resource abundance	Abandon options Reduce debt Create options
3B		Create an application composer module to facilitate inclusion of customer-written components in the component architecture	High quality	Resource abundance	Create options
3C		Include ability to add third-party components via application composer	High quality	Resource scarcity**	Abandon options

*With high ability to transfer technical debt **With low ability to transfer technical debt

default system modules that helped customers modify the default system behavior (e.g., data processing, report definitions, and results printing). Design move 2A entailed exposing certain function calls to end users and creating a tool set to append end-user written arguments to the exposed calls. Move 2B facilitated the integration of customer-written code with default business logic in order to customize the system behavior, and move 2C involved the development of a new set of tools for customers' use. These tools allowed users to create and manage custom extensions themselves.

The custom extension design option conceived by move 2A helped Softsys services teams address customer-specific needs quickly in the short term without waiting for the standard mass-market releases from the product development team. This helped the business pursue a dual low-cost product plus value-added services strategy and fend off competition from both incumbent product vendors such as SAP and Oracle and system integration service companies such as Accenture and Capgemini, all of whom were aggressively targeting the small and medium business enterprise software sector. However, as a result of moves 2B and 2C, designers also accumulated higher levels of technical debt because of the explosive growth in the number of custom extension points and the inability of the designers to govern and control the quality of customer-written software code. Bad customer-written code often caused undesirable system hangs and crashes that were hard to diagnose and rectify. Thus, as shown in Figure 2(a), Softsys had transitioned into a debt-constrained state.

By 2000, the balancing act between the firm's product development and services divisions became critically vulnerable. As the number of custom extensions grew from a handful to thousands with the rapid growth in the product's installed base, it became impossible for the product development team to track, test, and ensure the compatibility of the large set of custom extensions against new product releases. Thus, the locally optimized customer-specific software code built by Softsys system integrators using the custom extensions could rarely be preserved during regular product upgrades. This situation started a vicious cycle of building and discarding custom extensions with little sustained value to the installed base. Further, Microsoft announced new middleware components that were incompatible with the custom extensions designed by Softsys. Hence, Softsys eventually chose to abandon the custom extension option and move toward Microsoft's new middleware offerings.

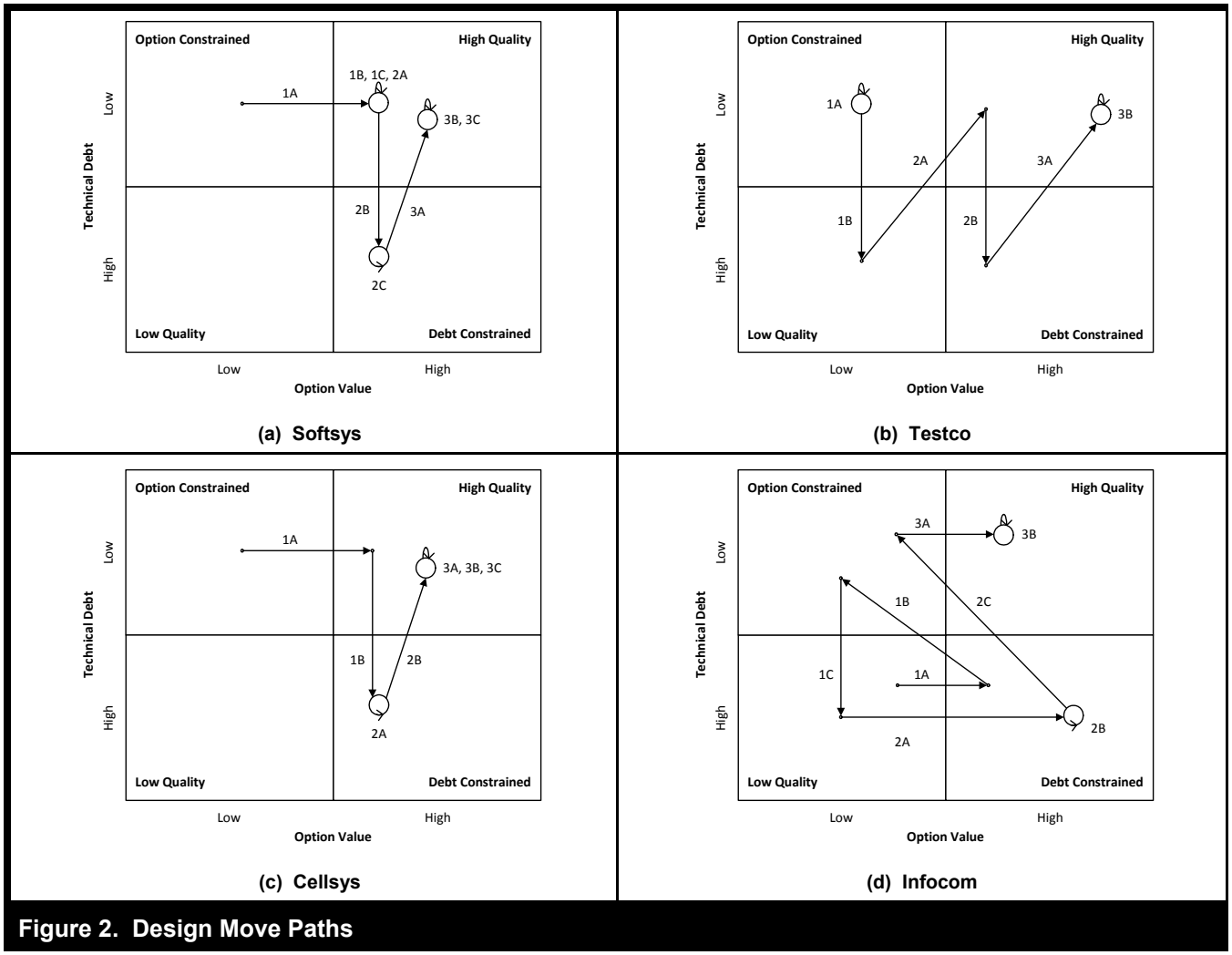

Figure 2. Design Move Paths

The abandonment of its custom extension option was the starting point for the third set of Softsys design moves (3A, 3B, and 3C), which focused on developing a product development architecture that was more robust to the cyclical standardization and customization needs triggered by the firm's dual product plus services strategy. Move 3A involved separating the default system modules from customer-written code and moving the original system modules to a new architecture that supported component-based system development. Move 3B entailed the development and integration of a new application composer module that facilitated the inclusion of customer-written components. Move 3C helped designers to easily identify and swap customer-written components and easily trace their dependencies on the original system components. These design moves established a robust component-based architecture platform for product development—an option that could be used to achieve flexibility in product customizations without compromising product development efficiency or the ability to achieve benefits from mass customization. As shown in Figure 2(a), these design moves reduced technical debt and moved the firm into a high-quality design capital state.

The modular component-based architecture enabled Softsys services teams to integrate third-party software easily and at the same time helped the product development teams to test and verify cross-platform compatibility more efficiently. Softsys leveraged the design options created by the component architecture move to develop the capability to participate in "consortium bids," that is, to partner with multiple third-party vendors in order to jointly service larger well-established firms that typically spend more on IT. Our observation period for the Softsys case ended in 2010 with the completion of move 3C, which entailed abandoning some of

the older component-based architecture design options in pursuit of newer technologies and processes such as service-oriented architecture and model-driven development. By 2010, Softsys had successfully built a sustainable partner ecosystem consisting of key players such as IBM Global Services, and was well positioned for the next phase of growth with an installed base of more than 1,000 clients, including several Fortune 100 companies such as Boeing.

Testco

Testco is a multinational firm with more than 60 years of experience in developing test and measurement equipment for electrical circuits, wireless networks, and industrial equipment. It is a member of a diversified business group with operations in medical technologies and industrial test and measurement equipment, with product sales of about US$16 billion in 2011. Our case study traces the evolution of a strategic hardware and software product developed by Testco's wireless test and measurement unit beginning in 1999. The design moves enacted by the firm between 1999 and 2004, the conditions under which these moves were made, and their impact on Testco's design capital are summarized in Table 2 and Figure 2(b).

In 1999, Testco had low levels of both option value and technical debt. In this option-constrained state, Testco's design moves 1A and 1B depict how the firm attempted to exploit its recently acquired assets in networking solutions to explore new business opportunities in a neighboring technological landscape, namely, short-range wireless communication. Move 1A, enacted in early 1999, helped Testco to reuse a key signal processing module of an existing wireless network connectivity product, and in turn initiate a new product line in short-range wireless communication test and measurement equipment. Testco designers did so by exposing the interfaces of the signal processing module in the existing product, enabling potential reuse of the module in a different context—an option that aided rapid product development. Testco's next design move, 1B, integrated the reused signal processing module with a special purpose best-of-breed product architecture. The best-of-breed architecture created by move 1B helped Testco designers preserve the reused signal processing module written in the C programming language, but pursue new product development using the Microsoft Visual C++ and Java programming languages. However, the interfaces between the components written in different programming languages had to be constantly modified as the new languages rapidly evolved and their standard libraries kept changing. Thus, Testco accumulated technical debt due to the best-of-breed architecture and could not fully exploit the options embedded in the reused signal processing module.

The first set of design moves (1A and 1B) enabled the Testco product development team to pursue rapid new product development despite being new to the domain. Moreover designers were able to manage the uncertainty that arose from intense competition between the Microsoft and Java technology platforms. In Testco's industry, speed is of the essence as test and measurement products need to be several cycles ahead of the downstream product development projects in which they are used. Thus, the options created by Testco's first set of design moves were instrumental in aiding the firm to launch its products quickly in an uncertain and immature technological environment. However, as shown in Figure 2(b), the net result of the first set of moves (1A and 1B) was that Testco had migrated to a state of low-quality design capital because of the technical debt incurred in pursuing its best-of-breed architecture.

By early 2000, Microsoft announced extensive support for the popular hardware tools used for wireless testing and measurement applications, and provided a clear roadmap for the .NET platform and C# programming language, which positioned Microsoft technologies to dominate the test and measurement market. At the same time, Testco's wireless test and measurement products had gained traction in the market. In this environment, Testco designers enacted design move 2A by completely rewriting the signal processing module using Microsoft technologies and creating a new signal analyzer component. This move helped designers reduce the accumulated technical debt due to the earlier best-of-breed design and aided the development of an architecture that facilitated self-contained "sandboxes" (isolated environments to experiment with new technologies and wireless standards). As a result, Testco was able to position its test and measurement products as standards-neutral in an environment where short-range wireless standards such as Bluetooth were still emerging and key players such as Ericsson, Motorola, Nokia, Intel, and Microsoft had different competing conceptualizations of the market. As shown in Figure 2(b), this design move enhanced the quality of the firm's design capital. In its subsequent design move (2B), Testco sought to relax the rigid compatibility constraints of the new signal analyzer components. Further, its designers made the interfaces of the experimental sandboxes more accessible and allowed open-source software communities to create libraries for different versions of standards and easily map them to configurations of Testco signal analyzer components. Although contributions from open source communities helped the firm negotiate its product development trajectory in an uncertain standards environment, integrating these contributions was a challenge. To overcome

Table 2. Testco Design Moves

Design Move	Strategic Intent	Design Actions	State of Design Capital	Strategic Contingencies	Impact of Design Move
1A	Exploit existing technology to rapidly develop a new product	Expose interfaces to signal processing module of acquired product	Option constrained	Resource abundance	Create options
1B		Extract signal processing module and integrate it with a new best-of-breed system architecture	Option constrained	Resource scarcity	Increase debt
2A	Manage standards uncertainty, cultivate open-source ecosystem	Replace signal processing module with a completely rewritten signal analyzer module, add a new experiment workbench module	Low quality	Resource abundance*	Create options Reduce debt
2B		Expose interfaces to experiment workbench and signal analyzer modules, add mapping tools to allow customer-written configurations to be plugged in	High quality	Resource scarcity**	Abandon options
3A	Exploit learning to move to adjacent technology landscapes	Reengineer dependencies between workbench and signal analyzer modules; move maps, experiment workbench, and signal analyzer modules to a new product family architecture	Debt constrained	Resource abundance	Create options Reduce debt
3B	Support dominant standards to sustain ecosystem	Add a digital rights management module to the product family architecture	High quality	Resource scarcity**	Abandon options

*With high technical capability **With low ability to transfer technical debt

it, Testco engineers had to abandon certain design options in its newly created signal analyzer module that imposed rigid standardization and reliability constraints, and in the process inadvertently allowed poor-quality code to creep into the system. Thus, as Testco sought to cater to different open-source communities, it incurred higher technical debt and moved to a debt-constrained design capital state.

By 2002, Testco had garnered more than 60 percent of the short-range wireless test and measurement market. The third set of design moves (3A and 3B) were enacted to capitalize on this dominant position by further consolidating the product architecture. Specifically, these moves generalized the interfaces between the core components of the product such as the signal analyzer component and the experimental open-source packages, and added robust standards traceability mechanisms. By generalizing the interfaces in the product architecture through move 3A, Testco designers created options in the product architecture to support the development of a family of products based on the same underlying platform. The improved APIs offered by this new product family architecture helped Testco designers more efficiently perform quality checks on the open-source code contributed by the end-user community, and thereby avoid costly maintenance. Testco leveraged the options created by the product family architecture to maximize its scope economies by catering to different industry segments beyond the original short-range wireless sector, such as video and audio signal analyzers. As shown in Figure 2(b), this design move again enhanced the quality of Testco's design capital.

Our observations of Testco's product development trajectory culminated with the shelving of the specific short-range wireless test product we studied, but Testco successfully expanded the product family architecture built from the original platform. Move 3B, the final design move that we observed, helped product managers trace, control, and terminate product variants by adding a digital rights management module to every product in the family. By controlling the "rights" to add to the product family, Testco designers were able to strategically terminate variants that were perceived as undesirable to the overall product ecosystem. For example, when a dominant Bluetooth standard emerged in the market, Testco abandoned open-source product variants that violated the dominant standard. While the inclusion of rights management entailed abandoning options related to certain non-standard product variants, it enabled Testco to establish a rich platform ecosystem that allowed wide-ranging customization while retaining the ability to manage the ecosystem through effective control of undesired variation. As a net result, Testco's design capital continued to remain in a high-quality state.

Cellsys

Cellsys is a mobile application platform development firm, established in 2006, which focuses on operator- and platform-independent product development for mobile phones. In 2011, Cellsys served more than 16 million end-users and had a stable clientele of more than 10 leading mobile telecommunication operators such as Vodafone, France Telecom's Orange, and Hutchison's Three. Cellsys was also recognized as a technological innovator and had won several awards including the Red Herring Global 100. The design moves enacted by Cellsys between 2006 and 2010, the conditions under which these moves were made, and their impact on the firm's design capital are summarized in Table 3 and Figure 2(c).

As a startup company in 2006, Cellsys primarily focused on creating a portfolio of applications written in Objective-C for various devices on Apple's Mac OS X and iOS platforms, hoping to generate revenues from the growing user base for Apple's consumer devices. Move 1A extended the design of the firm's social media aggregator application by including additional modules to capture finer-grained details about user behavior in the various social media applications such as Facebook, Twitter, Orkut, and Yahoo! Messenger. The move created an option for Cellsys to mine user behavior on social media platforms and offer personalized recommendations, as well as mobile games and advertisements. However, Apple reviewers who control the distribution of applications through the iTunes App Store rejected Cellsys's first product, a social media application suite, built on the design option created by move 1A. Citing unauthorized features and API usage, Apple reviewers objected to the feature of Cellsys suite that facilitated automatic installation of user-generated content and applications on the iPhone without going through the iTunes App Store. That is, Apple reviewers enforced their control and objected to the "platform within a platform" scenario embedded in the Cellsys design.

Resilient to the rejection from the iTunes App Store, Cellsys designers pursued a platform-independent design (design move 1B) and positioned the product for distribution to a wide range of other mobile platforms, including Microsoft's Windows Mobile, Google's Android, Nokia's Symbian, and Sun's J2ME. In addition to this "multi-homing" strategy, the Cellsys social media application suite was redesigned to be distributed as a standalone installer through the official marketplaces of the mobile platforms as well as through independent application stores on the Internet. This design move helped Cellsys achieve a quick take-off, attracting more than 10 million users through a peer-to-peer distribution network that leveraged the digital word-of-mouth effects of its social applications. However, it increased the firm's technical debt because of the need to repeatedly change the installation package of the Cellsys application to address the various incompatibilities across the rapidly evolving mobile platforms, thereby dragging the firm into a debt-constrained state.

Despite being laden with high technical debt, Cellsys was spurred by the success of the multi-homing social media application and began to add applications targeted at mobile telecommunication operators to the Cellsys social media suite (move 2A). This option enabled Cellsys to pursue a product family strategy encompassing both retail end-users as well as the mobile operators' corporate clientele. Cellsys leveraged the design capital built on its multi-homing design option to develop a new value proposition for the operators: they could offer sticky services to end users without worrying about the incompatibilities between the mobile devices that these users might carry. Through this value proposition, in 2008 Cellsys was able to bring on board its platform more than 10 leading mobile operators in Europe, the Middle East, South East Asia, and Australia. As the product family strategy of Cellsys matured, designers split the features of the Cellsys platform into two distinct products, one catering to retail end users and the other to mobile operator corporate customers (move 2B)—a design which reduced the technical debt of the multi-homing architecture. This helped Cellsys designers shed debt-prone integration mechanisms between the two systems, scale their product development efforts quickly, and cater to the needs of the two distinct user bases more efficiently. As a net result, the firm transitioned to a high-quality design capital state.

The third set of design moves (3A, 3B, and 3C) focused on creating options for enabling user-led innovation. Designers created an open API and an end-user accessible software development kit (move 3A). They created an option to allow third-party programmers to develop applications, such as social media games, for the Cellsys platform. The next design move (3B), created an iTunes-style application store and an option for the mobile telecommunication operators to launch their own platform-independent (but operator-specific) applications for end users. The third design move (3C) created a similar application store for retail end users. These design moves continued to reinforce the high quality of the firm's design capital and established a broad reach in the programmer communities associated with three major mobile platforms—Windows, Android, and Java. All of these communities were eager to build applications and distribute them to the company's large base of end users and network operators. The continued high quality of its design capital benefitted Cellsys in executing its digital business strategy through a slew of digital offerings, including home-grown and

Table 3. Cellsys Design Moves

Design Move	Strategic Intent	Design Actions	State of Design Capital	Strategic Contingencies	Impact of Design Move
1A	Multi-homing and platform independence	Integrate the iPhone social media aggregator app with a new system including meta-data collection and game center framework modules	Option constrained	Resource abundant	Create options
1B		Create variants of the system for Android, Symbian, Blackberry, Windows Mobile, and Java platforms	High quality	Resource scarcity*	Increase debt
2A	Pursue a product family strategy	Add an operator services module to the Cellsys product	Debt constrained	Resource abundance	Create options
2B		Create two distinct product designs for retail end users and telecommunication network operators	Debt constrained	Resource abundance	Reduce debt
3A	Enable user-led innovation in the retail and operator ecosystems	Expose the APIs of the system to end users, create a software development kit for end-user application development	High quality	Resource abundance	Create options
3B		Create an operator app store module for corporate customers	High quality	Resource abundance	Create options
3C		Create a marketplace module for retail end users	High quality	Resource abundance	Create options

*With high ability to transfer technical debt

partner-developed games as well as white-labeled products and services for a variety of telecommunication operators.

Infocom

Infocom is an integrated information and communication services provider with over US$2 billion in annual revenues from over two million end users of its mobile telecommunications, cable television (TV), and broadband Internet service offerings. We studied the evolution of the firm's core IT systems, which supported the three service divisions of the company from 2001 until 2010. The set of design moves enacted by Infocom during this period, along with the conditions under which these moves were made and their impact on the firm's design capital, are summarized in Table 4 and Figure 2(d).

Infocom came into existence in 2000 as a result of a three-way merger between firms that operated separately in the cable TV, residential broadband, and mobile telecommunication sectors. Immediately after the merger, the company turned its focus to integrating the back-end processes of the three separate service divisions without disrupting services for their existing customers, with the goal of gaining economies of scale in the administrative and operational functions of the company. Because the firm's IT personnel did not have the luxury of clean-slate integration, they had to constantly firefight, managing the day-to-day operation of the existing services while working to consolidate the firm's back-end processes and systems. Therefore, the firm's design capital was of poor quality. The first set of design moves (1A, 1B, and 1C) were enacted with the aim of achieving loose coupling between the various back-end IT systems, including the billing and account management applications of the three services divisions of the firm. The first design move (1A) exposed the interfaces of these back-end systems and specified the necessary adapters that could hook the applications to a common database schema. This created an option to pursue a loose coupling approach for integrating the back-end systems without disrupting services for existing customers. As a result of the expansion of option value, the firm transitioned to the debt-constrained state.

Once the adapters became operational, Infocom's IT personnel attempted to build a common customer relationship management (CRM) system for the firm on top of the established loosely coupled integration mechanism. Early studies exploring the feasibility of such a design revealed fatal incompatibilities between the billing applications of the three service divisions, which were a hurdle to building a common CRM platform. While the loose coupling approach enabled the coexistence of the different applications, it did not scale up to the performance needs of the planned CRM platform and required costly manual intervention in reconciling errors that arose during data integration. Hence, Infocom abandoned the common database schema built on the loose coupling design option and reverted back to three separate billing applications for the three service divisions (move 1B). This reduced the technical debt induced by the earlier loose-coupling

Table 4. Infocom Design Moves

Design Move	Strategic Intent	Design Actions	State of Design Capital	Strategic Contingencies	Impact of Design Move
1A	Integrate three major service lines (mobile, cable TV, broadband ISP)	Expose the billing application interfaces of the mobile, TV, and broadband ISP business units	Low quality	Resource scarcity	Create options
1B		Create and integrate a custom CRM module into the individual billing applications of the three business units	Debt constrained	Resource scarcity	Abandon options Reduce debt
1C		Create an account manager module to collate billing details for customer's mobile, TV, and ISP accounts	Option constrained	Resource scarcity	Increase debt
2A	Enable self-service for customers	Replace legacy billing applications with new billing modules	Low quality	Resource abundance*	Create options
2B		Replace legacy billing applications with new integrated CRM/billing system, move all customer accounts into the new system	Debt constrained	Resource abundance	Create options
2C		Replace account manager module with a new application, migrate all accounts to new CRM system using unique self-service IDs	Debt constrained	Resource scarcity	Abandon options
3A	Enable cross-selling and bundling of services	Expose integrated CRM/billing system interface to social media applications	Option constrained	Resource abundance	Create options
3B		Create and integrate a marketing, promotion, and loyalty module that can use data from social media platforms such as Facebook	High quality	Resource abundance	Increase debt

*With low technical capability

design, but also reduced the options available to designers to implement a common Infocom CRM platform. Thus, a net result of move 1B was to transition the firm's design capital to an option-constrained state.

After abandoning the loosely coupled design, Infocom's IT organization was under pressure to find alternate ways to move toward a common billing and CRM mechanism for the three service divisions. They responded by implementing a system that collected customer information (e.g., billing details) from the three separate billing applications and presented it to an end user through a common website portal for account management (move 1C). This design was far from ideal and imposed additional technical debt related to maintaining three different applications serving the same business functionality. Further, although Infocom customers could check their bills through a common web interface, it was still cumbersome for them to have three different accounts with separate bills for their services. This resulted in a surge in billing-related queries and account-management tickets for Infocom engineers, further reducing their capacity to work toward a common CRM platform. Thus, as shown in Figure 2(d), move 1C returned Infocom to a poor quality design-capital state with high technical debt and low option value.

Infocom's second set of design moves (2A, 2B, and 2C) were enacted to address the need for tighter integration between the various service divisions. Using a fresh infusion of capital by the firm's top management to support an initiative for tighter integration, designers replaced legacy billing applications with modern commercially available products (move 2A). These products had well-established mechanisms for facilitating cross-application integration. This made feasible the integrated CRM platform that was earlier attempted in vain. Infocom IT personnel, working with contracted systems integration professionals, then replaced the individual CRM application modules in the legacy billing systems with a new integrated CRM database (move 2B). This in itself created more design options for launching specific applications in the future. For example, to reconcile the differences in billing cycles of the three services for different customers (TV, broadband, and mobile services), Infocom launched a new account management module that offered a self-service mechanism for customers to log in and choose a design for an integrated bill. Thus, move 2A transitioned Infocom to a debt-constrained state and move 2B further expanded the firm's option value while it remained debt constrained overall.

The new account management module and its self-service feature, however, had an unexpected consequence of complicating the firm's customer rewards and loyalty programs. Although engineers had ported the older rewards and loyalty points into the integrated CRM database, they had not anticipated the variety of ways that customers would want to bundle their three service bills and the corresponding loyalty points. Some customers preferred to keep their bills and asso-

ciated loyalty points separate for certain billing cycles, but have them bundled for certain other billing cycles. The integrated CRM database and the new account management module could not automatically handle all of these customer-created loyalty point bundles, often resulting in erroneous printing of reward points tallies in the monthly bills sent to subscribers. Customer care agents and, in turn, engineers were burdened with a growing volume of requests for resolving problems with the custom bundling of loyalty points. Eventually, Infocom designers, on the advice of external system integration consultants, abandoned the account management module that allowed customers to create self-service billing bundles (move 2C) and replaced all self-created billing and loyalty point bundles by customers with standardized ones. Customers were issued standardized service bundles and account numbers, which they could use to electronically check their bills and service usage in real time. While the abandonment of some self-service account management mechanisms helped Infocom reduce maintenance costs, it also eradicated the customization options for end users. Thus, by the end of this set of design moves, the firm's design capital had once again become option constrained.

Infocom's final set of design moves began with one that exposed the APIs of the tightly integrated CRM and billing systems (move 3A). This created an option to develop marketing and loyalty modules that were accessible through end-user accounts in social media applications such as Facebook (move 3B). This move initially imposed technical debt on designers due to the immature and nonstandard interfaces of the various social media applications that were not compatible with each other. For example, the marketing and loyalty modules developed for Facebook were not readily portable to other social network platforms such as Orkut or LinkedIn. However, as the various social media platforms matured, Infocom engineers were able to leverage third-party social media integration tools to reduce the burden of integration. Thus, moves 3A and 3B helped Infocom to sustain its high-quality design capital, enabling marketing managers in the three service divisions to launch and promote new service bundles and leverage cross-selling opportunities across the three types of service offerings.

Summary

As seen in Figures 2(a) and 2(c), the patterns of design moves at Softsys and Cellsys are visually similar. They represent the design moves of firms operating in mass-customized, high-growth product environments, where strategies that entail high debt are possible because the debt can be repaid by using rents extracted from locked-in customers. Both firms achieved high-quality design capital fairly early in their product life cycles, but rapid growth and demands from heterogeneous customer bases led them to accumulate technical debt, dragging them into the debt-constrained zone. The firms were able to leverage their debt-taking strategy through entrepreneurial actions—launching systems integration services in the case of Softsys and operator-oriented services in the case of Cellsys—that helped them grow and generate revenue. Using their resources well, Softsys and Cellsys eventually paid down their debt through a series of moves that enabled them to achieve high-quality design capital.

Figure 2(b) illustrates Testco's design moves, which are those of a technically competent firm operating in an environment of high clockspeed and volatile standards. Testco's designers managed uncertainty in short-range wireless protocol and programming language standards by constantly pursuing a wide range of options. Some of the options were debt-inducing because the volatile standards rendered the original designs irrelevant, but Testco's designers were able to write off the debt by abandoning the debt-laden options and quickly reorienting product development using alternative designs. Every time Testco found itself in a low-quality or debt-constrained state, its engineers were quick to enact design moves that improved the quality of the firm's design capital. Eventually, they were able to develop a robust platform architecture that supported multiple product lines.

The spiraling pathway of Infocom, shown in Figure 2(d), illustrates the struggles of product designers inheriting legacy systems and constantly fire-fighting to satisfy the firm's business needs. Although Infocom pursued an aggressive business strategy to offer bundled TV, mobile, and broadband services to its customers, its engineers struggled with limited design options to integrate the legacy systems. Plagued by frequent shifts in resource availability and external dependence for know-how, the firm found itself in a vicious cycle and oscillated between the option-constrained and debt-constrained states. Eventually, a large infusion of resources at the corporate level helped Infocom forge long-term contracts with external vendors. The firm was able to achieve high-quality design capital that could robustly support its business strategy to grow by offering integrated information and communication services.

Design Capital and Design Moves: Toward a Design-Based Logic of Digital Business Strategy

In our conceptual development, we identified design capital and design moves as fundamental to the emerging logic of digital business strategy. Design capital was defined in terms

of two dimensions, option value and technical debt, with high-quality design capital being characterized by high option value and low technical debt. Our case studies suggest that high-quality design capital places firms in a superior position to execute competitive actions. Each of the four firms in our study gained significant advantages as the quality of their design capital improved. They were able to launch digitally enabled products and services with greater speed and scale, react more rapidly to market opportunities or competitive threats, and more effectively shape their business ecosystems. However, not all firms are successful in possessing high-quality design capital at all times. All four case firms were constrained by high debt or lack of options (or both) at various times during our observation period.

The role of design moves in our conceptual model is to provide a structured way of thinking about the actions taken by firms in pursuit of high-quality design capital. By classifying design moves according to their effects on a firm's design capital (increasing or decreasing option value and technical debt), we can plot a sequence of moves as a path through the two-dimensional space defined by our design capital map (Figure 1). The four paths shown in Figure 2 provide a compact summary of our case evidence. They also offer a plausible target for explanatory theorizing, in the sense that it might be desirable for a theory of digital business strategy to explain why these paths are observed instead of others (or predict a firm's next move given its path to date). While this may be a worthy goal, it is fraught with both conceptual and practical difficulties, including the need to account for the influence of corporate-level strategy and industry-specific trends. Instead, we adopt the more modest goal of explaining the observed transitions between design capital states, using only information about the current state of a firm's design capital and a minimal set of additional contingencies.[5]

As indicated in the earlier discussion of our research methodology, resource munificence proved to be an important contingency in every design capital state. We observed that the level of resource munificence (or organizational slack available to designers) often explained why firms in the same state made moves that took them in different "directions" (e.g., from option-constrained design capital either rightward to high quality or downward to low quality). Higher levels of resource munificence allowed firms to move toward higher-quality design capital or leverage their stock of designs more effectively in creating digital offerings. On the other hand, lower levels of resource munificence tended to constrain firms and compel them to either increase their technical debt or fail to exploit their options. We treated the level of resource munificence itself as exogenous, but noted that it was frequently cyclical: resources tended to be abundant at the beginning of a new project or when the firm was doing well, and scarce as product development deadlines approached or when the firm was under pressure to respond to a market opportunity or competitive threat.

Technical capability, or the ability to deploy needed technical skills in executing a design move, emerged in our analysis as an important contingency. Although technical capability is related to resource munificence, we observed instances in which organizations with abundant resources were still constrained by the level of technical capability available for a particular design move. Similarly, the ability to transfer technical debt emerged in our analysis as an important contingency as we sought to explain why some firms under resource scarcity tended to abandon design options, whereas others tended to take on more technical debt. We observed that firms could sometimes shift the burden of their technical debt to their customers or other members of their business ecosystem such as system integrators. For example, customers who are faced with high switching costs may be forced to bear the costs of incompatibility between releases of a vendor's product. Since firms that are able to transfer debt to other parties can significantly reduce their exposure to the future costs of their design decisions, they might be more inclined to make debt-increasing moves than firms that must bear these costs in full.

Figure 3 summarizes the four pairs of propositions that emerged from our case studies. They are overlaid on a design capital map to illustrate that each pair applies to a specific quadrant. The figure further indicates the strategic contingencies that apply to each proposition, and the predicted impact of a design move enacted under the specified conditions. Table 5 presents the same information in tabular form, along with a summary of the case evidence, both supporting and contradictory, for each proposition. In some moves, the predicted impact occurred, but it did not result in a change of design capital state, whereas, in other moves, the predicted impact occurred along with other impacts that were not predicted. These instances are noted with asterisks to facilitate reconciling the table with the four paths shown in Figure 2.

Option-Constrained Design Capital

Startup firms and new business units tend to possess option-constrained design capital (the top-left quadrant of Figure 3). We observe that such firms can make two distinct kinds of

[5]This is loosely analogous to a Markov model of a dynamical process. In reality, a firm's choice of design moves may be path-dependent and affected by many contingencies in a complex way. However, we believe it is still useful to view our case evidence through the lens of a simple "memoryless" model (Feller 1971).

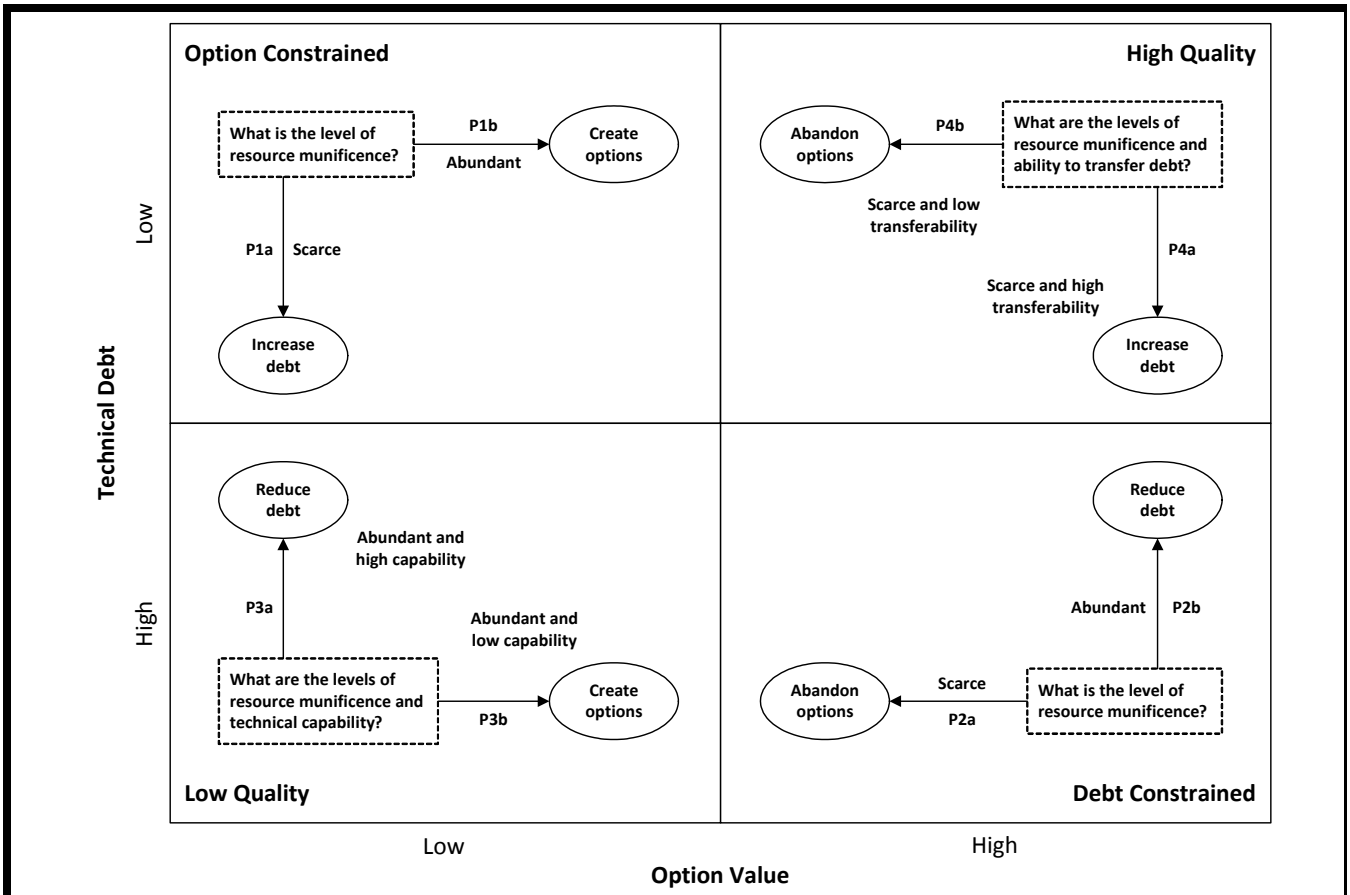

Figure 3. Schematic View of Propositions

design moves: one kind increases the option value of its designs (i.e., a move to the right), while the other kind increases its technical debt (i.e., a move downward).

To explain why a firm would make one kind of move rather than another, we focus attention on the relative level of resource munificence due to its strong theoretical and empirical support in our setting. When resources are scarce, designers experience pressures to address urgent business needs without the necessary slack to lay a firm foundation for the future. Even skilled and well-intentioned designers often cut corners, exploiting existing options for short-term gain in ways that make it harder to create or exploit other options in the future. This leads to an accumulation of technical debt. For example, Testco reused an existing signal processing module when it was under pressure to launch a new product on an extremely aggressive schedule (Testco 1B). Since speed was essential, Testco's designers made a conscious choice to accelerate the product cycle at the cost of accumulating technical debt, which they expected to eliminate by replacing the module in a future release cycle. Similarly, in the Infocom case, the firm accumulated technical debt by collating data from three legacy billing applications—rather than consolidating them—in order to address an urgent need to deliver integrated utility bills to customers (Infocom 1C).

In contrast, when firms have abundant resources, designers have the necessary slack to enact moves with a longer planning horizon, and often use this opportunity to create options that can be harvested in the future. Evidence from our cases lends support to this narrative. Designers at Softsys, Cellsys, and Infocom successfully applied abundant resources to create design options, thereby shifting their respective firms' design capital from an option-constrained state to a high-quality one. Softsys created a new configuration engine (Softsys 1A), Cellsys created an option for multi-homing, allowing its application to run on a variety of mobile platforms (Cellsys 1A), and Infocom exposed an interface for social media applications (Infocom 3A).

Based on this case evidence, we state our first pair of propositions:

Table 5. Proposition and Case Evidence Summary

Proposition	Specified Conditions		Predicted Impact of Design Move	Case Evidence	
	State of Design Capital	Strategic Contingencies		Supporting	Contradictory
P1a	Option constrained	Resource scarcity	Increase debt	Testco 1B Infocom 1C	None
P1b	Option constrained	Resource abundance	Create options	Softsys 1A Cellsys 1A Infocom 3A Testco 1A*	None
P2a	Debt constrained	Resource scarcity	Abandon options	Infocom 1B**, 2C**	Softsys 2C
P2b	Debt constrained	Resource abundance	Reduce debt	Cellsys 2B Softsys 3A** Testco 3A**	Cellsys 2A Infocom 2B
P3a	Low quality	Resource abundance High technical capability	Reduce debt	Testco 2A**	None
P3b	Low quality	Resource abundance Low technical capability	Create options	Infocom 2A	None
P4a	High quality	Resource scarcity High ability to transfer debt	Increase debt	Softsys 1C*, 2A*, 2B Cellsys 1B	None
P4b	High quality	Resource scarcity Low ability to transfer debt	Abandon options	Softsys 3C* Testco 2B, 3B*	None

*Predicted impact occurred but did not change design capital state
**Predicted impact occurred along with other impacts not predicted

Proposition 1a: *Under option-constrained design capital and resource scarcity, a firm's design moves will tend to increase the technical debt of its design capital.*

Proposition 1b: *Under option-constrained design capital and resource abundance, a firm's design moves will tend to create design options in its design capital.*

Debt-Constrained Design Capital

In a debt-constrained design capital state, a firm possesses high option value, but it is weighed down by high levels of technical debt. Although its option richness gives it an advantage in being able to launch many offerings, the cost of exercising these options (e.g., in development effort to work around known bugs in a system on which these offerings depend) hampers its speed and effectiveness.

As in the option-constrained state, the feasible paths out of a debt-constrained state depend on the level of resource munificence. Under resource scarcity, investments in reducing technical debt through additional development efforts are unlikely to be feasible, but firms can also meet their short-term obligations by abandoning debt-laden modules and their associated design options. While this might still leave a firm with a high level of technical debt (shifting its position horizontally to the left on the design capital map into a low-quality state), we were not surprised to see option abandonment used as a debt reduction strategy, resulting in diagonal moves from debt-constrained to option-constrained states. For example, the designers at Infocom abandoned their efforts to build a common database schema across three separate billing applications (move 1B). While this move eliminated a potentially valuable option for unifying its billing platforms, it also freed the firm from risky and expensive integration efforts. Similarly, when Infocom abandoned a new account management module that was causing headaches for customers (move 2C), this move foreclosed the options embedded in the rewards and loyalty programs that were tied to this module, but it also reduced the firm's customer support costs.

Conversely, debt-constrained firms that have access to abundant resources can afford to reduce their debt without abandoning options, thereby moving toward the high-quality design capital state. For example, when Cellsys designers wanted to add operator-specific features to the Cellsys platform, they restructured the system into two different product variants, one for retail end users and the other for telecommunication operators (move 2B). This design move eliminated the technical debt associated with testing and ensuring compatibility in a tightly integrated system catering to two

different customer bases. At a later point, Cellsys designers pursued options for enabling user-led innovations, and they were then more easily able to add software development kits and application stores specific to each customer base. Two other moves (Softsys 3A and Testco 3A) also propelled firms from a debt-constrained design capital state to a high-quality state. These moves were more complex as they involved the creation and/or abandonment of design options as well.[6]

These observations lead to our next pair of propositions:

Proposition 2a: *Under debt-constrained design capital and resource scarcity, a firm's design moves will tend to abandon design options.*

Proposition 2b: *Under debt-constrained design capital and resource abundance, a firm's design moves will tend to reduce its technical debt.*

Although these propositions may be intuitive given the structure of the design capital map, our cases offer mixed evidence for them. For example, Softsys's move 2C unfolded in a way that is counter to proposition 2a. Our proposition would have predicted the abandonment of the custom extension option by Softsys designers, since the firm was resource scarce and carrying a high level of debt at the time. But instead, Softsys designers accumulated more debt by continuing to develop custom extensions. Paradoxically, this locked the firm into a virtuous cycle. Rather than being paralyzed by the accumulated debt, Softsys's system integration services division simply charged its clients to fix the problems that arose. Having invested in Softsys enterprise software applications and extensively customized the applications to their needs, these clients faced high switching costs and were essentially locked into the product. Thus, Softsys was perversely incentivized to accumulate more debt by creating additional custom extension points, since the resulting costs could be transferred to its clients. As noted before, this suggests that the ability to transfer debt is an important additional contingency to consider. We do so explicitly in the fourth pair of propositions below.

More countervailing evidence is offered by Cellsys 2A and Infocom 2B, two moves that did not conform to proposition 2b. Both of these moves created more options instead of reducing debt when the respective firms were in a debt-constrained state with abundant resources. Cellsys added an operator-focused module and Infocom installed an integrated CRM database. Through discussions with designers at the firms, we learned that they initially responded to business pressures to create more functionality, but then quickly realized that they could not exploit these options without addressing the debt they had created. In fact, both firms addressed the high-debt situation in their next moves: Cellsys completely refactored its system to separate out retail- and operator-specific modules (Cellsys 2B) and Infocom abandoned their ill-conceived account management module that allowed overly complex customizations (Infocom 2C).[7]

Low-Quality Design Capital

We next consider low-quality design capital, a state in which a firm possesses few design options, and its ability to exercise the options it does possess is impeded by high technical debt. Although clearly undesirable, this state is not uncommon; two of our four case firms experienced it during our observation period.

Once again, resource munificence strongly influences the design moves available to a firm in this state. However, when designers face both low-quality design capital and a resource-scarce environment, their choice of design moves is over-constrained. On one hand, they are limited in their ability to extract value from their existing design capital, either by exercising options that were created in the past or borrowing from the future in the form of technical debt. On the other hand, without additional resources they cannot invest in improving the quality of their designs. Our conceptual model offers little guidance in this situation, other than to suggest that it is unsustainable; either the product or project needs to be "bailed out" (e.g., by investors or other parts of the firm), or higher management needs to "pull the plug" on it. Both of these outcomes are common in practice, although we only observed the former in our study. In the Infocom case, IT personnel in each of the firm's three service divisions were

[6]Both of these moves involved technological discontinuities (Anderson and Tushman 1990) and user-led innovation (von Hippel and Katz 2002). Softsys took advantage of the emergence of a component-based software development paradigm, and Testco embraced the emerging Microsoft .NET platform to build a product family architecture. Moreover, both firms had experimented with user community-generated code and sought ways to control the open innovation process. We highlight the need to develop a more nuanced understanding of the way these factors influence design moves as an open issue for future research.

[7]These examples of countervailing evidence highlight the potential for the coding of design move boundaries ("chunking" of moves) to affect tests of our propositions. We adopted a conservative approach, treating the evidence as contradictory even though it would have been consistent if we had either combined successive moves or relaxed our interpretation of the propositions to "look ahead" farther into the future.

initially overconstrained by aging legacy systems and lack of funding (Infocom 1A), but were able to exit the low-quality state with the aid of fresh resources that became available as a result of merging the three divisions into a single new operating company.[8]

In the absence of resource constraints, designers can exit a low-quality design capital state either by reducing debt or creating options. At Testco, designers reduced their technical debt by completely rewriting a legacy application using a newer programming language (Testco 2A). These efforts also succeeded in expanding the option value of the firm's design capital. In contrast, Infocom IT personnel created an option for tighter integration with a new CRM platform after substituting newer billing applications for the existing legacy ones (Infocom 2A). Their efforts were aimed primarily at increasing option value rather than reducing technical debt. Although both design moves occurred under conditions of resource abundance, the organizations that enacted them differed substantially in their level of technical capability. The teams at Testco had three times as much software development experience as the teams at Infocom. Moreover, Testco had the capability to build its own product components, whereas Infocom relied on external vendors for most of its product and component design. Thus, Testco designers had the know-how to avoid longer-term debt obligations and the necessary capabilities to take an aggressive stance toward reducing technical debt. In contrast, the inexperienced and less technically skilled Infocom IT personnel focused on creating options (largely driven by requirements handed to them by business unit managers); the importance of limiting technical debt to a manageable level became salient only at a later stage.

Based on the above evidence, we propose the following:

Proposition 3a: *Under low-quality design capital and resource abundance, a firm's design moves will tend to reduce technical debt if the technical capability of the firm is high.*

Proposition 3b: *Under low-quality design capital and resource abundance, a firm's design moves will tend to create design options if the technical capability of the firm is low.*

[8]The Infocom merger is an example of an organizational design move, another topic that warrants further exploration but lies beyond the scope of the present paper. See Woodard and West (2011) for a discussion of strategic design decisions that span the domains of technology and organizations.

High-Quality Design Capital

When a firm enjoys high-quality design capital with high option value and low technical debt, its design capital can confer unfettered competitive advantage. As noted above, high-quality design capital enhanced our case firms' ability to launch digitally enabled products and services, react to market opportunities and competitive threats, and shape their business ecosystems.

If resources are scarce, however, a firm may need to "draw down" the value of its design capital by abandoning options or taking on technical debt, resulting in a shift toward either to the option-constrained or debt-constrained state. We observed that differences in a firm's ability to transfer the costs of technical debt to other members of its ecosystem (including end users) provided a systematic explanation for the differing transitions we observed. Softsys, for example, achieved speed in system implementation during the early stages of the product life cycle by matching end-user requirements with previous configurations in the firm's knowledge repository (Softsys 1C). These configurations were often a highly imperfect match, but the costs of further narrowing the gap between what a client wanted and what the knowledge repository offered was borne by the client rather than the firm. When clients further modified the Softsys-recommended configurations, they typically signed separate maintenance contracts to cover support for these customizations, which effectively transferred the future debt obligations to the firm's clients.

In contrast, at a later stage in the Softsys product life cycle when the quality of the firm's design capital had improved but its designers faced greater resource scarcity, they blocked moves that would have incurred debt obligations and chose to abandon options instead (Softsys 3C). The firm's environment had significantly changed, notably due to fact that several of its third-party system integration partners were now using the platform architecture it had created. Softsys realized that these partners, which included well-known global IT service providers, would vigorously resist changes that imposed additional costs or undermined their own complementary development efforts. Softsys therefore found it considerably more difficult to transfer its debt obligations to other members of its ecosystem. Hence, Softsys designers chose to abandon options rather than incur new technical debt, instead relying on its partner ecosystem to enhance the option value of its platform architecture.

A similar pattern was seen in the design moves of the other firms in our study. Cellsys was able to accumulate technical debt (Cellsys 1B) and transfer some of it to end users whose

data was locked into the Cellsys application. However, Testco had to abandon options in situations where it could not transfer debt associated with maintaining support for incompatible standards implementations (Testco 2B, 3B).

Based on these observations, we propose:

Proposition 4a: *Under high-quality design capital and resource scarcity, a firm's design moves will tend to increase technical debt if the firm's ability to transfer technical debt to other members of its ecosystem is high.*

Proposition 4b: *Under high-quality design capital and resource scarcity, a firm's design moves will tend to abandon design options if its ability to transfer technical debt to other members of its ecosystem is low.*

What if a firm is fortunate enough to be in a position of high-quality design capital and resource abundance? As a mirror of the overconstrained situation considered earlier (low-quality design capital and resource scarcity), this describes an underconstrained situation for designers that favors experimentation and entrepreneurial actions. Designers in this situation have the freedom to explore alternative designs that can create enormous value and transform the relationships among members of their business ecosystem. Competing successfully in such an environment requires effective sense-making and the ability to cope with complexity and uncertainty (Brown and Eisenhardt 1997).

One observed pattern did not fit neatly into our conceptual model: firms with high quality design capital tended to create product or service platforms that, in turn, enabled them to sustain a state of high-quality design capital. We observed two distinct kinds of platform strategies, consistent with the typology proposed by Gawer (2009): one focused on internal economies of scale and scope, and the other focused on influencing an external ecosystem. Softsys designers experimented with different ways to improve client-specific system performance by developing different customization mechanisms (Softsys 1B), which helped the firm launch and consolidate a new internal business division purely focused on system integration services. At a later point of time in the product's evolution, Softsys designers created an application composer (Softsys 3B), another novel customization mechanism that allowed external partners to create compatible applications. Similarly, in the Cellsys case, designers first created an internal platform to port applications across the different mobile platforms (Cellsys 3A), and later pursued external platform development by engaging third-party developers through design moves that yielded software development kits and application stores for the Cellsys platform (Cellsys 3B and 3C). In the Infocom case, designers in the high-quality design capital state chose to focus on a purely internal platform to help launch marketing campaigns through social media (Infocom 3B).

The above-mentioned pattern in our case data shows that resource abundance coupled with high-quality design capital (i.e., low technical debt and high option value) engender platform creation, but the way the platforms were created and used varied. In the early stages of product development, although Softsys and Cellsys possessed high-quality design capital, they were smaller players in their respective business ecosystems and not in a position to wield control over suppliers or customers. They chose to create internal platforms that eventually helped them develop specialized expertise and capabilities to differentiate from competitors and increase market share. In contrast, at a later stage in their evolution, when both firms had gained a more central presence in their ecosystems and had developed mechanisms to manage the diversity of business partners, they chose to develop external platforms. This case evidence suggests an interesting linkage between a firm's ability to appropriate value from its platform ecosystem and its decision to focus on internal or external platform creation during high-quality design capital regimes, which we highlight as an opportunity for further study.

Discussion

Firms pursuing digital business strategies must manage a fundamental tension between the need to support flexible adaptation of their products and services to changing market conditions, and the need to provide stable value appropriation mechanisms to extract economic rents and reinvest them in innovative activities. The competitive actions of such firms are receiving fresh scrutiny as scholars and practitioners increasingly focus their attention on situations in which companies fuse IT with their products and services for competitive advantage (El Sawy et al. 2010; Tanriverdi et al. 2010; Yoo et al. 2010). In this research, we developed a conceptual framework that emphasizes the strategic importance of the cumulative stock of designs owned or controlled by a firm (design capital), and the sequence of discrete strategic actions that increase, reduce, or modify a firm's design capital (design moves). We conceptualized design capital as a two-dimensional construct comprising option value and technical debt, and adopted the design move as a unit of analysis to explore the duality between design moves and design capital. We deployed our conceptual framework to examine four case studies and developed a set of propositions on the relationship

between design capital states and the subsequent design moves. The conceptual framework and the propositions developed in the paper can serve as a robust foundation for theorizing about the logic of digital business strategy.

Contributions

This paper makes several contributions to theory development. We advance the notion that conceptualizations of digital business strategy can and should be grounded in the strategic role of design. In digital business environments, resource allocation decisions about organization design and competitive strategy are difficult to separate from decisions about product development and broader design decisions, and therefore need to be looked at through the "fusion view of IS" (El Sawy 2003). In these environments, IT investments are enacted through design moves, many of which have the structure of real options whose value may be discovered through the actions of designers and other environmental conditions that unfold over the period of product development. Moreover, resource allocation decisions of digital businesses are not taken in isolation, but are driven by the collective behavior of the customers, competitors, and complementors who interact in a business ecosystem (Iansiti and Levien 2004). Hence, examining the resource allocation decisions and performance outcomes of IT investments without explicitly considering investments in design activities will result in a biased valuation of the returns on IT by failing to adequately account for the different pathways through which firms build and leverage design capital. The conceptual framework, propositions, and the case illustrations presented in this paper go beyond the "alignment view of IT" (Henderson and Venkatraman 1993) and embrace the strategic role of design. Thus, this paper takes a step forward in answering the call for fresh inquiry into how IT and business capabilities are fused together in the case of continually adapting modern digital businesses (El Sawy et al. 2010; Tanriverdi et al. 2010; Yoo et al. 2010).

At the heart of our conceptual model is a concern shared by many strategic frameworks: the fundamental tension between the short term and the long term, with uncertainty serving as a wedge. A variety of concepts have been offered to address this concern, including the exploration–exploitation dichotomy of Levinthal and March (1993), processes of improvisation and experimentation (Brown and Eisenhardt 1997), and the need for organizational ambidexterity (Raisch et al. 2009). The conceptual framework advanced in this study contributes to reconciling the tension between the short-term and long-term actions of a continuously adapting firm by focusing on the generative and path-dependent nature of design evolution. Design moves can be viewed as traces of sense-making in dynamic and turbulent environments where competitive advantages are often short-lived (Brown and Eisenhardt 1997; D'Aveni et al. 2010). In this sense, our conceptualization highlights ways in which design options and the corresponding penalties of incurred technical debt can link the complexity of an uncertain future with a firm's decision to manage its design capital for short- and long-term competitive advantage. Digital options are at once a means of preserving the opportunity to capitalize on a new technology or practice, and of mitigating the risks induced by technological and market uncertainty. The concept of technical debt encompasses the locked-in nature of designs, the performance costs of inappropriate technology choices, and the corresponding opportunity costs of missing disruptive technologies that can move a firm to a higher performance plane. Examples from our cases include fostering cross-platform economies of scale (Cellsys); reusing, recombining, and at times abandoning designs (Testco, Infocom); balancing digital services and product businesses by investing in design for the scalability of services (Softsys); and investing to delay full-scale commitment while preserving the ability to act quickly (all four cases).

Finally, we contribute to an emerging body of interdisciplinary literature on modularity and design (e.g., Baldwin and Clark 2000; Cai and Sullivan 2006; LaMantia et al. 2008; MacCormack et al. 2006), which aims to model and explain the strategic consequences of designers' actions. By more precisely illustrating how designs are generative and limited by technical and organizational realities, we explain how path dependence and path creation are both jointly facilitated by resource allocation decisions and design artifacts. Product development and design decisions need to be examined by grounding them in the specific environmental context of firms in order to become useful as a tool to understand the logic of digital business strategy. Our empirical approach facilitates this by using design moves to help assess the broader implications of the actions of designers, including architectural and strategic impacts. Further, our empirical approach operationalized the design capital state of a firm using a parsimonious and measurable set of variables such as option value and technical debt, and demonstrated how to rigorously trace the transitions in design capital states of a firm by sequencing design moves. Thus, this research bridges modularity concepts in strategy (Baldwin and Clark 2000) and software engineering (Brown et al. 2010; Guo and Seaman 2011; Nugroho et al. 2011), and contributes to a broader goal of establishing an empirically traceable link between the design of digital artifacts, competitive strategies built on these designs, and firm-level performance.

Implications for Practice

The significance of our research to practitioners involved in digital innovation is exemplified in a recent postmortem analysis of the failure of Myspace:

> Part of the reason Myspace struggled to keep up with emerging technology companies was its site architecture…speed to market was essential. Friendster knockoffs were popping up everywhere. Myspace's founders [Chris DeWolfe and Tom Anderson] decided to build the site using ColdFusion, a simplistic programming language. "ColdFusion, even back then, in the engineering world, was thought to be a sort of Mickey Mouse type of technology," says DeWolfe. "But it was so easy to use that we could just crank it out quickly. We blew out Friendster. We blew out Tribe.net. We blew out everyone." They also created what DeWolfe calls "technology debt." By 2005 the site had outgrown ColdFusion. At that point it was too late to switch over to the open-source-code software favored by developers; changing would have delayed the site for a year or two just as it was exploding in popularity. The easiest move, says DeWolfe, was to switch to .NET, a software framework created by Microsoft. "Using .NET is like Fred Flintstone building a database," says David Siminoff, whose company owns the dating website JDate, which struggled with a similar platform issue (Gillette 2011).

To better inform decision makers faced with situations like the one described above, it would be helpful to view the decisions made by designers and corporate strategists in a holistic manner. This research advances a framework that integrates the decision-making perspectives of designers and corporate strategists; the impact of discrete actions of designers can be measured and traced by corporate strategists using the design capital map of our framework (Figure 1) in order to make judicious resource allocation decisions. Thus, the conceptual framework and empirical approach advanced in this study contribute to the development of a theoretically sound and practically relevant tool for IT and product development governance.

Future Work

Designs (as an important form of capital) are at the heart of the strategic framework advanced in this study. We acknowledge that factors affecting designs are located at multiple levels of a firm's hierarchical structure (e.g., engineers, product managers, corporate strategists, etc.). Our ability to apply the current conceptual framework is limited by the fact that the interactions among agents at multiple levels of a firm's hierarchy and their individual decision calculus are difficult to observe directly. This could be explored in further work, and we believe that such exploration would be facilitated by the empirical approach of using design moves as a unit of analysis. Shedding light on the interactions between agents in the domains of design and strategy might provide a behavioral context to the forces that shape the duality between design moves and design capital, which might help us uncover more nuanced strategic contingencies.

Due to the limitations of our case sample, we could not comprehensively examine the impact of a few environmental and contextual factors we observed in cases that could potentially shape the transitions of firms from one design capital state to another. For example, we did not have sufficient basis to fully analyze the variance between our cases on industry clockspeed (Mendelson and Pillai 1998), competition intensity, and the level of environmental turbulence (D'Aveni et al. 2010; Pavlou and El Sawy 2006). Expanding the empirical analysis to firms in a more diverse set of industry domains might help us generalize our propositions and develop a more comprehensive contingency framework.

Several other ways in which the theoretical perspectives developed in this paper can be extended by future research include (1) classifying the types of design moves and digital options in a variety of contexts, systematically modeling their economic properties in more formal detail; (2) exploring the behavioral propensities of designers and their attitudes and beliefs toward technical debt and the corresponding impact on the patterns of design moves that they enact; (3) investigating IT governance mechanisms that help in making design capital parameters such as technical debt visible for upper-echelon managers in charge of strategy; and (4) developing a generalizable process model of value creation and capture in design-oriented ecosystems by further investigating design moves and ecosystem dynamics such as the actions of competitors, complementors, and customers and the influence of their activities. We see these as promising directions that can contribute to a more comprehensive theory of digital business strategy grounded in the strategic role of design, and look forward to pursuing them in future work.

Acknowledgments

We are grateful to the editors of the *MIS Quarterly* Special Issue on Digital Business Strategy for encouragement and guidance on the development of this paper, as well as to our review team for detailed and constructive feedback. We are also indebted to Carliss Baldwin

and Alan MacCormack for sharing their insights on the relationship between option value and technical debt, while fully absolving them of responsibility for errors or omissions in the way we have applied their ideas. Jason Woodard and Narayan Ramasubbu thank the School of Information Systems at Singapore Management University for generously supporting this research, and Ted Tschang thanks Ravi Sharma of Nanyang Technological University for supporting one exploratory phase of this study through a Singapore National Research Foundation grant.

References

Adner, R., and Kapoor, R. 2010. "Value Creation in Innovation Ecosystems: How the Structure of Technological Interdependence Affects Firm Performance in New Technology Generations," *Strategic Management Journal* (31:3), pp. 306-333.

Adner, R., and Levinthal, D. A. 2004. "What Is *Not* a Real Option: Considering Boundaries for the Application of Real Options to Business Strategy," *Academy of Management Review* (29:1), pp. 74-85.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S. 1977. *A Pattern Language: Towns, Buildings, Construction*, New York: Oxford University Press.

Anderson, P., and Tushman, M. L. 1990. "Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change," *Administrative Science Quarterly* (35:4), pp. 604-633.

Baldwin, C. W., and Clark, K. B. 1997. "Managing in the Age of Modularity," *Harvard Business Review* (75:5), pp. 84-93.

Baldwin, C. Y., and Clark, K. B. 2000. *Design Rules, Volume 1: The Power of Modularity*, Cambridge, MA: MIT Press.

Baldwin, C. Y., and Clark. K. B. 2006. "Modularity in the Design of Complex Engineering Systems," in *Complex Engineering Systems: Science Meets Technology*, D. Braha, A. A. Minai, and Y. Bar-Yam (eds.), New York: Springer, pp. 175-206.

Baldwin, C. Y., and MacCormack, A. 2011. "Finding Technical Debt in Platform and Network Architectures," paper presented at The Mathworks, Inc., April 6.

Banker, R. D., Bardhan, I. R., Chang, H., and Lin, S. 2006. "Plant Information Systems, Manufacturing Capabilities, and Plant Performance," *MIS Quarterly* (30:2), pp. 315-338.

Barney, J. 1991. "Firm Resources and Sustained Competitive Advantage," *Journal of Management* (17:1), pp. 99-120.

Beath, C. M., and Ives, B. 1986. "Competitive Information Systems in Support of Pricing," *MIS Quarterly* (10:1), pp. 85-96.

ben-Aaron, D. 2011. "Nokia Tumbles on Concern Partnership with Microsoft 'May Kill' Phonemaker," *Bloomberg*, February 11 (available at http://www.bloomberg.com/news/2011-02-11/nokia-joins-forces-with-microsoft-to-challenge-dominance-of-apple-google.html; retrieved on July 15, 2011).

Blandford, R. 2011. "Understanding Nokia's Smartphone Strategy Decision" (available at http://mediafiles.allaboutsymbian.com/understanding-nokia-smartphone-strategy.pdf; retrieved on July 15, 2011).

Boudreau, K. 2010. "Open Platform Strategies and Innovation: Granting Access vs. Devolving Control," *Management Science* (56:10), pp. 1849-1872.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., Sangwan, R., Seaman, C., Sullivan, K., and Zazworka, N. 2010. "Managing Technical Debt in Software-Reliant Systems," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, November 7-11, Santa Fe, NM, pp. 47-52.

Brown, S. L., and Eisenhardt, K. M. 1997. "The Art of Continuous Change: Linking Complexity Theory and Time-Paced Evolution in Relentlessly Shifting Organizations," *Administrative Science Quarterly* (42:1), pp. 1-34.

Cai, Y., and Sullivan, K. J. 2006. "Modularity Analysis of Logical Design Models," in *Proceedings of the 21st IEEE International Conference on Automated Software Engineering*, September 18-22, Tokyo, Japan, pp. 91-102.

Castrogiovanni, G. J. 1991. "Environmental Munificence: A Theoretical Assessment," *Academy of Management Review* (16:3), pp. 542-565.

Chen, B. X. 2012. "In Flop of H.P. TouchPad, an Object Lesson for the Tech Sector," *The New York Times*, Technology Section, January 1 (available at http://www.nytimes.com/2012/01/02/technology/hewlett-packards-touchpad-was-built-on-flawed-software-some-say.html; retrieved on February 23, 2012).

Cross, N. 2011. *Design Thinking: Understanding How Designers Think and Work*, London: Berg Publishers.

Cunningham, W. 1992. "The WyCash Portfolio Management System," in *Proceedings on Object-Oriented Programming Systems, Languages, and Applications* (Addendum), October 18-22, Vancouver, Canada, pp. 29-30.

Cusumano, M. A. 2010. *Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Uncertain World*, New York: Oxford University Press.

D'Aveni, R. A., Dagnino, B. G., Smith, K. G. 2010. "The Age of Temporary Advantage," *Strategic Management Journal* (31:13), pp. 1371-1385.

Dess, G. G., and Beard, D. W. 1984. "Dimensions of Organizational Task Environments," *Administrative Science Quarterly* (29:1), pp. 52-73.

Dhar, V., and Sundararajan, A. 2007. "Information Technologies in Business: A Blueprint for Education and Research," *Information Systems Research* (18:2), pp. 125-141.

Eisenhardt, K. M., and Graebner, M. E. 2007. "Theory Building from Cases: Opportunities and Challenges," *Academy of Management Journal* (50:1), pp. 25-32.

El Sawy, O. A. 2003. "The IS Core IX: The 3 Faces of IS Identity: Connection, Immersion, and Fusion," *Communications of the AIS* (12), pp. 588-598.

El Sawy, O. A., Malhotra, A., Park, Y., and Pavlou, P. 2010. "Seeking the Configurations of Digital Ecodynamics: It Takes Three to Tango," *Information Systems Research* (21:4), pp. 835-848.

Feller, W. 1971. *An Introduction to Probability Theory and its Applications, Volume 2* (2nd ed.), New York: Wiley.

Fowler, M. 2009. "Technical Debt Quadrant," October 14 (available at http://martinfowler.com/bliki/TechnicalDebtQuadrant.html; retrieved on August 1, 2011).

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading, MA: Addison-Wesley.

Garud, R., and Kumaraswamy, A. 1995. "Technological and Organizational Designs for Realizing Economies of Substitution," *Strategic Management Journal* (16:1), pp. 93-109.

Garud, R., Kumaraswamy, A., and Langlois, R. N. (eds.). 2003. *Managing in the Modular Age: Architectures, Networks, and Organizations,* Malden, MA: Blackwell.

Gawer, A. 2009. "Platform Dynamics and Strategies: From Products to Services," in *Platforms, Markets and Innovation*, A. Gawer (ed.), Cheltenham, UK: Edward Elgar, pp. 45-76.

Gillette, F. 2011. "The Rise and Inglorious Fall of Myspace," *Bloomberg Business Week*, June 27.

Goffman, E. 1981. "Replies and Responses," in *Forms of Talk*, Philadelphia, PA: University of Pennsylvania Press, pp. 5-77.

Guo, Y., and Seaman, C. B. 2011. "A Portfolio Approach to Technical Debt Management," in *Proceedings of the Second International Workshop on Managing Technical Debt, International Conference on Software Engineering*, May 21-28, Honolulu, HI, pp. 31-34.

Helfat, C. E., and Raubitschek, R. S. 2000. "Product Sequencing: Co-evolution of Knowledge, Capabilities, and Products," *Strategic Management Journal* (21:10/11), pp. 961-980.

Henderson, R. M., and Clark, K. B. 1990. "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms," *Administrative Science Quarterly* (35:1), pp. 9-30.

Henderson, J. C., and Venkatraman, N. 1993. "Strategic Alignment: Leveraging Information Technology for Transforming Organizations," *IBM Systems Journal* (32:1), pp. 4-16.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence*, Ann Arbor, MI: University of Michigan Press.

Iansiti, M., and Levien, R. 2004. *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*, Boston: Harvard Business School Press.

Ives, B., and Learmonth, G. P. 1984. "The Information System as a Competitive Weapon," *Communications of the ACM* (12:3), pp. 1193-1201.

LaMantia, M. J., Cai, Y., MacCormack, A., and Rusnak, J. 2008. "Analyzing the Evolution of Large-Scale Software Systems Using Design Structure Matrices and Design Rule Theory: Two Exploratory Cases," in *Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture*, February 18-21, Vancouver, Canada, pp. 83-92.

Levinthal, D. A., and March, J. 1993. "The Myopia of Learning," *Strategic Management Journal* (14:S2), pp. 95-112.

Lu, Y., and Ramamurthy, K. 2011. "Understanding the Link Between Information Technology Capability and Organizational Agility: An Empirical Examination," *MIS Quarterly* (35:4), pp. 931-954.

MacCormack, A., Rusnak, J., and Baldwin, C. Y. 2006. "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," *Management Science* (52:7), pp. 1015-1030.

Mace, M. 2010. "What's Really Wrong with BlackBerry (and What To Do About It)" Mobile Opportunity blog (http://mobileopportunity.blogspot.com/2010/10/whats-really-wrong-with-blackberry-and.html; retrieved on July 15, 2011).

Makadok, R. 2001. "Toward a Synthesis of the Resource-Based View and Dynamic-Capability Views of Rent Creation," *Strategic Management Journal* (22:5), pp. 387-401.

Melville, N., Kraemer, K., and Gurbaxani, V. 2004. "*Review* Information Technology and Organizational Performance: An Integrative Model of IT Business Value," *MIS Quarterly* (28:2), pp. 283-322.

Mendelson, H., and Pillai, R. 1998. "Clockspeed and Informational Response: Evidence from the Information Technology Industry," *Information Systems Research* (9:4), pp. 415-433.

Mithas, S., Ramasubbu, N., and Sambamurthy, V. 2011. "How Information Management Capability Influences Firm Performance," *MIS Quarterly* (35:1), pp. 237-256.

Morris, C. R., and Ferguson, C. H. 1993. "How Architecture Wins Technology Wars," *Harvard Business Review* (71:3), pp. 86-95.

Nugroho, A., Visser, J., and Kuipers, T. 2011. "An Empirical Model of Technical Debt and Interest," in *Proceedings of the Second International Workshop on Managing Technical Debt, International Conference on Software Engineering*, May 21-28, Honolulu, HI, pp. 1-8.

Osterwalder, A., and Pigneur, Y. 2010. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*, Hoboken, NJ: Wiley.

Pavlou, P. A., and El Sawy, O. 2006. "From IT Leveraging Competence to Competitive Advantage in Turbulent Environments: The Case of New Product Development," *Information Systems Research* (17:3), pp. 198-227.

Pavlou, P. A., and El Sawy, O. A. 2010. "The 'Third Hand': IT-Enabled Competitive Advantage in Turbulence Through Improvisational Capabilities," *Information Systems Research* (21:3), pp. 443-471.

Pentland, B. T. 1992. "Organizing Moves in Software Support Hot Lines," *Administrative Science Quarterly* (37:4), pp. 527-548.

Piccoli, G., and Ives, B. 1995. "*Review*: IT-Dependent Strategic Initiatives and Sustained Competitive Advantage: A Review and Synthesis of the Literature," *MIS Quarterly* (29:4), pp. 747-776.

Porter, M. E. 1980. *Competitive Strategy,* New York: Free Press.

Porter, M. E. 2001. "Strategy and the Internet," *Harvard Business Review* (79:3), pp. 62-78.

Porter, M. E., and Millar, V. E. 1985. "How Information Gives You Competitive Advantage," *Harvard Business Review* (63:4), pp. 149-160.

Ray, G., Muhanna, W. A., and Barney, J. B. 2005. "Information Technology and the Performance of the Customer Service Process: A Resource-Based Analysis," *MIS Quarterly* (29:4), pp. 625-652.

Raisch, S., Birkinshaw, J., Probst, G., and Tushman, M. L. 2009. "Organizational Ambidexterity: Balancing Exploitation and Exploration for Sustained Performance," *Organization Science* (20:4), pp. 685-695.

Sambamurthy, V., Bharadwaj, A., and Grover, V. 2003. "Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms," *MIS Quarterly* (27:2), pp. 237-263.

Schön, D. A. 1983. *The Reflective Practitioner: How Professionals Think in Action*, New York: Basic Books.

Smith, K. G., Ferrier, W. J., and Ndofor, H. 2001. "Competitive Dynamics Research: Critique and Future Directions," in

Handbook of Strategic Management, M. Hitt, R. E. Freeman, and J. S. Harrison (eds.), Oxford, UK: Blackwell, pp. 315-361.

Staw, B. M., and Szwajkowski, E. 1975. "The Scarcity-Munificence Component of Organizational Environments and the Commission of Illegal Acts," *Administrative Science Quarterly* (20:3), pp. 345-354.

Sullivan, K. J., Chalasani, P., Jha, S., and Sazawal, V. 1999. "Software Design as an Investment Activity: A Real Options Perspective," in *Real Options and Business Strategy: Applications to Decision Making*, L. Trigeorgis (ed.), London: Risk Books, pp. 215-262.

Sullivan, K. J., Griswold, W. G., Cai, Y., and Hallen, B. 2001. "The Structure and Value of Modularity in Software Design," in *Proceedings of the 9th International Symposium on Foundations of Software Engineering*, September 10-14, Vienna, Austria, pp. 99-108.

Tanriverdi, H., Rai, A., and Venkatraman, N. 2010. "Reframing the Dominant Quests of Information Systems Strategy Research for Complex Adaptive Business Systems," *Information Systems Research* (21:4), pp. 822-834.

Teece, D. J. 1986. "Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing and Public Policy," *Research Policy* (15:6), pp. 285-305.

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.

Van Maanen, J. 1988. *Tales of The Field: On Writing Ethnography,* Chicago: University of Chicago Press.

von Hippel, E., and Katz, R. 2002. "Shifting Innovation to Users via Toolkits," *Management Science* (48:7), pp. 821-833.

West, J. 2003. "How Open Is Open Enough? Melding Proprietary and Open Source Platform Strategies," *Research Policy* (37:2), pp. 1259-1285.

Woodard, C. J. 2008. "Architectural Control Points," in *Proceedings of the Third International Conference on Design Science Research in Information Systems and Technology*, May 7-9, Atlanta, GA, pp. 359-363.

Woodard, C. J., and West, J. 2011. "Four Perspectives on Architectural Strategy," in *Proceedings of the 32nd International Conference on Information Systems*, December 4-7, Shanghai, Paper 21.

Yin, R. K. 2009. *Case Study Research: Design and Methods*, Thousand Oaks, CA: Sage Publications.

Yoo, Y., Henfridsson, O., and Lyytinen, K. 2010. "The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research," *Information Systems Research* (21:4), pp. 724-735.

Zittrain, J. L. 2006. "The Generative Internet," *Harvard Law Review* (119:7), pp. 1974-2040.

About the Authors

C. Jason Woodard is an assistant professor in the School of Information Systems at Singapore Management University. His research explores the evolution of complex product and system designs, with a focus on the relationship between their architecture and the competitive strategy of their designers. Topics of recent work include compatibility and standardization, strategic design moves, and the dynamics of platform ecosystems. He studied business, economics, and computer science at Harvard University, where he received a Ph.D. in Information, Technology, and Management in 2006. Prior to that, he worked for IBM as a technical evangelist for the company's efforts related to Java, XML, and Linux. He received an A.B. from Princeton University in 1997, where he created an independent concentration to study information technology and society.

Narayan Ramasubbu is an assistant professor at the Katz Graduate School of Business at the University of Pittsburgh. Previously, he was an assistant professor at the School of Information Systems at Singapore Management University. He received the bachelor of engineering degree from Bharathiar University, India, and the Ph.D. degree from the University of Michigan, Ann Arbor. Prior to his academic career, he was a senior developer at SAP AG and CGI Inc. His research interests include software engineering economics with a focus on globally distributed product development and services delivery; the design, implementation, and governance of enterprise information systems; end-user interaction and user-led innovation; and IT strategy and generating business value from IT.

F. Ted Tschang is an associate professor of strategic management in the Lee Kong Chian School of Business at the Singapore Management University. His research focus has been on developing theories of creativity, design, and organization (teams) through the study of designed, experiential products, such as study of the videogame development process. He is also working on a virtual world ethnography focused at understanding community forms of organization and design. Prior to this, he worked at the Asian Development Bank Institute on the software industry takeoff and capability building process, as well as at the United Nations University and the Belfer Center at Harvard's Kennedy School of Government. He holds a Ph.D. in public policy and management from Carnegie Mellon University, as well as graduate degrees in electrical engineering and economics.

V. Sambamurthy (Ph.D., University of Minnesota, 1989) is the Eli Broad Endowed Professor at the Eli Broad College of Business at Michigan State University. His research examines how firms successfully leverage information technologies in their business strategies, products, services, and organizational processes. His research adopts the perspectives of CIOs and top management teams. Most of his work has been funded by the Financial Executives Research Foundation, the Advanced Practices Council (APC), and the National Science Foundation. His work has been published in journals such as *MIS Quarterly*, *Information Systems Research*, *Decision Sciences*, *Management Science*, *Organization Science*, and *IEEE Transactions on Engineering Management*. He has served as a senior editor for *MIS Quarterly*, departmental editor for *IEEE Transactions on Engineering Management*, and Americas editor for *Journal of Strategic Information Systems*. Recently, he completed a six-year term as the editor-in-chief of *Information Systems Research*. He was selected as a Fellow of the Association for Information Systems in 2009.

Appendix

Design Moves and Modular Operators

We define a design move as a discrete strategic action that changes the structure or function of a digital artifact (product or system). Design changes tend to follow patterns because designers tend to face similar problems and solving them in similar ways can afford reuse and reduce coordination costs (Alexander et al. 1977; Gamma et al. 1995). Baldwin and Clark (2000) identified six patterns that occur in the design of modular systems: splitting, substituting, augmenting, excluding, inverting, and porting.[9] Building on Holland (1975), they called these patterns modular operators and illustrated their economic properties through an extended case study of the computer industry.

Although we take a different approach in the main text of the paper, one can in principle characterize a design move in terms of one or more modular operators. The operators describe the generic action to be taken (e.g., split a module into submodules) while the context of the move provides the arguments (e.g., which module is to be split into how many pieces). Design moves can thus be viewed as modular operators instantiated within situational contexts. We demonstrate this approach in Table A1, which shows our enumeration of the Softsys design moves using the modular operators.

While modular operators provide a tantalizingly powerful way to describe complex sequences of design changes, we discovered that it is difficult (perhaps even impossible) to uniquely characterize a design move in terms of modular operators—at least without direct evidence of the designers' intent. This is because the same design change could be achieved using different but equivalent combinations of modular operators. For example, a design change accomplished using the inversion operator can also be achieved using a combination of substitution, augmentation, and exclusion.

While this discovery dampened our hope of connecting our empirical analysis more tightly to Baldwin and Clark's operators, it does not in any way affect the validity of our analysis itself. That is, a definitive description of a design move in terms of modular operators is not necessary for analyzing whether the design move increases or decreases option value and/or technical debt. This is especially true when researchers are able to triangulate data on option value and technical debt from multiple sources of information such as design documents, interviews with designers, and source code versioning catalogs. Thus, using design moves as a unit of analysis is empirically sound even in the absence of direct observational data on modular operators.

Table A1. Design Moves and Modular Operators

Design Move	Coding Using Modular Operators
1A	INVERT (scattered configuration information) and AUGMENT (configuration scripts) to create configuration engine module
1B	SUBSTITUTE (configuration engine) and AUGMENT (configuration database) to add new logging functionality to the configuration engine
1C	INVERT (configuration database) to SUBSTITUTE (configuration settings) during new implementation
2A	INVERT (scattered function calls) and AUGMENT (tool set) to add segments to function calls
2B	Allow customers to AUGMENT (customer-written code) to designated exit points to create new functionality
2C	SUBSTITUTE (tool set) to help customers create their own exit points for custom function calls
3A	EXCLUDE (customer-written code and function calls) and PORT (core functional modules) to a component-based architecture
3B	AUGMENT (application composer module) to component architecture
3C	Allow partners to AUGMENT (third-party components) to component repository and SUBSTITUTE (original components) during implementation

[9]Splitting involves breaking up a system into modules; substituting is exchanging one module design for another; augmenting is the addition of a new module to the system; exclusion is the removal of a module from the system; inverting makes visible design information that was formerly hidden (e.g., by creating an interface); porting enables a module designed for one system to be used with another.