Strategic Management Journal
Strat. Mgmt. J., **33**: 137–161 (2012)
Published online EarlyView in Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/smj.947
Received 20 January 2010; *Final revision received 13 June 2011*

DOES COMPLEXITY DETER CUSTOMER-FOCUS?

SENDIL K ETHIRAJ,[1]* NARAYAN RAMASUBBU,[2] and M.S. KRISHNAN[1]
[1] *University of Michigan, Ann Arbor, Michigan, U.S.A.*
[2] *School of Information Systems, Singapore Management University, Singapore*

Economic models suggest that firms use a simple cost-benefit calculation to evaluate customer requests for new product features, but an extensive organizational literature shows the decision to implement innovation is more nuanced. We address this theoretical tension by studying how firms respond to customer requests for incremental product innovations, and how these responses change when the requested innovation is complex. Using large sample empirical analyses combined with detailed qualitative data drawn from interviews, we find considerable variance in the relationship between customer demands, complexity, and investments in incremental innovations. The qualitative study revealed the importance of organization structures, competitive pressures, and incentives for resource allocation processes. Copyright © 2011 John Wiley & Sons, Ltd.

INTRODUCTION

The management literature largely assumes that firms prioritize customer demands, and yet operating a customer-focused firm is more often the exception than the norm. A survey of Amazon.com produces 9,000 books about improving customer-focus in firms, suggesting that a customer-focus is both desirable and difficult to implement. The classical academic canon supporting a customer-focus argues that customers spur firms to invest in innovation and channel that investment in particular directions (Schmookler, 1966). Subsequent research explores the boundaries of customers' impact on innovation, finding, for example, that firms are less likely to prioritize customer demands during radical innovation for fear of cannibalizing existing products (Tushman and Anderson, 1986); that specialized communication routines within an organization can impede customer-focus during architectural innovation (Henderson and Clark, 1990); and that a myopic focus on current customers can undermine meeting future customer needs (Christensen, 1997). Despite these boundaries, most innovation literature continues to assert that demand driven innovation (DDI) (Reinganum, 1985) will predict a firm's *incremental* innovation efforts. This assumption runs counter to the literature on the microfoundations of decision making, which argues that organizational decision making processes condition a firm's investment decisions (Bower, 1970; Burgelman, 1994). Customer requests filtered through a firm's decision-making structure may not survive, and those that do may be accorded a lower priority than other firm goals. Reconciling this apparent contradiction—between the theoretical canon on customer-focused

Keywords: innovation; customer-focus; complexity; qualitative study; software industry
* Correspondence to: Sendil K Ethiraj, University of Michigan, 701 Tappan Ave, #R4442, Ann Arbor, MI 48109, U.S.A.
E-mail: sendil@umich.edu

innovation and the empirical literature on the microfoundations of organizational decision making—is the focus of this paper.

The literature on the individual and organizational processes affecting investment in innovation is large and crosses several disciplines. Some study individual-level effects, such as managerial risk aversion (Kahneman and Lovallo, 1993), choice bracketing (Read, Loewenstein, and Rabin, 1999), and escalation or de-escalation of commitment (Noda and Bower, 1996). Others consider organization-level effects, such as product complexity (Baldwin and Clark, 2000), organization structures (Argyres, 1996; Gilbert, 2006), and organizational incentives (Henderson and Cockburn, 1994). The innovation impact of product complexity, in particular, presents an interesting theoretical tension. On the one hand, complex designs improve product integration and product performance (Ulrich and Eppinger, 1999). On the other hand, product complexity can undermine intermediate innovation outcomes, including product upgradability (Garud and Kumaraswamy, 1995), maintenance costs (Banker, Davis, and Slaughter, 1998), ease of outsourcing (Schilling and Steensma, 2001), and design and development time (Ulrich, 1995). This suggests that firms producing complex products must trade off between optimizing the performance of current products and pursuing incremental innovations to maintain and improve products for future sales. If customers value current product performance *and* the incremental innovation of new features, product complexity has the unusual theoretical property of affecting customer needs both positively and negatively. Even an organization that advocates customer primacy and strives to meet each need (Christensen and Bower, 1996) will sometimes be forced to fulfill one customer's need ahead of (or instead of) another's. The potential for conflict increases as products become more complex.

Our central assertion is that complexity creates organizational constraints that will alter firms' incentive to be customer-focused, for two reasons. First, changing a complex product creates a cascade of impacts across interdependent units of the firm (Ulrich, 1995). This cascade reduces the likelihood a firm will invest in innovation, especially when changes are hard for engineers to anticipate and coordinate *ex ante*. Prior research suggests that individuals struggle to integrate information across multiple subsets (Dawes, Faust, and Meehl, 1989). Decision makers who seek a common denominator between subsets will still struggle to make choices because individuals often find it difficult to compare across conceptual categories (Thaler, 1985). Firms may be similarly reluctant to pursue complex innovations because (1) information is more difficult to integrate across firm units, and because (2) proposed projects without integrated information will appear more risky to decision makers (Kahneman and Lovallo, 1993; Thaler, 1999).

The second reason complexity creates organizational constraints on incremental innovation is that complexity has path-dependent effects over time. Complex products entail a large number of interdependencies between firm units. Knowledge about interdependencies becomes embedded and obscured over time in firm routines dictating what must (or must not) be done to produce a product (MacDuffie, 1997). Failing to recognize a product's underlying interdependencies makes it more difficult to effect change when change is needed. For example, Winter and Szulanski's (2001: 739) study of Banc One's acquisition process showed that, while a template used to convert acquired banks to Banc One's methods was mostly successful, the firm was forced to reinvent it several times because its components were tacit and poorly documented; when someone critical to the conversion left the firm, knowledge of various interdependencies left with them. Other research has found that knowledge about underlying interdependencies becomes more tacit as products become more complex (Cusumano and Selby, 1998). An increase in tacit knowledge makes it difficult for firms to accurately calculate the benefits and costs of an innovation—a step that is central to the DDI model.

This study examines how firms prioritize customer requests for incremental innovation by combining large sample empirical analyses with detailed qualitative data from interviews. We first test our hypotheses using a panel dataset of incremental innovation decisions made by a single firm in the test and measurement instruments industry. This firm produces and sells a Bluetooth protocol analyzer and makes periodic upgrades to its product in the form of software and hardware changes. Each time the firm considered investing in an innovation requested by a customer, we collected data about two phases of decision making: (1) the decision to innovate (or not) in response to

DOI: 10.1002/smj

a customer request; and, (2) the method of implementation—standardization or customization. We supplemented the econometric results with in-depth interviews of the managers and engineers involved in implementing a customer request. Taken together, our results document how the complicated interactions of individual motivations and organizational constraints respond to product complexity—and systematically undermine a firm's incentive for prioritizing customer requests.

Our principal theoretical contributions are twofold. Our results confirm that complexity has a significant impact on software change decisions, as argued in the existing complexity literature. Innovating firms are buffeted by macroeconomic uncertainties, technological changes, forces of industry structure, and internal organizational constraints, among other forces. Efforts to be customer-focused, as with all managerial decisions, require firms to make trade-offs. Changing products to meet customer requests is especially difficult when products are complex because changes requires buy-in and technological innovation from managers in multiple areas of the firm. Within the complicated process of organization decision making, customer priorities may be diluted or lost entirely.

Our second theoretical contribution is to the innovation literature. Much of the extant literature advocates for the power of incentives or for the primacy of organizational sources (see Henderson, 1993 for an important exception). We adopt a behavioral view of decision making to assert that firm processes for decision making are critical to understanding the motives and objectives in play. We observe that even customer innovation requests with demonstrated market demand (or lack of demand) are not universally embraced (or rejected). This suggests that organizational decision-making processes—rather than market incentives alone—are a critical component of the decisions firms make. We aim to enhance our understanding of firm innovation decisions by exploring why firms sometimes produce decisions that are consistent with the DDI model and other times produce decisions shaped by organizational constraints, including product complexity.

The following section outlines prior research on investment in incremental innovation and sets up the principal hypotheses we examine empirically.

INVESTMENT IN INNOVATION

Systematic research interest in innovation was sparked by Schumpeter (1934), who argued that innovations in the economy—defined as creating new combinations to generate an economic surplus—came from small entrepreneurial firms. In later work, Schumpeter reversed course and argued that large organizations drove innovation (Schumpeter, 1950). Today, his theories of innovation fuel inquiry into both the importance of firm resources and endowments and the role of firm decision-making processes.

The DDI Model

Theories based on market demand attribute innovation to the profit motive or the power of incentives. The DDI model, specifically, contends that innovation and technological change respond to customer demand (Schmookler, 1966). Customer demand is a function of customer preferences. As customers' tastes, income levels, and budget constraints change, so do their preferences. Preference changes will shift the demand curve and promote technological change and innovation. In essence, then, the DDI model argues that changes in demand trigger changes in estimates of market size, which affect a firm's incentive to innovate. This assumes that expectations about market size are positively correlated with profitability, such that customer needs alter market size, and market size alters investment in innovation (Acemoglu and Linn, 2004). The mechanism linking market size to investment in innovation is profitability.

The original formulation of the DDI model applied to all forms of innovation, but subsequent research suggests that market incentives will be muted when an innovation cannibalizes the existing market for the incumbent (Arrow, 1962), when the competence required to invest in the innovation is different from what the incumbent firm possesses (Tushman and Anderson, 1986), and when the innovation alters the product's architecture (Henderson and Clark, 1990). The explanatory power of the DDI model is still presumed to be robust in the case of incremental innovation—defined as an innovation that preserves the market for the existing product (Arrow, 1962). This is because both the economic logic, based on incentives, and the organizational logic, based on inertia, converge to reinforce the dominance of an

DOI: 10.1002/smj

incumbent firm engaging in incremental innovation (see Henderson, 1993). In related work, Christensen and Bower (1996) show that in the wake of a disruptive innovation (i.e., the emergence of new market needs), incumbent firms failed because their organizational processes were designed to satisfy existing customers. Thus, even organizational theories supporting the role of organizational inertia argue that customer demand will spur investment in incremental innovation. The *de facto* hypothesis becomes:

> *Hypothesis 1: All else equal, customer demand for an incremental innovation increases the likelihood a firm will invest in that innovation.*

Complexity and the likelihood of investing in innovation

The implications of complexity have been explored in a variety of business decisions, including innovation (Ethiraj, 2007), product design (Ulrich, 1995), industry structure (Baldwin and Clark, 2000), and make-or-buy (Langlois and Everett, 1992). With respect to the impact of complexity on innovation, several studies document a positive relationship between complexity and incremental innovation, measured in either person hours (Banker *et al*., 1998; Subramanian, Pendharkar, and Wallace, 2006) or the frequency of maintenance work (Kemerer and Slaughter, 1997). In effect, these studies found that complexity increased the time and effort required to innovate. Banker *et al*. (1993) used regression estimates of person hours to estimate the cost function of software maintenance and concluded that complexity poses significant dollar costs.[1] These findings are consistent with the DDI model in that a firm's profitability calculus should include the cost of complexity, but complexity *alone* should have no independent effect on the decision to innovate.

The reason complexity might, in fact, have an independent effect on the decision to innovate is that complexity is known to affect individual behaviors. For instance, Chan (2000) found that increasing complexity also increased the lead time needed to meet maintenance needs. This suggests that complexity might create behavioral aversion among engineers, resulting in delayed attention to tasks. In a related vein, den Besten, Dalle, and Galia (2008) examined the impact of complexity on the degree of collaboration among engineers across a variety of open source projects. They found that higher program complexity reduced participation rates, confirming that complexity might have significant behavioral effects. What is not clear is whether participation falls because (1) complex products have higher interdependencies between modules (Baldwin and Clark, 2000) and are difficult to coordinate simultaneously, or because (2) complexity increases coordination demands on programmers that cannot be managed easily in an open source development environment. What is clear is that complexity changes the cost of innovation *and* has subtle but important behavioral effects. Do those behavioral effects change the likelihood that a firm will engage in incremental innovation? If they do, complexity's *ex ante* behavioral effects may be more relevant than its *ex post* cost implications. If nothing else, exploring the behavioral effects of complexity, and their bearing on firm investments in incremental innovation, introduces an important source of heterogeneity to the study of decision making within and across firms.

At the organization level, prior theory suggests that complexity has at least one important effect on the likelihood of investing in innovation. Increasing interdependencies among various elements of a product makes it more difficult to engage in product innovation (Baldwin and Clark, 2000). In exploring why complexity affects maintenance effort, Banker *et al*. (1998) present complexity as a function of the number of information cues a firm must process, the strength of interdependencies among those information cues, and the changes in the relationships among these information cues over time. These facets of complexity increase managers' cognitive burdens by obscuring their perception and understanding of information cues. These effects will become stronger over time as knowledge about interdependencies becomes embedded in organizational routines (Lippman and Rumelt, 1982), and as product designers leave the firm without documenting product decisions and the interdependencies those decisions created (Winter and Szulanski, 2001). Ultimately, a poor understanding of a product's underlying interdependencies will vastly increase the cognitive effort required to engage in design changes—an

[1] The caveat, however, is that several prior studies have found inconsistent relationships between dollar costs of maintenance and complexity (see Banker *et al*., 1993 for a review of the prior literature).

DOI: 10.1002/smj

effect that has been shown empirically in the study of other organizational outcomes (Novak and Eppinger, 2001; Nutt, 1998; Simonin, 1999).

As interdependence among the elements of a product increases, making improvements to one part may cause significant disruptions for other parts. For example, the open source Internet browser Firefox was developed by a core group of 12 full-time programmers of the Mozilla foundation with the assistance of about 80 code contributors worldwide (Lohr and Markoff, 2004). Firefox is a completely redesigned and rewritten Web browser that, according to anecdotal evidence, overcomes many of the security vulnerabilities of Microsoft Internet Explorer in a more compact package while also offering new features such as tabbed browsing and pop-up blockers. This propelled Firefox to a five percent market share within a few months (Stross, 2004), while Microsoft struggled to add new features to Internet Explorer. The following quotes suggest that Microsoft's innovation efforts were undermined by an increase in interdependence:

> The incipient rise of Firefox, some analysts say, points to an inherent weakness in a fundamental Microsoft business strategy: tying more and more products and features to its monopoly product, the Windows operating system. Internet explorer is tightly bound to Windows, a move that Microsoft says improves the browser's performance.
>
> This strategy, the analysts say, means that *innovation in much of the company's software tends to move in lockstep with Windows development, and that pace has slowed as the operating system has become larger and increasingly complex* (Lohr and Markoff, 2004: C4, emphasis added).

And

> [T]abs are what hooked me, he [Microsoft's director of product management for Windows, Gary Schare] told me, referring to the ability to open within a single window many different Web sites and move easily among them, rather than open separate windows for each one and tax the computer's memory. Firefox has tabs. Other browsers do, too. *But fundamental design decisions for Internet Explorer prevent the addition of this and other desiderata without a thorough update of Windows, which will not be complete until 2006 at the earliest* (Stross, 2004: 3.5, emphasis added).

Thus, we hypothesize that:

Hypothesis 2a: A firm will be less likely to invest in incremental innovation when a product is more complex.

The complexity of change ('change complexity') also has an impact on innovation investment. At the individual level, innovations that are complex—ones that span multiple elements of the product—create a unique set of behavioral challenges. Prior research suggests that individuals find it difficult to integrate information across diverse sources because it is difficult to make information comparable across distinct categories (Dawes *et al*., 1989; Thaler, 1985). Projects without aggregated information will appear to be more risky and thus affect investment decisions. Thaler (2000) conducted an experiment with executives and found that business unit heads were more reluctant than CEOs (chief executive officers) to invest in risky projects because CEOs were privy to information about all projects and were better able to estimate aggregate risks. In addition, cost estimates for complex projects may be inflated because it is hard to estimate the individual efforts required for interdependent tasks (Kahneman and Lovallo, 1993).[2] As a result, individuals are less likely to advocate and pursue innovations that are complex.

At the organization level, incremental innovations spanning multiple product modules are less likely to be pursued because they are less likely to be successful, in part because they demand greater coordination among development teams. Henderson and Clark (1990) found that incumbent firms are less likely to invest in architectural innovation—innovations that span multiple product modules—because organizational communication routines and filters are customized to an existing product. Innovations that span multiple product modules require a deep understanding and

[2] We thank an anonymous reviewer for suggesting this possibility. Indeed this was empirically supported in the data (see Figure 2).

DOI: 10.1002/smj

communication of the underlying interdependencies among organization teams. Knowledge that has not been codified impedes effective innovation even if that knowledge resides somewhere within the organization (Szulanski, 1996). Hansen (1999) confirmed the increased coordination burden associated with complexity when he found that complex knowledge cannot be transferred without strong ties between individuals. By implication, implementing innovations that span multiple product modules will demand strong ties between product teams. When strong ties do not exist, innovating across modules becomes less likely.

Given the role of organization structures in mitigating complexity, a key question is whether strong ties are likely to exist between product teams. When product teams are predicated on the logic of autonomy and independence, their mandate is to pursue innovations within the bounds of the responsibility assigned to them. Autonomy reduces the possibility of cross-team innovation initiatives, but it also mitigates the cost of repeated and ongoing coordination, which would be required to manage tightly interdependent teams. Organization structures in software firms specifically, are often designed explicitly to maximize autonomy and limit reciprocal interdependencies, such that product teams have weak ties by design (Cusumano and Selby, 1998; Thompson, 1967). This will create a strong incentive to pursue innovations that lie within each team's purview and to ignore innovations that span multiple teams. Thus, we hypothesize that:

Hypothesis 2b: A firm will be less likely to invest in incremental innovation when a proposed change is more complex.

In sum, the central assertion of this paper is that product complexity and change complexity can constrain investment in incremental innovation. The following section describes the setting in which we test our hypotheses.

BLUETOOTH PROTOCOL ANALYZER

The product we study here is a 'Bluetooth protocol analyzer.' Bluetooth is a short-range wireless communication standard that allows connectivity between electronic equipment using frequencies in the 2.45 GHz range. Bluetooth-enabled devices communicate with each other using low-powered signals and a language of commands and responses, known as protocols, specified in the Bluetooth Standard. Individual Bluetooth devices can be wirelessly connected to each other to form a 'personal area network' of electronic devices. For example, a personal area network could consist of a Bluetooth-enabled mobile phone and a wireless headset.

The Bluetooth protocol analyzer is used to test and measure Bluetooth signals during the production of Bluetooth consumer products. Thus, typical customers of the Bluetooth protocol analyzer are not end-users themselves but the firms producing Bluetooth-enabled consumer products. Firms producing Bluetooth-enabled devices need to test the interoperability of their devices with others that could be used in the end-consumer's personal area network.

The Bluetooth protocol analyzer includes a hardware component and a software component. The hardware component includes an air-sniffing probe to intercept wireless communication traffic between Bluetooth devices, a cable-sniffing interface to intercept wired communication traffic, and a serial and parallel port interface for connection with a personal computer (PC). The software component involves firmware stored in the memory of the hardware component and a packaged software product for installation on the client's PC. The physical hardware and the firmware stored in the protocol analyzer intercept data moving between Bluetooth devices and pass it to the client software package for further analysis. Captured data is mapped to the typical levels recommended by the Bluetooth standard and subjected to statistical quality analysis. The software package helps define the specifications and configures Bluetooth devices used in the testing procedure, controls the data sources, represents results in a graphical manner, and prepares testing reports.

Evolution of the Bluetooth protocol analyzer

We examine the evolution of client software for protocol analyzers to understand the impact of customer demand and complexity on the decision to innovate. The evolution of protocol analyzer software depends on two categories of actions: changes that are involuntary and changes that are voluntary.

DOI: 10.1002/smj

Involuntary changes in protocol analyzer software are triggered by one or several of three sources. First, changes in Bluetooth standards trigger changes in hardware, software, or both. The Bluetooth standard is governed by a Bluetooth special interest group of more than 1,000 electronic equipment manufacturers. Changes to the Bluetooth standard must be incorporated in the next release of a protocol analyzer, so changes in standards almost always necessitate changes to protocol analyzer client software. Second, hardware component changes can trigger software changes. Protocol analyzer hardware includes semiconductor components. When design and process improvements in the semiconductor industry allow allied industries to reduce hardware costs, the protocol analyzer industry sometimes updates its hardware, firmware, and client software to exploit those cost reductions. Finally, changes to the operating system (OS) software on which the client software runs will necessitate changes to protocol analyzer software. The Bluetooth protocol analyzer packaged software component is designed to run on the Windows and Solaris operating systems. The Bluetooth protocol analyzer software component interacts with the OS using application programming interfaces (APIs). As newer operating systems are released, the APIs they support are upgraded and the protocol analyzer software must be modified. Involuntary changes are not the focus of this study, but we do control for such triggers.

Voluntary changes in protocol analyzer software

Studying voluntary changes to the protocol analyzer software allows us to observe the managerial decisions those changes reflect. Protocol analyzers are essential equipment for consumer electronics manufacturers. As firms build consumer products based on evolving Bluetooth standards, they need reliable equipment to test the basic functioning and interoperability of their products. Often, the diagnostic data required by different firms may be different. Consequently, the standardized protocol analyzer client software usually meets only the most common needs. When firms encounter specific needs that are not implemented in the software, they initiate a request to the protocol analyzer manufacturer to incorporate the feature they want. Customer requests for new product features allow us to examine the circumstances under which firms implement incremental innovations to their software (von Hippel, 1976). Meeting requests may require changes in hardware, changes in firmware, and/or changes in client software. There is a discretionary and irreversible research and development (R&D) cost associated with meeting each request. Thus, in the decision to meet a customer request we can measure the cost and benefit of fulfillment. Requests that are fulfilled may be implemented with a customized delivery to the client making the request or by incorporating the change into the software's next release.

This context allows us to measure and track the evolution of the software product and also to observe discrete decisions about product enhancements over time. These decisions may be consistent with a cost-benefit calculus (i.e., DDI) or they may systematically deviate from it. Variations along the cost-benefit dimension, coupled with variations in request fulfillment, allow us to tease out the impact of complexity on the decision to innovate.

The context of protocol analyzer software is also attractive because it allows us to control for, or rule out, several competing explanations. First, there is not much uncertainty about the cost-benefit calculus. It has been well documented that the total lifetime cost of maintaining a software program is 40 percent or more of the cost of developing it, and that fixing a defect has a 20–50 percent chance of introducing another defect (Brooks, 1995: 121–122). Other studies show that while the number of modules increased linearly with every incremental release of IBM's OS/360 (a large operating system to run a mainframe computer), the number of software modules modified increased exponentially with each release (Krishnan, Mukhopadhyay, and Kriebel, 2004). Thus, unlike, for example, the disk-drive industry (Christensen and Bower, 1996), uncertainty about performance trajectories relative to costs is not a significant concern for protocol analyzer software.

Second, while there is evidence of learning curves in software development, the relevant learning in our setting is at the level of the individual. Prior research has shown that software productivity has strong individual-level effects (Fong Boh, Slaughter, and Espinosa, 2007). Learning can be an important alternative explanation if a firm acquires a new software product and staffs a new development team to work on it. This is not the case

DOI: 10.1002/smj

in our empirical setting. The firm featured in our research developed its product from scratch and performed in-house maintenance throughout the versions we study. Thus, large-scale changes in maintenance personnel did not occur. In addition, the presence of public and open standards in the Bluetooth product removes a large source of firm-specific knowledge that is typically acquired in learning-by-doing.

Finally, the Bluetooth protocol analyzer is a standalone product whose only complement is the computer on which the client software runs (typically Unix and Windows). Since changes to the OS have been less frequent than changes to the protocol analyzer software, we are less concerned about the effect of complementarities as an alternative explanation (Schilling, 2003). We do control for the size of the installed base of protocol analyzer software.

LARGE SAMPLE STUDY

Sample and data

We collected data from a leading test and measurement firm that, in 2004, operated in 20 countries, recorded over $1.5 billion in sales, and held 675 active technology patents. Using data from one firm severely limits the generalizability of the study, but the rich data presented here is difficult (if not impossible) to obtain across a wider set of firms. Moreover, using data from a single firm is not unique to this paper. There is a long history in the management journals of publishing papers with single firm data, be they case studies (Doz, 1996; Marginson, 2002) or large sample disaggregated data from a single firm (see e.g., Dencker, 2009; Ethiraj *et al*., 2005; Huckman, Staats, and Upton, 2009; Sinclair, Klepper, and Cohen, 2000; Ton and Huckman, 2008).

Because we cannot identify the firm for confidentiality reasons, we refer to it here as 'Measuretronics.' Measuretronics invests about one third of its revenues in R&D across five businesses, including oscilloscopes, logic analyzers, video test products, telecommunications equipment, and optical sensor products. The firm has about 400 customers in various industries, including silicon vendors, electronic standards compliance testers, communication protocol stack developers, communication equipment manufacturers, and telecommunication application developers. Figure 1 presents Measuretronics's organization chart.

The protocol analyzer client software spanned a life cycle of 28 versions in a five-year period.

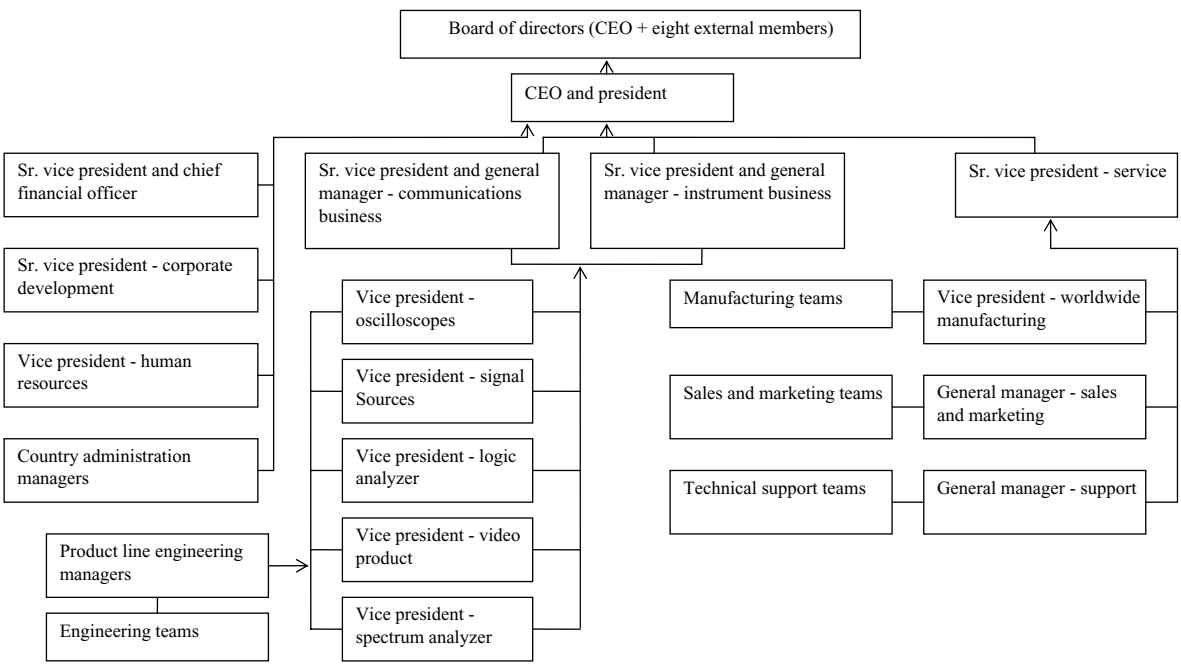

Figure 1. Measuretronics organization chart

DOI: 10.1002/smj

The data collection for this project covered all 28 versions and involved an in-depth exploration of the functional features and complexity of the product and the firm's product development process. It included interviews with product managers, program developers, customers, and Bluetooth domain experts. In all, the efforts required to gather the data took three years.

Decisions to implement incremental innovations in protocol analyzer software at Measuretronics involved three major steps. The first step involved collection and consolidation of customer requests. The firm uses an online system called 'customer support network' to log all customer requests and feedback, and each request is assigned a unique ticket number. The second step is a thorough analysis of the feature request tickets. This involves removing duplicate tickets, grouping tickets from different customers that have similar content, and validating the content of each customer ticket. During the validation process, a product management specialist checks if the requested feature is already present in the released product versions. If the requested feature is not available in the current product, the ticket is scheduled for further action.

In the last step of the innovation decision process, the product management team works with the development team to analyze the technical feasibility of a change request, checks if the requested feature is shipped by a competitor, and generates a customer survey. The customer survey is designed to obtain installed-base feedback on the proposed feature. A survey e-mailed to existing customers of the Bluetooth protocol analyzer solicits feedback on the proposed feature (or change). To discourage casual feedback or nonresponse, customers are also informed of the impact of the proposed change in terms of code modifications and future service-pack installations. When the opinions of the installed base have been consolidated, the firm makes a final decision to fulfill or reject the requested feature (or change).

When we completed our data collection, 203 customer-generated product requests had been submitted. We filtered duplicate requests with assistance from the product management team at the research site, resulting in 152 unique customer requests. Of those, 120 (79%) customer requests were fulfilled (or completely processed) and 32 (21%) were not fulfilled. The important issue for this study is whether the customer requests required incremental innovation. Our analysis indicated that a bulk of the customer requests qualified as incremental innovations because they entailed feature requests. A few were requests for bug fixes, which we did not count as incremental innovation because they indicate the failure of promised features. Table 1 provides a sample of customer requests categorized into incremental innovations and bug fixes, respectively. We included all customer requests in our analyses in order not to bias the sample, and controlled for bug-fix-type requests in the empirical analysis.

The 120 fulfilled requests were separated into two groups: those that were integrated into the next packaged release (92), and those that were pursued as customization projects through contractual relationships with the customer (28).

MEASURES

Dependent variables

We examine whether complexity affects incremental innovation decisions, which entails measuring incremental innovation. The empirical literature measuring incremental innovation has adopted two distinct approaches summarized here as input-side or output-side. Input-side measures track the expenditure on innovations such as R&D dollars or patents (e.g., Schmookler, 1966). Output-side measures track the surplus associated with the creation of new combinations (e.g., Trajtenberg, 1990). In this study, we employ two proxies for incremental innovation that are consistent with output-side measures. Note that all of these innovations were incremental in that they involved changes to an existing product. A radical innovation, in contrast, involves an entirely new product, which in the context of software would be a new code base (Henderson, 1993). Our two innovation measures are:

Fulfilled. Coded 1 if the feature requested by the customer was fulfilled, and 0 otherwise. This measure fulfills the criteria from Schumpeter (1934) in that any feature added to the existing product is clearly a new combination.

Standardized. A fulfilled request was either standardized into the next release or customized for the customer making the request. This indicator variable is coded as 1 if the customer request was standardized, and 0 otherwise. This measure reflects an output-side measure because a new combination has been implemented. While

DOI: 10.1002/smj

Table 1. Classification of customer requests

Change request	Nature of request
The application should support standard Windows XP UI conventions and keyboard shortcuts	Bug fix
The application should support printing (similar to MS Office applications)	Bug fix
The application may provide 'TIP OF THE DAY' whenever it is started, similar to Microsoft Visual Studio	Bug fix
The application shall support hiding of columns based on right mouse click on the column header	Bug fix
Add/remove Bookmark option shall enable the user to toggle the bookmark on or off in the leftmost column in the Packet List window for the row highlighted	Bug fix
File/Open dialog shall support viewing of file comments of the selected file in the comment field. The user will not be able to edit the comment field	Bug fix
When 'start capture' button is pressed, it should turn to green color	Bug fix
Error packet generator should allow users to delete all error sequences	Bug fix
Toggle Hex-ASCII in Payload window shall allow the user to switch the display of the payload data for the highlighted packet in the Packet Analyzer grid window between the hexadecimal format and the ASCII format	Feature
The application should be a Windows Multi-Document Interface application (i.e., be able to display multiple windows from application at same time with each view having its own menus and sub-menus allowing users to switch between the different windows)	Feature
The application should support new packet-level displays including Higher Level Data Link Control (HDLC), and Point-to-Point protocol (PPP)	Feature
The application should support search capability (text search). This should allow the user to find specific packet types, protocol messages, and connected hosts	Feature
The application should provide a new synchronization wizard to help users connect between their devices listed in user profiles automatically, showing connection status, and any errors to user at start-up	Feature
The application may support page references for the standards specifications. i.e., if the user clicks on an element on the info panel, the application may inform about the page in the Bluetooth specification corresponding to that element or show the Bluetooth specifications information in the online help	Feature
Users should be able to extract 'voice data' packets from the log files created	Feature
The application should provide profile support for hands' free profile, headset profile, synchronization, and basic printing	Feature
The application should provide a drift-compensator that allows calculation of the unit drift necessary for synchronization of FHS packets	Feature
The application should include a new test program interface and scripting language to help users to automate their testing procedures	Feature
The application should provide a high-resolution time stamp	Feature
The application should provide RF parametrics like received signal strength indicator	Feature

customization only generates returns from the customer who requested the feature, standardization diffuses the innovation across the customer base.

Independent variables

Demand

We employed three proxies to capture the effect of customer demand on the cost-benefit trade-off (i.e., profitability) inherent in any investment decision. We sought to capture the value of the proposed feature to both the installed base and to the customer that made the request. Together, the two proxies approximate the expected revenues from implementing the feature. We used a third proxy—estimated cost—to capture the expected cost of implementing the feature.

Installed-base value

For every change request received, the firm sends a survey to all existing customers of the product. They are asked to indicate whether the proposed feature will be useful to them. We measured installed-base value for the proposed feature as the percentage of customers who indicated a new feature would be useful. This variable ranges from 0–100.

DOI: 10.1002/smj

Customer importance

A customer initiating a feature request is required to indicate its importance. This variable is coded on a Likert scale ranging from 1–5 and measures perceived importance of the proposed feature.[3] Higher values reflect greater importance to the customer.

Estimated cost (000s)

For each request the vendor firm received, it generated an estimate of the cost of implementing the feature. The estimate includes the direct cost of manpower and the time required to implement the feature and test a new version before rollout.

COMPLEXITY

We constructed two proxies to measure product complexity at the time the request was received, and three proxies to measure the complexity of the requested change.

Product complexity

Interdependencies

In practice, it is extremely difficult to measure the number of interdependencies within a software product. Typically, however, the number of interdependencies is related to software size (Jones, 2000). We collected four different measures of software size at the time the request was received: number of lines of code, number of functions, number of objects, and number of classes. All four measures were highly correlated (>0.95), which precluded using all of them. We also noticed that the more specialized object-oriented dependency metrics, such as coupling between objects, and the depth of inheritance measures correlated highly with the size measures and increased linearly with the number of classes in the system. Hence, we used the number of classes to estimate the level of interdependencies in the software product. A class represents the highest level of aggregation in object-oriented programming and is typically defined as a cohesive package that encapsulates a set of variables, functions, and objects (Booch, 1994). Our results are robust to the use of any of the four proxies for interdependence.

Product age

As the code base of software ages, it undergoes iterations that increase its complexity. We measured product age using the version number of the software (1–28) on the date of the customer request. As outlined in Hypothesis 2a, we expect product complexity to increase with the software version number.

Change complexity

Nature of change

Measuretronics characterized the magnitude of change required to the existing code base. A customer request was considered a minor change when only one module was affected, and a major change when modifications affected two or more modules. While changes spanning multiple modules do not necessarily span multiple organizational subunits, prior research suggests that organization structures tend to mirror product architectures (Henderson and Clark, 1990). This variable is coded 1 if the change was major.

Objects modified

For the subset of customer requests that were fulfilled, we collected data on the number of objects that were modified. In object-oriented programming, an object is a self-contained functional entity that consists of both data and procedures to manipulate the data (Booch, 1994). This variable captures the number of objects that were modified to

[3] The codes and the description of the evaluation were as follows: 1 (The change is preferred by the customer but is not essential for day-to-day productive operations of the customer. The customer does not intend to be a test partner for implementing the change); 2 (The change is preferred by the customer but is not essential for day-to-day productive operations. However the customer agrees to be a test partner for implementing the change); 3 (The change is needed for the day-to-day operations. The customer is willing to accept roundabout or alternate ways of implementing the feature and does not intend to be a pilot and test partner for the change. The customer cannot allocate resources to simulate a production environment); 4 (The change is needed for the day-to-day operations of the customer. The customer is NOT willing to accept roundabout or alternate ways of implementing the feature but can wait until the feature is released in future versions. The customer is willing to simulate a test production environment); and, 5 (The feature requested is essential and time critical. The customer is willing to allocate resources for piloting. The customer might stop using the vendor's product if the feature is not made available).

DOI: 10.1002/smj

implement the change request. The larger the number of objects modified, the greater the complexity of the request. Because we have this data only for fulfilled projects, it is included in only part of the estimation (details follow in the next subsection).

Intermediate releases

For the subset of requests that were fulfilled, we obtained the number of internal intermediate releases before the change was finalized. A larger number of intermediate releases reflect greater change complexity.

Other controls

Standard violation

Upon receiving a customer request for a new feature, Measuretronics verifies whether the request violates the Bluetooth protocol standard. Standard violations alone are not cause for ignoring a request. However, our interviews suggested that standard violations pose fundamental software conflicts because they increase the number of modules impacted by a proposed change. Thus, we employed an indicator variable coded 1 if the customer request violated the Bluetooth protocol in use when the request was made.

Version conflict

For every customer request received, Measuretronics also assesses whether the proposed feature will cause incompatibility with previous releases of the product (violating backward compatibility). Implementing features that violate backward compatibility are a function of a change to the OS or a change in the use of APIs. Under either condition, the likelihood of version conflict is amplified. We created an indicator variable coded 1 if a proposed change required changes that were incompatible with previous public releases of the product.

Competition

A long-standing assertion in the empirical literature on innovation is the link between innovation and market structure (see Kamien and Schwartz, 1982 for a review). We controlled for the effect of competition on the incentive to implement customer requests using an indicator variable coded 1 if a competitor firm offered the feature requested by the customer.

Market share

An alternative explanation for dampened incentive for innovation is externalities (David, 1985). The disruptive effects of innovations are amplified when a firm has a larger installed base. We sought to control for such externalities using the vendor's market share of the Bluetooth protocol analyzer product category (at the time the request was received) as a proxy.

Features bundled

We collected data on the number of features that were bundled into a fulfilled request. We included this control because the number of features built into a request can affect the incentive to standardize or customize request fulfillment. The greater the number of features built into a request, the greater the likelihood that a larger proportion of the installed base of customers would find it useful, thus creating an impetus for standardization.

Time constraint

We sought to measure the time gap in days between the date the request was logged and the scheduled release date of the next version of the product. Software releases are often made on predetermined schedules (Cusumano and Selby, 1998), and the time pressure to complete scheduled modifications increases as the next scheduled release date approaches. This is likely to depress the incentive to fulfill requests or to standardize them.

ANALYSIS

We performed the analysis in two stages. In the first stage, we estimated a binary choice model of the likelihood of customer request fulfillment. We included three sets of predictors—demand, complexity, and other controls to account for alternative explanations. Thus, the first stage probit equation was,

$$P(z_i = 1) = \int_{-\infty}^{\beta m_i + \delta c_i + \tau w_i} \phi(u_1) du_1 = \Phi(\beta m_i + \delta \& c_i + \tau w_i) \quad (1)$$

DOI: 10.1002/smj

where, z_i, is the fulfillment of request i generated by customer, j, m_i, is a vector of demand covariates, c_i, a vector of complexity covariates, w_i a vector of controls, and $u_1 \sim \mathrm{N}(0, 1)$, is the standardized normal variable. Since multiple requests may be generated by the same customer, we adjusted the estimated standard errors for the resultant heteroskedasticity.

In the second stage, we estimated the likelihood that a customer request would be implemented using standardization (or customization). The standardization (or customization) of a request is observed only if the request is fulfilled. Thus, the likelihood of standardization is given by the following equation,

$$\begin{aligned} P(s_i = 1) &= P(s_i = \mathbf{1}|z_i > 0) \\ &= \int_{-\infty}^{\vartheta m_i + \eta c_i + \kappa w_i} \phi(u_2) du_2 \\ &= \Phi(\vartheta m_i + \eta c_i + \kappa w_i) \end{aligned} \quad (2)$$

Because standardization, s_i, is observed only when a request is fulfilled, it creates a nonzero correlation between u_1 and u_2 in the two equations. In this case, a simple probit estimation of Equation 2 yields biased estimates of the coefficients (Maddala, 1983). To correct for this selection bias, we estimated from Equation 1 the hazard, h_i, for each observation, as explained below.

Let x_i denote a vector of all covariates included in Equation 1 and $\hat{\gamma}$ represent a vector of corresponding estimated coefficients. Then, the hazard, h_i, for each observation is,

$$h_i = \begin{cases} \phi(x_i\hat{\gamma})/\Phi(x_i\hat{\gamma}), & z_i = 1 \\ -\phi(x_i\hat{\gamma})/\{1 - \Phi(x_i\hat{\gamma})\}, & z_i = 0 \end{cases}$$

Equation 2 was then augmented with the hazard, h, estimated in the first stage,

$$\begin{aligned} P(s_i = 1|z_i > 0) &= \int_{-\infty}^{\vartheta m_i + \eta c_i + \kappa w_i + \rho\sigma h_i} \phi(u_2) du_2 \\ &= \Phi(\vartheta m_i + \eta c_i + \kappa w_i + \rho\sigma h_i) \end{aligned} \quad (3)$$

Because the covariance of the error terms are nonzero, Equation 3 is identified only when there is at least one variable in Equation 3 that is not included in Equation 1 (Maddala, 1983: 120). We used two variables not included in Equation 1—intermediate releases and features bundled—to identify Equation 3. We again employed probit estimation for Equation 3 and corrected the estimated standard errors for heteroskedasticity due to multiple requests from the same customer.

Large sample study results

Table 2 presents the descriptive statistics and the correlation matrix of variables employed in the estimation. The one caveat in interpreting the correlations in Columns (1) and (2) of Table 2 is that the two dependent variables, fulfilled and standardized, are categorical variables. They are reported here merely to provide an indication of the direction of the relationship. The correlation matrix, seen from the negative signs on the correlation coefficients of the complexity measures with the first dependent variable (fulfilled), lends support to the argument that complexity affects innovation decisions.

We offered two microfoundations for our theory explaining the link between complexity and the likelihood of investing in innovation. First, at the individual level, we argued that more complex projects will generate higher variance in the cost estimates, which will impede the use of the cost-benefit calculus. We divided completed projects into two groups based on complexity and examined the relationship between estimated completion time (a proxy for cost) and actual completion time. We plot this in Figure 2. We see that the cost estimates for less complex projects are fairly accurate and the confidence interval is relatively narrow. In contrast, the costs for more complex projects are more likely to be underestimated and, furthermore, their confidence interval is very large. With a large confidence interval, the use of a cost-benefit calculus in decision making is prone to error. This confirms the face validity of the theory that it is more difficult to estimate costs for highly complex projects.

Second, at the organization level, we argued that product complexity increases with time. Given the normal personnel turnover associated with R&D teams, we predicted an increase in product complexity as the knowledge behind interdependencies becomes tacit. More tacit knowledge will weaken the understanding of cause-effect relationships and hamper the implementation of changes. While we do not have a direct measure of tacit knowledge,

DOI: 10.1002/smj

Table 2. Means, SD, and correlation matrix of variables employed in estimation (N=152)

	Variable	Mean	SD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	Fulfilled	0.779	0.416	1.00															
2	Standardized	0.767	0.425	—	1.00														
3	Installed-base value	42.597	26.9	0.41	−0.24	1.00													
4	Customer importance	3.948	0.955	0.29	0.18	0.45	1.00												
5	Estimated cost (000s)	244.381	164.692	0.04	0.15	0.16	0.30	1.00											
6	Interdependencies	170.656	69.385	−0.41	0.34	−0.24	0.06	0.09	1.00										
7	Standard violation	0.292	0.456	−0.31	−0.67	0.01	−0.13	−0.17	−0.01	1.00									
8	Nature of change	0.409	0.493	−0.56	0.24	−0.25	0.10	0.34	0.39	0.04	1.00								
9	Product age	12.682	7.803	−0.50	0.44	−0.28	0.01	0.12	0.83	0.01	0.44	1.00							
10	Version conflict	0.481	0.501	−0.21	−0.27	−0.22	−0.29	−0.18	0.06	0.37	−0.07	0.01	1.00						
11	Competition	0.455	0.5	0.24	0.22	0.25	0.30	0.19	−0.16	−0.29	0.06	−0.12	−0.78	1.00					
12	Time constraint	202.559	172.851	−0.37	0.35	−0.30	0.10	0.05	0.44	0.05	0.40	0.37	0.06	−0.04	1.00				
13	Market share	26.078	20.784	−0.62	0.25	−0.33	−0.07	0.03	0.74	0.09	0.46	0.87	0.08	−0.17	0.38	1.00			
14	Features bundled	3.338	6.846	0.25	0.20	0.12	0.12	−0.05	−0.21	−0.21	−0.16	−0.35	−0.01	0.04	−0.11	−0.33	1.00		
15	Intermediate releases	18.416	16.452	0.59	0.26	0.02	0.23	0.05	0.15	−0.31	−0.33	0.06	−0.09	0.10	0.08	0.01	0.04	1.00	
16	Objects modified	63.87	65.085	0.01	−0.40	−0.05	0.04	−0.15	−0.24	0.20	0.02	−0.20	−0.13	0.21	0.01	−0.12	−0.14	−0.06	1.00

Note: Correlations greater than 0.16 are significant at 5% or less.

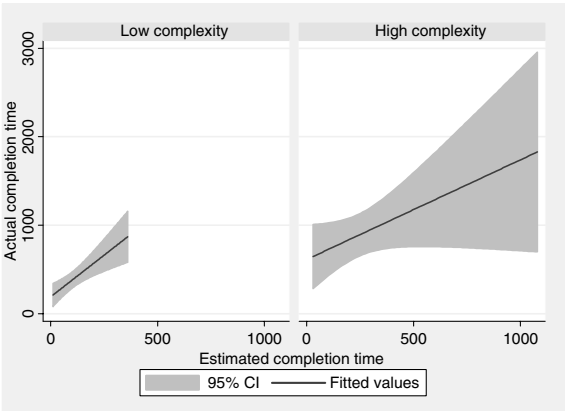

Figure 2. Graphing actual project completion time on estimated time by nature of change

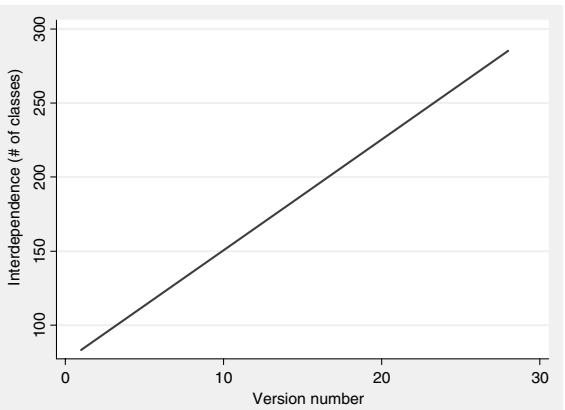

Figure 3. Plot of interdependence over product versions

we do track interdependence over time. Figure 3 plots the level of interdependence over the 28 versions of the software for which we have data. We see from Figure 3 that interdependence grows linearly as the product ages. This again constitutes evidence consistent with our theory.

Table 3 presents the results of the bivariate probit estimation. Column (1) presents the DDI model predicting the likelihood of fulfilling a customer request for a feature. Of the three variables that reflect the DDI model, only customer importance was marginally significant ($p <= 0.10$). The importance of the feature to the installed base and its estimated cost had no significant effect on the likelihood of fulfilling the customer request. Thus, there appears to be little empirical support for the DDI hypothesis, Hypothesis 1. The controls for alternative explanations were all significant and in the expected direction.

Columns (2) and (3) present the results of the complexity model predicting the likelihood of fulfilling a customer request for a feature. The difference between the two specifications is the inclusion of interdependencies in the Column (3) model. In Column (2), both complexity measures are negative and significant, as predicted. Increases in product age reduce the likelihood of fulfilling a customer request, providing support for Hypothesis 2a. In addition, as the nature of change spans multiple modules, the likelihood of fulfilling the request declines, providing support for Hypothesis 2b.

The specification in Column (3) includes interdependencies. The coefficient on interdependencies is negative and highly significant, suggesting that increasing product interdependencies is negatively related to the likelihood of investing in innovation. This provides additional support for Hypothesis 2a. Whereas all the other variables continue to remain unchanged from the specification in Column (2), product age becomes nonsignificant in Column (3) when product interdependence is included in the specification. Product age proxies many organizational pathologies, including inertia. Remarkably, interdependence swamps the effects of age. We explored the possibility that complexity mediates the relationship between age and the likelihood of investing in innovation. The causal test for mediation outlined in Baron and Kenny (1986) was completely met, that is, (1) product age is statistically significant in the absence of interdependencies; (2) regression of interdependencies on product age yielded a statistically significant coefficient (7.492; $p<0.000$); (3) interdependencies is statistically significant; and, (4) product age turns nonsignificant. In addition, following the procedures outlined in MacKinnon *et al*. (2002), we performed a series of product of coefficient tests for mediation and found that the difference was highly significant in all cases ($z=>10.07$; $p<0.000$). This suggests that complexity *is* perhaps the reason for inertia.

Model (4) estimates the DDI and complexity models together. The DDI variables are all nonsignificant. The complexity variables—interdependencies and nature of change—continue to be negative and significant. The sign and significance of the other controls are similar to those in Model (3). The mediation effect continues to be robust. In terms of the marginal effects, at the mean level of interdependence, the probability of fulfilling a request decreased by 0.19. For one standard

DOI: 10.1002/smj

Table 3. Bivariate probit estimates

	Hyp. sign	(1) Fulfilled	(2) Fulfilled	(3) Fulfilled	(4) Fulfilled	(5) Standardized
Product complexity						
Interdependencies (H2a)	−			−0.013*	−0.014*	−0.021*
				(0.006)	(0.006)	(0.009)
Product age (H2a)	−		−0.098*	0.034	0.048	0.251*
			(0.049)	(0.069)	(0.082)	(0.103)
Change complexity						
Nature of change (H2b)	−		−2.103**	−2.270**	−2.180**	3.507**
			(0.342)	(0.400)	(0.259)	(0.917)
Objects modified (H2b)	−					−0.027**
						(0.007)
Intermediate releases (H2b)	−					−0.023
						(0.024)
Market incentives						
Installed-base value (H1)	+	0.017			0.012	0.060**
		(0.011)			(0.012)	(0.018)
Customer importance (H1)	+	0.455+			0.343	−0.055
		(0.276)			(0.227)	(0.315)
Estimated cost (H1)	−	−0.000			−0.000	−0.009**
		(0.001)			(0.000)	(0.003)
Other controls						
Market share		−0.045**	−0.065**	−0.068**	−0.061**	0.158**
		(0.007)	(0.018)	(0.019)	(0.021)	(0.058)
Competition		0.329	0.935*	1.056*	0.844*	1.228**
		(0.338)	(0.344)	(0.342)	(0.328)	(0.459)
Standard violation		−1.15**	−1.138*	−0.906*	−0.809*	
		(0.429)	(0.601)	(0.410)	(0.367)	
Version conflict		0.056	−0.056	−0.315	−0.381	−3.400**
		(0.386)	(0.497)	(0.487)	(0.477)	(0.984)
Time constraint		−0.002*	−0.000	−0.001	−0.001	−0.001
		(0.001)	(0.001)	(0.001)	(0.001)	(0.002)
Features bundled						0.494**
						(0.135)
Mills ratio						8.614**
						(2.645)
Constant		0.896	3.028**	2.216*	0.931	4.461**
		(0.640)	(0.781)	(0.725)	(0.678)	(1.638)
Observations		152	152	152	152	120
Wald chi^2		85.16**	121.36**	105.23**	308.80**	196.61**
Pseudo R^2		0.592	0.638	0.655	0.699	0.810

Robust standard errors in parentheses.
+ significant at 10%; * significant at 5%; ** significant at 1%.

deviation above the mean level of interdependence, the probability of fulfilling a request decreased by 0.27. The probability of fulfilling a customer request for a complex change was about 0.17 lower than that for less complex changes. This provides strong support for Hypotheses 2a and 2b. Hypothesis 1 was uniformly unsupported.

Finally, Model (5) presents the estimates of the likelihood of standardizing a customer request into future versions of the software. Because the likelihood of standardizing (or customizing) a customer request is conditional on the likelihood of fulfilling a request, we condition the estimation of this equation on the likelihood of fulfilling a request. We identified this equation using three variables not included in the fulfillment equation: objects modified, intermediate releases, and features bundled. The Mills ratio is positive and significant, suggesting that the standardization decision is indeed conditional on the fulfillment

DOI: 10.1002/smj

decision.[4] In contrast to the results in Model (4), the DDI variables are significant in explaining the decision to standardize a request. At the mean value of a feature to the installed base, the likelihood of standardizing a request increases by 0.0045. However, the importance that a customer attaches to a request is not a significant predictor of the decision to standardize or customize the request. At mean levels of estimated cost for implementing a request, the likelihood that a request will be customized decreases by 0.0035. This is consistent with our expectations. With the importance to the installed base held constant, a feature is more likely to be customized if the added cost of standardizing cannot be recovered from the installed base. This is because price is uniform for the standardized version of the software but customization generates additional revenue from the customer that requested the feature. Thus, if the expected increase in unit revenue from the new feature times the installed base is less than the cost of implementing the feature, the firm chooses to customize rather than standardize. This is strongly supportive of the DDI argument.

We included five complexity measures in Model (5). We dropped standard violation from the equation because it perfectly predicts customization. Both product-complexity measures and two of the three change-complexity measures were statistically significant in explaining standardization. At mean levels of interdependence, the probability of standardization of the customer request decreases by 0.005. At mean levels of product age, the probability of standardization increased by 0.0004. For complex projects, the likelihood of standardization increases by 0.0054. Finally, at mean levels of objects modified, the probability of standardization decreases by 0.0027.

Prima facie, the contrasting results for two of the complexity variables (product age and nature of change) seem at odds with our expectations. In fact, the effect of increasing complexity is reflected in version conflict and the number of objects modified, both of which increase the likelihood of customization. These two variables held constant, major changes and an increase in product age increase the likelihood of standardization. Upon talking with the development engineers, we understood that as the change spans multiple modules, it is more efficient to implement the change in the primary code base than to maintain distinct code sets corresponding to different customer requests. Finally, the sign and significance of the other controls were as expected.

In summary, the large sample empirical analyses suggest that in the case of software products, complexity is indeed an important driver of the decision to invest resources in incremental innovation. We find that whereas customer demand is an important predictor of the decision to *standardize* a customer request, it has little predictive power in the decision to *fulfill* a request. The important unanswered question at the conclusion of the empirical analyses is what accounts for the observed results. What kinds of managerial and/or organizational decision processes explain these empirical patterns? The qualitative study in the following section examines this question.

SMALL SAMPLE QUALITATIVE STUDY

Sample and data

The qualitative component of the study involved the heads of product development, product management, and marketing, and also some programmers and sales staff. We interviewed a total of six persons in five rounds over a four-month period. The interviews were semi-structured, lasted 30 to 45 minutes each, and were recorded. The first round of interviews was open-ended and focused broadly on the responsibilities of the individuals, what kinds of work they perform, how their goals are set, what kinds of projects they pursue, and so on. Once this round of interviews was completed, we transcribed the audio recordings and analyzed the transcripts in an effort to understand the pattern observed in the large sample empirical analyses. While we obtained some answers, this review often raised other questions, prompting a new round of interviews with the same individuals. We systematically narrowed our focus in each round and terminated the interviews in the fifth round when we saw that most of our questions had been answered and that no new information was being revealed. Interview transcripts covered 40 typed, single-spaced pages, which form the basis for our analyses and inference.

[4] We also tested for normality of the probit selection equation. The quantile plot of the predicted probit scores against the normal distribution was linear as expected.

DOI: 10.1002/smj

The main focus of our qualitative study was to understand the innovation decision-making process and how it might account for the observed results. Starting from the decision-making process, our probes radiated across other linked and related issues that emerged. We began by identifying the groups responsible for the fulfillment and standardization decisions. We found that the product management group administered the management process for customer feature requests in close consultation with the product development group. Once the product development group agreed to fulfill a customer request, the product management group involved the marketing group in deciding if the request would be met with standardization or customization. The separation of decision responsibilities between product development and marketing provides a baseline intuition for why the product development group might be focused on complexity or technical characteristics of the product and the marketing group on the profitability of the product. However, this intuition may be rooted in a variety of underlying causes. Our interviews focused on uncovering potential causes and how they fit together to produce the empirical pattern that showed up in our large sample analyses.

Qualitative study results

The R&D budget allocation in Measuretronics happened in a top-down fashion. The technical board of the company decided on priority areas for research and provided financial resources and specific investment instructions to the division heads. This practice is somewhat in contrast to what Noda and Bower (1996) found in their study of investment decisions at Bellsouth, where the top management showed low interest in technological decisions. The decision to invest in Bluetooth also emanated from the board and, as this quote from the division head indicates, allowed the R&D team to dominate the process:

> With the BPA (Bluetooth protocol analyzer), when we started there was no market; even the initial Bluetooth standards document was not released. Many of the conferences and workshops were attended by only technical staff. Bluetooth as a technology was not proven. It was not clear whether it was competition to the existing IEEE [Institute of Electrical and Electronics Engineers] wireless LAN standards. Nobody knew what market segments will adopt this technology. Our initiative to invest in the wireless protocol analyzer with a focus on Bluetooth was driven by the board's vision to enter the wireless testing segment.

Having received the board's directive, Measuretronics decided to enter this technology segment through an acquisition. At the time of the acquisition, the firm was focused more on the technology and less on customer needs. Focusing on a product's technical characteristics is not unusual in the early stages of development. Research in the glass, cement, and minicomputers industries documents a similar technical focus in the early stages of a technology (Anderson and Tushman, 1990). In the case of the Bluetooth industry, the Bluetooth standard was close to being adopted as Measuretronics made its acquisition, increasing the pressure on the firm to release a product into the market quickly.

The division head continued:

> We did not have a short-range wireless testing product line at that time. The initial conceptual design of the product was purchased from Digicon (name disguised). We do not know if Digicon did any marketing investment to come up with the design. But I think the initial planning process was more focused on engineering viability. By the time we had got the conceptual design, the initial Bluetooth standard document was in final stages and Microsoft was planning the next release of its IDE platform.
>
> So our immediate focus was on getting the design to meet the Microsoft standards and Bluetooth standards. The engineering division took control from the start. They designed the initial specifications for the migration from Digicon to Measuretronics standards. The marketing guys had minimal role to play as much of the work was designed around compliance with standards.

Because the Bluetooth project was mandated by top management, and because of market uncertainty around the desired features, the firm's R&D team was placed at the helm of the development

DOI: 10.1002/smj

effort. The R&D team tended to draw its information from technical conferences and standards body meetings, which shaped its agenda around features. Ultimately, the product's initial design had limited functionality, as recognized by the head of product development:

> Eventually we released the first bare minimum version of the BPA. At that time our product was priced four times more than the nearest competitor, with fewer features. Many of our competitors were smaller players. The Measuretronics brand propelled us to a few good sales. But eventually customers started to complain. Unfulfilled requests started to pile up and costs had to go down. Engineering was given the goal to reduce costs, improve stability and focus on viability—things like, can we do this feature in software simulation rather than hardware simulation.

It was against this backdrop of an initial product release that the company began receiving feature requests from customers. Similar patterns of early product launches complemented by rapid customer-induced innovations have been documented in open source software (Lakhani and von Hippel, 2003). While fulfilling customer feature requests fell in the voluntary changes category, there were involuntary changes triggered by standards bodies that the company had to meet as well. Product development labor had to be allocated between the voluntary changes and the involuntary changes. They managed this trade-off by first prioritizing involuntary changes mandated by Bluetooth standard changes and competition releases, as explained by the program manager:

> Implementing all the feature requests is not possible. As I said before, the BPA product spans multiple industry segments. So we received diverse and contrasting feature requests. We had a team of only seven core design engineers who could individually build the features. Implementing the features require[s] a deep understanding of the emerging standards and norms that cannot be readily imparted through training. In fact there was no readymade material available for training. So we had to filter the feature requests.
>
> The engineering team reviewed the requests to check for hardware feasibility, compatibility with standards and [that they] be implementable using Microsoft VC++. For each feature we also check if installed-base versions will be affected. For each selected feature we do the COCOMO [constructive cost model] estimation and give a time estimate. Release timings are usually fixed based on Bluetooth conferences and competition releases. Since we know the approximate dates for these events, we prepare the preliminary functional specifications for the next release candidate.

Customer feature requests that were complex to fulfill (i.e., those that spanned multiple modules) also demanded high labor input. The product development managers faced a problem of allocating a fixed supply of development engineers among alternative projects. The range of projects fell broadly into two categories described earlier: voluntary changes in software, that is, meeting customer requests; and involuntary changes in software, that is, changes mandated by changes in hardware, new standards, or changes in firmware. From their articulation of priorities, it was clear that involuntary changes took priority over voluntary ones because being slow to complete involuntary changes jeopardizes the entire installed base. Thus, involuntary changes were linked to survival in the face of competitive pressures and received priority in allocation of R&D engineers (Aghion *et al*., 2001). The remaining capacity was allocated to feature requests from customers. That said, the allocation process of development engineers within the firm does not sufficiently explain why profitability considerations appear systematically less relevant. Our interviews with the head of product development suggested that the firm could not feasibly hire and train development engineers in response to excess demand because the technical labor market does not seamlessly adjust to short-run ebbs and flows in demand. In addition, top-down resource allocation and goal-setting processes made it difficult for subunits to engage in anticipatory hiring.

> We always try to distribute our experienced personnel over different programs. Outside hiring is limited to entry level programmers or people with specific skills like C#

DOI: 10.1002/smj

> or embedded programming. We wanted to hire about 10 engineers this year and it has been hard for us. We find lots of application programmers but we need electrical engineers with good hardware and software design skills. Out of the 10 positions, we have managed to fill only two positions in a six-month window. And out of the two new hires, one has already left because we could not promise any onsite opportunity for him. So it is very difficult to hire new personnel for our programs. Because of product line organizing it is difficult to have a big bench size by mass hiring. We do not know for sure if projects will be allocated in the future and having a big bench size without an adequate project pipeline is expensive.

While training and a shortage of skilled labor appear to be important constraints on the fulfillment of customer requests, it is likely a spurious reason. Even if the firm was able to rapidly hire and fire technical personnel in response to spikes in customer requests, it is unlikely that complex requests would be fulfilled. As noted earlier, the prior literature on software maintenance highlights the challenges of learning in the presence of turnover. Our interviews with the development engineers provided some insight into the primacy of complexity issues. The engineers placed a strong emphasis on minimizing disruptions in other parts of the organization. For instance, design changes that might necessitate hardware changes were rarely even discussed. The division of labor within the organization discouraged such changes.

> For enhancement programs (i.e., customer requests for features), we usually check if the base reference architecture needs a major revamp with respect to any emerging technologies or new standards. Until pushed to the limit, we tend not to change the core architecture. We want the core architecture to be stable so that we can focus more on the applications that deliver functionality.
>
> The reference architecture is the backbone of a product. It should not only facilitate the things marketing and engineering are asking for in the current release but should support future requirements without big changes. Architectural changes are difficult because we will have to redesign all the software interfaces and change the hardware layout, which has ripple effects on assembly and manufacturing. So keeping the architecture stable is a high priority.

While the quote above explains why complexity considerations might dominate fulfillment decisions, it says little about why customer demand or profitability is systematically uncorrelated with this decision. Our interviews suggested that the organization structure and budgeting process of product development, as well as the incentive system rewarding the engineers, partly explained why customer demand was uncorrelated with customer request fulfillment (Bower, 1970). The R&D activity in the firm is broken down progressively and assigned to program committees that are responsible for specific modules. These program committees enjoy significant autonomy over decisions that affect only their module. They are discouraged from pursuing changes that cross boundaries into other program committees. The gain from a proposed change has to be substantial before it is advanced for discussion across the program committees.

> At the board level, R&D budgets are either allocated to the instruments business or communications business. Inside each of these businesses there are several product lines and the budget for each product line is allocated by the general manager of the business line in consultation with the vice presidents for each product line. At the product line level, we form several program committees that decide on and pursue specific projects each year. Typically, a program committee consists of the product line technical head, key product engineers, and marketing members. It is headed by a program manager with strong product development experience.

The link to profitability was largely muted because product development occurred mostly at the program level, where choices about development were only remotely connected to product success. The choices of product development engineers were driven by two important criteria linked to their incentives: on-time completion of tasks and multitasking. The first criterion resulted in

DOI: 10.1002/smj

the engineers avoiding complex projects. Complex projects usually involved multiple modules, and coordinating changes across modules usually involved interfacing with other program teams. Because this introduced an additional element of uncertainty and affected the time required to complete a task, program teams tended to avoid fulfilling customer requests that spanned multiple modules. In addition, measuring multitasking created a perverse incentive among the development engineers (Hölmstrom and Milgrom, 1991). Specifically, it biased them toward fulfilling more requests rather than less, subject to the sole constraint of available engineering man-hours. In other words, the product development group chose to pursue two projects that would demand 100 man-hours rather than a single project that would take the same 100 man-hours—even if the latter was more profitable. We found that the annual performance appraisal of the product development engineers was heavily weighted toward the number of projects (identified by a unique ticket number) they worked on during the year. This would naturally give the product development group incentive to maximize the number of projects pursued.

> Product success is determined by the market share we have captured and profitability. But program success is different from product success. A product success (or failure) may be a function of several programs. So we cannot fully determine the success of a program based on product success or failure. At the program level, we evaluate engineering team members' contribution to the product features in terms of design ideas, minimizing build errors, on-time completion of tasks, and multitasking.

The analyses of the interview data suggests that the organization structure guiding the allocation of resources and the choice of projects, the primacy of product development engineers in the development process, their preoccupation with architectural stability, and their incentive structure, accounted for why profitability concerns were muted in the fulfillment decision. However, this does not explain why profitability became salient in the customization or standardization decision. Our interviews revealed a straightforward explanation (also related to the organization structure and incentives) for this, as recounted by the marketing manager:

> Engineering had to fulfill the numerous unfulfilled customer feature requests. They had a process to sort them into functional specs for releases—finding the viability of requests, to see if there are standards conflicts and so on. Once the engineering team decides on the viability and cost estimation, we decide on the fulfillment priority and mode of fulfillment based on market timing and profit considerations. We usually work closely with the development team to generate project effort estimates and sales and accounting to work on pricing and providing a scheduling estimate to the customer. If the customer agrees to the pricing and scheduling estimate, then we work with the legal team to develop a contract and an SLA [service level agreement].
>
> The marketing personnel are compensated based on product profits, forecasting accuracy, success of outbound marketing campaign, and customer satisfaction. We are constantly working closely with our customers and the accounting department to ensure that we are focused on profits.

In sum, the qualitative study identified three underlying elements of the organizational decision-making process that might prevent the DDI model from explaining investment in incremental innovation. First, *organization structure* drives the resource allocation process (Bower, 1970). In the case of Measuretronics, the allocation process was top-down rather than customer driven. The board tended to track technological developments rather than customers because the business of test and measurement often involved the development of products for which there was no established (or only nascent) customer needs. Thus, a customer-focus was historically less important among the product development engineers. Further, the actual development work was accomplished within autonomous program teams, which fragmented their attention and predisposed them to avoid taking on projects that would involve coordination across program teams. Thus, complex projects that spanned multiple modules were often avoided. Second, the engineering team had little control over the timing and choice of involuntary changes in the product. Because survival depended on rapid compliance with new and/or evolving

DOI: 10.1002/smj

standards, their development priorities were often dictated by developments in technical committees, such as IEEE, over which Measuretronics had no control. Thus, competitive pressures shifted the focus from customer requests to meeting and tracking technical changes in standards. In other words, survival dominated profitability in the decision-making process (Schaffer, 1989). Finally, we found that the incentives provided to the development personnel skewed their focus to maximize multitasking (i.e., taking on more projects) and fulfilling multiple requests regardless of how profitable they were (Hölmstrom and Milgrom, 1991). The same incentive structure explains why marketing personnel were more focused on profits in their decision to pursue customization or standardization.

DISCUSSION

Despite widespread agreement in both the scholarly and managerial literatures on the value and importance of prioritizing customers, achieving a customer-focused firm appears to be an elusive goal. In studying this puzzle, we examined whether firm decisions to allocate resources for incremental innovation primarily reflect customer demand or organizational constraints such as complexity. Our empirical analysis reveals interesting patterns. Whereas organizational considerations appear to drive the decision about whether or not to allocate resources, customer demand appears to have an impact on *how* firms implemented an investment. Having concluded the study, we believe the following four implications are important.

First, it is clear from our study that the challenges of being customer-focused emanate from the competing trade-offs of being technologically focused or working within organizational constraints. While the challenges of managing interdependent objectives are undeniable, being customer-focused is far from trivial or self-evident. Prioritizing customers is a choice like any other choice within organizations, and customer-focus is not equally valuable to all firms. Simply judging firms on the criterion of being customer-focused does a poor job of recognizing the myriad objectives that organizations pursue. The complexity involved in making decisions, in particular, inevitably advances some objectives while undermining others. In other words, the calculus of prioritizing customers is precisely that—a calculus—rather than a universal ideal that must always be pursued.

Second, the qualitative study revealed that fulfillment decisions were dependent upon the firm's resource allocation and decision-making processes. Arguably, firms (as opposed to the market) are a response to complexity in dealing with multiple objectives, such as staying abreast of technological change, meeting customer needs, generating shareholder value, keeping up with the competition, or lowering costs (Kogut and Zander, 1996). Because a firm cannot meet all of its objectives simultaneously, trade-offs are inevitable (Ethiraj and Levinthal, 2009). Consequently, firm organization structures may be seen as unique responses to the trade-offs among competing firm objectives (Chandler, 1962).

In the case of Measuretronics, the decision-making process was top-down and accorded primacy to the firm's technological environment. Top management believed the firm's mandate was to remain at the forefront of technological development, where customer needs had yet to arise. In industries experiencing rapid technological change, or in the case of new products where customer needs are ill-formed, paying close attention to customer requests may actually *increase* the competitive risk for firms. Under such circumstances, paying closer attention to competitors than to customers may enhance firm survival. In fact, depending on the particular environment and the kind of decision being made, customers, competitors, suppliers, employees, or regulatory entities may all become important levers guiding innovation decisions. As with radical innovation and architectural innovation, the primacy of customer-focus in incremental innovation decisions cannot be presumed.

Third, before we jump to the inference that our study exposes a flaw in the design of Measuretronics, we need to recognize that design solutions to complexity are only imperfect (Baldwin and Clark, 2000). Organizations may choose to artificially limit the zone of operation of each microunit for one of two reasons. Often the full cost of accomplishing a given task cannot easily be estimated, such as when there are interdependencies between elements of a system (Thaler, 2000). For instance, the full cost of adopting fuel-cell technology in cars is largely unknown since the fuel-cell engine

DOI: 10.1002/smj

shares interdependencies with refueling, distribution, and the adaptation of existing safety features (National Academy of Sciences, 2004). When the full cost of doing something cannot be accurately estimated, encouraging cross-unit communication to pursue complex innovation is unlikely to solve the problem. Significant interdependencies preclude the accurate estimation of total costs and benefits of an innovation. Because an economic basis for a go or no-go decision is difficult to achieve in any case, an arbitrary design rule—in this case, that a program team should confine itself to tightly bounded changes—appears reasonable.

Fourth, if interdependencies make it difficult to estimate the cost of managerial action, allowing unbounded innovation exposes an organization to severe bargaining problems. A key question is whether bargaining is an efficient mechanism for conflict resolution (Cyert and March, 1963). Asking managers to bargain continually to determine if an innovation should be pursued will debilitate action and freeze organizations around the status quo (Ethiraj and Levinthal, 2009). Under this circumstance, the organization structure and the division of responsibility provides a mechanism for the quasi-resolution of conflict and enables positive managerial action, though it appears globally inefficient. Perhaps the contradiction appears smaller if we believe that activities move from the market to the firm only when complexity precludes global efficiencies in the first place. Complexity, by definition, precludes the identification of optimal decisions and raises the importance of behavioral processes in decision making (Rivkin, 2000). Thus, the important research question here is how organizations should choose among multiple paths in innovation decisions, each of which presents attendant costs and benefits. Admittedly, we have only scratched the surface of this phenomenon. Much work remains before we can explain how organizations make innovation decisions in the face of complexity.

Finally, an important managerial implication arises from the effect of various organizational factors on innovation decisions. Organizations might prioritize one or more goals such as customer-focus or cost minimization. Implementing those goals, however, might be complicated by organizational considerations. Our data suggest that organizational policies for resource allocation and incentive systems often produce their own side effects, which may not align with organizational goals. While decision making can reflect unitary goals and a clear cost-benefit calculus in theory, in practice their implementation is undeniably linked with other organizational processes. Understanding the behavioral underpinnings of resource allocation decisions requires us to pay particular attention to organizational processes, which may filter information or alter how a decision is implemented.

This study has several limitations. First, it is based on a single industry with its own peculiar characteristics. It is not clear to what extent the substantive results of this paper are generalizable across industries. For instance, measuring complexity itself may be difficult in industries beyond software, limiting the study's external validity. Our study is also based on data from a single firm. Ideally, we would have included data from several firms, but accessing data this detailed is extremely difficult. In our case, it involved several years of data collection, ongoing negotiations with the subject firm, and signing nondisclosure agreements. This study also required detailed documentation of the process used to handle customer requests, which may be unique; many firms we contacted do not collect similar data. We believe that providing a detailed look at the decision-making process within one firm, in addition to a large sample empirical analysis, offers important insights about how complexity interacts with organizational characteristics to impact innovation activity.

ACKNOWLEDGEMENTS

We thank Editor Rich Bettis, two anonymous reviewers, and seminar participants at the University of Michigan for helpful comments. We appreciate the generous time and effort given by the engineers and managers at Measuretronics. We thank Erin Lee Martin for excellent editing. Errors and omissions remain our own.

REFERENCES

Acemoglu D, Linn J. 2004. Market size in innovation: theory and evidence from the pharmaceutical industry. *Quarterly Journal of Economics* **119**(3): 1049–1090.

Aghion P, Harris C, Howitt P, Vickers J. 2001. Competition, imitation and growth with step-by-step innovation. *Review of Economic Studies* **68**(3): 467–492.

Anderson P, Tushman ML. 1990. Technological discontinuities and dominant designs: a cyclical model of

technological change. *Administrative Science Quarterly* **35**(4): 604–633.
Argyres N. 1996. Capabilities, technological diversification and divisionalization. *Strategic Management Journal* **17**(5): 395–410.
Arrow K. 1962. Economic welfare and the allocation of resources for invention. In *The Rate and Direction of Inventive Activity: Economic and Social Factors.* Nelson RR (ed). Princeton University Press (for the National Bureau of Economic Research): Princeton, NJ; 609–626.
Baldwin CY, Clark KB. 2000. *Design Rules: The Power of Modularity*. MIT Press: Cambridge, MA.
Banker RD, Datar SM, Kemerer CF, Zweig D. 1993. Software complexity and maintenance costs. *Communications of the ACM* **36**(11): 81–94.
Banker RD, Davis GB, Slaughter SA. 1998. Software development practices, software complexity, and software maintenance performance: a field study. *Management Science* **44**(4): 433–451.
Baron RM, Kenny DA. 1986. The moderator mediator variable distinction in social psychological research: conceptual, strategic, and statistical considerations. *Journal of Personality and Social Psychology* **51**(6): 1173–1182.
Booch G. 1994. *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings: Redwood City, CA.
Bower JL. 1970. *Managing the Resource Allocation Process*. Harvard Business School: Boston, MA.
Brooks FP. 1995. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley: Reading, MA.
Burgelman RA. 1994. Fading memories: a process theory of strategic business exit in dynamic environments. *Administrative Science Quarterly* **39**(1): 24–56.
Chan TZ. 2000. Beyond productivity in software maintenance: factors affecting lead time in servicing users' requests. International Conference on Software Engineering Proceedings, October, San Jose, CA. *Software Maintenance.* IEEE Computer Society: Los Alamitos, CA; 228–235.
Chandler AD. 1962. *Strategy and Structure: Chapters in the History of the American Industrial Enterprise*. MIT Press: Cambridge, MA.
Christensen C. 1997. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harvard Business School Press: Boston, MA.
Christensen CM, Bower JL. 1996. Customer power, strategic investment, and the failure of leading firms. *Strategic Management Journal* **17**(3): 197–218.
Cusumano MA, Selby RW. 1998. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*. Simon & Schuster: New York.
Cyert RM, March JG. 1963. *A Behavioral Theory of the Firm*. Prentice-Hall: Englewood Cliffs, NJ.
David PA. 1985. Clio and the economics of QWERTY. *American Economic Review* **75**(2): 332–337.
Dawes RM, Faust D, Meehl PE. 1989. Clinical versus actuarial judgment. *Science* **243**(4899): 1668–1674.
den Besten M, Dalle JM, Galia F. 2008. The allocation of collaborative efforts in open-source software. *Information Economics and Policy* **20**(4): 316–322.
Dencker JC. 2009. Relative bargaining power, corporate restructuring, and managerial incentives. *Administrative Science Quarterly* **54**(3): 453–485.
Doz YL. 1996. The evolution of cooperation in strategic alliances: initial conditions or learning processes? *Strategic Management Journal*, Summer Special Issue **17**: 55–83.
Ethiraj SK. 2007. Allocation of inventive effort in complex product systems. *Strategic Management Journal* **28**(6): 563–584.
Ethiraj SK, Kale P, Krishnan MS, Singh JV. 2005. Where do capabilities come from and how do they matter? A study in the software services industry. *Strategic Management Journal* **26**(1): 25–45.
Ethiraj SK, Levinthal DA. 2009. Hoping for A to Z while rewarding only A: complex organizations and multiple goals. *Organization Science* **20**(1): 4–21.
Fong Boh W, Slaughter SA, Espinosa JA. 2007. Learning from experience in software development: a multilevel analysis. *Management Science* **53**(8): 1315–1331.
Garud R, Kumaraswamy A. 1995. Technological and organizational designs for realizing economies of substitution. *Strategic Management Journal*, Summer Special Issue **16**: 93–109.
Gilbert CG. 2006. Change in the presence of residual fit: can competing frames coexist? *Organization Science* **17**(1): 150–167.
Hansen MT. 1999. The search-transfer problem: the role of weak ties in sharing knowledge across organization subunits. *Administrative Science Quarterly* **44**(1): 82–111.
Henderson RM. 1993. Underinvestment and incompetence as responses to radical innovation: evidence from the photolithographic alignment equipment industry. *RAND Journal of Economics* **24**(2): 248–270.
Henderson RM, Clark KB. 1990. Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly* **35**(1): 9–30.
Henderson RM, Cockburn I. 1994. Measuring competence? Exploring firm effects in pharmaceutical research. *Strategic Management Journal*, Winter Special Issue **15**: 63–84.
Hölmstrom B, Milgrom P. 1991. Multitask principal agent analyses: incentive contracts, asset ownership, and job design. *Journal of Law Economics & Organization* **7**: 24–52.
Huckman RS, Staats BR, Upton DM. 2009. Team familiarity, role experience, and performance: evidence from Indian software services. *Management Science* **55**(1): 85–100.
Jones C. 2000. *Software Assessments, Benchmarks, and Best Practices*. Addison-Wesley: Reading, MA.
Kahneman D, Lovallo D. 1993. Timid choices and bold forecasts: a cognitive perspective on risk taking. *Management Science* **39**(1): 17–31.

DOI: 10.1002/smj

Kamien MI, Schwartz NL. 1982. *Market Structure and Innovation*. Cambridge University Press: Cambridge, UK.

Kemerer CF, Slaughter SA. 1997. Determinants of software maintenance profiles: an empirical investigation. *Journal of Software Maintenance-Research and Practice* **9**(4): 235–251.

Kogut B, Zander U. 1996. What firms do? Coordination, identity, and learning. *Organization Science* **7**(5): 502–518.

Krishnan MS, Mukhopadhyay T, Kriebel CH. 2004. A decision model for software maintenance. *Information Systems Research* **15**(4): 396–412.

Lakhani K, von Hippel E. 2003. How open source software works: free user-to-user assistance. *Research Policy* **32**(6): 923–943.

Langlois RN, Everett MJ. 1992. Complexity, genuine uncertainty, and the economics of organization. *Human Systems Management* **11**(2): 67–75.

Lippman S, Rumelt R. 1982. Uncertain imitability: an analysis of interfirm differences in efficiency under competition. *Bell Journal of Economics* **13**(2): 418–453.

Lohr S, Markoff J. 2004. In the battle of the browsers '04, Firefox aims at Microsoft. *New York Times,* 15 Nov., C4.

MacDuffie JP. 1997. The road to 'root cause': shop floor problem-solving at three auto assembly plants. *Management Science* **43**(4): 479–502.

MacKinnon DP, Lockwood CM, Hoffman JM, West SG, Sheets V. 2002. A comparison of methods to test mediation and other intervening variable effects. *Psychological Methods* **7**(1): 83–104.

Maddala GS. 1983. *Limited Dependent and Qualitative Variables in Econometrics*. Cambridge University Press: Cambridge, UK.

Marginson DEW. 2002. Management control systems and their effects on strategy formation at middle-management levels: evidence from a U.K. organization. *Strategic Management Journal* **23**(11): 1019–1031.

National Academy of Sciences. 2004. *The Hydrogen Economy: Opportunities, Costs, Barriers, and R&D Needs*. National Academies Press: Washington DC.

Noda T, Bower JL. 1996. Strategy making as iterated processes of resource allocation. *Strategic Management Journal*, Summer Special Issue **17**: 159–192.

Novak S, Eppinger SD. 2001. Sourcing by design: product complexity and the supply chain. *Management Science* **47**(1): 189–204.

Nutt PC. 1998. How decision makers evaluate alternatives and the influence of complexity. *Management Science* **44**(8): 1148–1167.

Read D, Loewenstein G, Rabin M. 1999. Choice bracketing. *Journal of Risk & Uncertainty* **19**(1–3): 171–197.

Reinganum JF. 1985. Innovation and industry evolution. *Quarterly Journal of Economics* **100**(1): 81–99.

Rivkin J. 2000. Imitation of complex strategies. *Management Science* **46**(6): 824–844.

Schaffer ME. 1989. Are profit maximizers the best survivors? A Darwinian model of economic natural selection. *Journal of Economic Behavior and Organization* **12**(1): 29–45.

Schilling M. 2003. Technological leapfrogging: lessons from the U.S. video game console industry. *California Management Review* **45**(3): 6–32.

Schilling M, Steensma KH. 2001. The use of modular organizational forms: an industry-level analysis. *Academy of Management Journal* **44**(6): 1149–1168.

Schmookler J. 1966. *Invention and Economic Growth*. Harvard University Press: Cambridge, MA.

Schumpeter JA. 1934. *The Theory of Economic Development*. Harvard University Press: Cambridge, MA.

Schumpeter JA. 1950. *Capitalism, Socialism, and Democracy*. Harper and Bros.: New York.

Simonin BL. 1999. Ambiguity and the process of knowledge transfer in strategic alliances. *Strategic Management Journal* **20**(7): 595–623.

Sinclair G, Klepper S, Cohen W. 2000. What's experience got to do with it? Sources of cost reduction in a large specialty chemicals producer. *Management Science* **46**(1): 28–45.

Stross R. 2004. The fox is in Microsoft's henhouse (and salivating). *New York Times,* 19 Dec., 3,5.

Subramanian GH, Pendharkar PC, Wallace M. 2006. An empirical study of the effect of complexity, platform, and program type on software development effort of business applications. *Empirical Software Engineering* **11**(4): 541–553.

Szulanski G. 1996. Exploring internal stickiness: impediments to the transfer of best practice within the firm. *Strategic Management Journal*, Winter Special Issue **17**: 27–43.

Thaler RH. 1985. Mental accounting and consumer choice. *Marketing Science* **4**: 199–214.

Thaler RH. 1999. Mental accounting matters. *Journal of Behavioral Decision Making* **12**(3): 183–206.

Thaler RH. 2000. Mental accounting matters. In *Choices, Values and Frames.* Kahneman D, Tversky A (eds). Cambridge University Press: Cambridge, UK; 241–268.

Thompson JD. 1967. *Organizations in Action*. McGraw-Hill: New York.

Ton Z, Huckman RS. 2008. Managing the impact of employee turnover on performance: the role of process conformance. *Organization Science* **19**(1): 56–68.

Trajtenberg M. 1990. A penny for your quotes: patent citations and the value of innovations. *RAND Journal of Economics* **21**(1): 172–187.

Tushman ML, Anderson P. 1986. Technological discontinuities and organizational environments. *Administrative Science Quarterly* **31**(3): 439–465.

Ulrich KT. 1995. The role of product architecture in the manufacturing firm. *Research Policy* **24**: 419–440.

Ulrich KT, Eppinger SD. 1999. *Product Design and Development* (2nd edn). McGraw-Hill: New York.

von Hippel E. 1976. Dominant role of users in scientific instrument innovation process. *Research Policy* **5**(3): 212–239.

Winter SG, Szulanski G. 2001. Replication as strategy. *Organization Science* **12**(6): 730–743.

DOI: 10.1002/smj