Strategic Management Journal
Strat. Mgmt. J. (2015)
Published online EarlyView in Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/smj.2354
Received 1 May 2013; *Final revision received 19 September 2014*

CAPABILITY EROSION DYNAMICS

HAZHIR RAHMANDAD[12]* and NELSON REPENNING[2]
[1] *Industrial and Systems Engineering, Virginia Tech, Falls Church, Virginia, U.S.A.*
[2] *System Dynamics, MIT Sloan, Cambridge, Massachusetts, U.S.A.*

The notion of capability is widely invoked to explain differences in organizational performance, and research shows that strategically relevant capabilities can be both built and lost. However, while capability development is widely studied, capability erosion has not been integrated into our understanding of performance heterogeneity. To understand erosion, we study two software development organizations that experienced diverging capability trajectories despite similar organizational and technological settings. Building a simulation-based theory, we identify the adaptation trap, a mechanism through which managerial learning can lead to capability erosion: well-intentioned efforts by managers to search locally for the optimal workload balance lead them to systematically overload their organization and, thereby, cause capabilities to erode. The analysis of our model informs when capability erosion is likely and strategically relevant.

INTRODUCTION

Explaining performance and productivity variations across firms in similar markets is a major concern of strategy scholars and economists (Gibbons and Henderson, 2010; Grant, 2010; Syverson, 2011). To explain this observation, several papers invoke the notion of capability—a set of routines that enable a firm to produce a particular output given a set of inputs and targets (e.g., develop new products) (Winter, 2003). This perspective holds that variations in performance across firms in the same industry can be explained by differences in their underlying capabilities (Hoopes and Madsen, 2008; Leiblein, 2011; Nelson and Winter, 1973). Capabilities in this view manifest as a configuration of organizational routines and processes that combine to create a complex (Denrell, Fang, and Winter, 2003), nonlinear performance landscape (Levinthal, 1997, 2000). The processes of imitation and adaptation on this landscape can lead to the emergence of different capability configurations (Rivkin, 2001; Rivkin and Siggelkow, 2003) even in the presence of competition (Lenox, Rockart, and Lewin, 2006, 2010), and thus the observed heterogeneity.

One of the earliest articulations of the role of capabilities in determining competitive outcomes (Dierickx and Cool, 1989) recognized that capabilities tend to behave like "stock" or "level" variables, meaning that they take time to both accumulate and erode. Subsequent research, however, has focused almost exclusively on the processes governing the accumulation of capabilities, implicitly assuming that once a particular configuration is achieved, the firm can continue to profit from any associated performance benefits until environmental change renders one or more capabilities obsolete and the corresponding configuration inferior. While turnover and forgetting may lead to the loss of some skills and minor disruption to routines, local adaptation allows the firm to rebuild the capability, and systematic erosion of capabilities is not likely. Explaining heterogeneous performance outcomes using this perspective thus requires understanding the mechanisms through which capabilities

Keywords: capability; erosion; resource-based view; simulation; system dynamics
*Correspondence to: Hazhir Rahmandad, E62-462, 100 Main St., Cambridge, MA 02142, E-mail: hazhir@vt.edu

accumulate and how that accumulation might differ across firms.

Despite the lack of attention given to it, erosion may play a significant role in determining competitive outcomes. Consider two hypothetical cases: (1) a population of firms in which it is very hard to find high-performing configurations but easy to keep them, and where environmental changes gradually render high-performing configurations ineffective; (2) a population in which the high-performing configurations are easy to find but also likely to be lost. In both cases, we would observe significant performance and capability heterogeneity, but the underlying mechanisms would be very different. Leaving aside imitation effects, differential capability acquisition rates will be the key mechanism in one, while the other will depend on differences in capability erosion rates. As first suggested by Dierickx and Cool (1989), the possibility of erosion could thus prove critical in explaining the distribution of firms' performance. Occupying the high-performing configurations in the strategic landscape might thus depend not only on the entry rate of firms to those configurations, but also on the (undesired) exit rates from these configurations, i.e., due to capability erosion.

Consider an empirical example. Several decades into the life of the software industry the basic routines constituting effective software development capability are well known and stable (Cusumano and Selby, 1995; Jones, 2000). Yet we do not find firms in this market uniformly mastering this capability, and many software projects continue to be significantly challenged. Figure 1 shows historical data on the fraction of software projects falling in the Standish surveys' successful, challenged, and failed categories (http://www.standishgroup.com/). While the category definitions and quality of such surveys could be debated, the basic finding that performance patterns show little convergence over time is informative. Assuming firms do not need decades to develop a capability (the elements of which are well known), then performance heterogeneity may better be explained by firms' inability to sustain this capability, i.e., capability erosion.

We begin by positioning capability erosion within the broader literature, including studies outside of strategy research that document cases of organizational downfall. Next, we draw on two case studies to describe the processes underlying capability erosion and how it can be the unintended outcome of organizational adaptation processes.

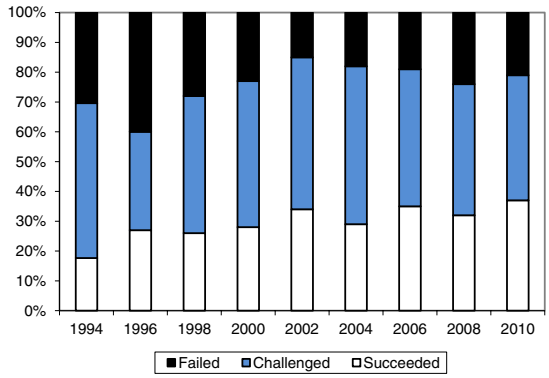

Figure 1. The fraction of IT projects surveyed by Standish group over years that fall into categories of successful (on time and budget), challenged (failing to meet time and budget goals), and failed (had to be abandoned). Source: http://www.standishgroup.com/

Building on these cases, we develop a simulation model to capture the mechanisms that regulate the capability erosion dynamics we observe and the role of managerial action in these dynamics. Analyzing the model allows us to identify the settings under which capability erosion dynamics would be relevant. We use our findings to build a theory that connects capability erosion to strategy and discuss the implications for research and practice.

Capability development and erosion

Explaining persistent performance differences across similarly positioned organizations has motivated several recent studies in both strategy and economics (Gibbons and Henderson, 2010; Grant, 2010; Syverson, 2011). Large sample studies have shown that a substantial portion of the variation in firm performance arises from differences in internal practices and capabilities even after controlling for industry and year effects (e.g., Bloom and Van Reenen, 2007; Mauri and Michaels, 1998; Rumelt, 1991; Spanos and Lioukas, 2001). Studies performed at a more micro-level reach similar conclusions. In situations ranging from heavy manufacturing—pig iron (Salter, 1960), steel (Ichniowski, Shaw, and Prennushi, 1997), and ship building (Argote, Beckman, and Epple, 1990)—to cooking meals for commercial aircraft (Chew, Bresnahan, and Clark, 1990) and human resource practices (Ichniowski and Shaw, 2003), in-depth inquiries repeatedly find significant differences in both the conduct and the outcomes of similar tasks. Large-scale organizational experiments have

DOI: 10.1002/smj

confirmed the significant performance impact of adopting alternative firm-level practices (Bloom *et al.,* 2013). This line of research suggests both that internal practices and capabilities matter, and that differences in those capabilities are not quickly eliminated by imitation (Henderson and Cockburn, 1994; Roberts and Dowling, 2002; Syverson, 2011).

For capabilities to have real explanatory power, they need to be understood and measured independently of the performance outcomes ascribed to them (Priem and Butler, 2001; Williamson, 1999). Therefore, there has been increasing interest in the mechanisms through which capabilities are built, developed, and matured (Helfat, 2000; Helfat and Peteraf, 2003). Building on Nelson and Winter (1982), several theorists have advanced the notion that capabilities result from the repeated execution of specific actions and experiential learning (Parmigiani and Howard-Grenville, 2011; Zollo and Winter, 2002). Early studies emphasized the routine-driven and less mindful aspects of capability dynamics (Schreyoegg and Kliesch-Eberl, 2007). More recently, the cognitive and hierarchical foundations of capabilities have received more attention (Felin *et al.,* 2012; Foss *et al.,* 2012; Gavetti, 2005; Levinthal and Rerup, 2006), and several studies have focused on the dynamic capabilities that modify operational capabilities and other organizational resources (Eisenhardt and Martin, 2000; Teece, 2007; Teece, Pisano, and Shuen, 1997; Winter, 2003).

Given the large and growing body of research on capability dynamics, it is curious that capability erosion, defined as systematic loss of effective capabilities already established in an organization, has received such little attention. If variations in capability levels are central to variations in firm performance, it is conceivable that variations in capability erosion can be part of the explanatory mechanisms for performance heterogeneity. Dierickx and Cool's classic paper underlines the potential relevance of resource erosion dynamics to performance heterogeneity (Dierickx and Cool, 1989), but subsequent research on capability dynamics and economic research on productivity variation have largely excluded erosion as a noteworthy possibility. Some formal models have considered the existence of organizational forgetting or erosion (Hodgson and Knudsen, 2004; Rahmandad, 2012) due to turnover or insufficient organizational memory systems (Argote, 2012), but the phenomenon has not been considered as strategically relevant, potentially because forgetting rates are seen as too small (Thompson, 2007) or not systematically different across competing firms. Overall, an existing capability is seen as subject to obsolescence, but the underlying routines are perceived as robust and persistent once established (Helfat and Peteraf, 2003).

Two different bodies of literature motivate a more serious look at erosion. First, research on learning curves suggests that organizations often forget at significant rates, empirically estimated to be as high as 40 percent per year, in settings from pizza franchises to aircraft production (Benkard, 2000; Darr, Argote, and Epple, 1995). Moreover, these rates vary significantly across studies (Benkard, 2000; Darr *et al.,* 1995; Thompson, 2007). However, these studies do not delve into the mechanisms of capability erosion and the sources of heterogeneity in forgetting rates.

Second, several case studies suggest that capability erosion can play a significant role in organization failure (e.g., Hall, 1976). Klein and Sorra (1996), Sterman, Repenning, and Kofman (1997), Easton and Jarrell (1998), Pfeffer, Sutton, and NetLibrary Inc. (2000), and Repenning and Sterman (2002) all document cases in which firms had access to valuable capabilities and resources but failed to capitalize fully on them or lost them over time. Carroll and colleagues (Carroll, Sterman, and Markus, 1997) and Zuashkiani et al. (Zuashkiani, Rahmandad, and Jardine, 2011) underline the potential for the loss of maintenance capability due to the reinforcing feedbacks that promote reactive maintenance. Repenning (2000, 2001) documents dynamics that tip product development practices into an inefficient firefighting mode causing firms to lose their ability to produce high-quality products. Imbalanced growth and goal adjustment have been shown to erode service capability and undermine a firm's success in settings ranging from the airline industry (Morecroft, 1997) to banking (Oliva and Sterman, 2001). Furthermore, the theory of normal accident posits that tightly coupled systems always include inherent failure modes due to dysfunctional interactions among system components that can compromise performance and safety of the large systems (Perrow, 1999; Rudolph and Repenning, 2002; Sagan, 1993).

While these studies provide examples of capability erosion, their impact on strategy research has been limited. To be strategically significant, organizational pathologies need to be more than

DOI: 10.1002/smj

infrequent cases of failure; they should reveal common mechanisms that could impact the *heterogeneity* of performance in a population of firms. Not all maintenance organizations fall into the reactive mode, rates of "normal accidents" vary significantly across industries, and forgetting rates are highly variable across different organizations, and it is exactly this variation that makes capability erosion an interesting mechanism to study. Why are capabilities subject to erosion in some settings and resilient in others? Studying the organizational characteristics, task structures, and feedback mechanisms that regulate capability erosion rates might allow us to identify both the settings in which capability erosion matters for understanding performance heterogeneity and the role of managers in moderating those dynamics. By connecting the two disparate literatures, an explicit theory of capability erosion offers the possibility of new explanatory mechanisms to understand firm heterogeneity and an enhanced understanding of organizational demise.

DATA AND METHODS

Our study of capability erosion is grounded in data from multiple-release software development. Product development (PD) is one of the major capabilities that influence firms' performance (Eisenhardt and Martin, 2000; Winter, 2003) and an important empirical setting for studying capabilities (Chao, Kavadias, and Gaimon, 2009). Moreover, the software industry displays significant heterogeneity in PD performance (Moløkken and Jørgensen, 2003) despite the large literature on best practices (Jones, 2000). While there is a rich literature looking at PD in general (see Brown and Eisenhardt, 1995, and Krishnan and Ulrich, 2001, for reviews), and single project dynamics (Abdel-Hamid and Madnick, 1991; Ford and Sterman, 1998; Lyneis and Ford, 2007), research on the dynamics of multiple-release product development processes has been more limited.

Our method for building theory about capability erosion is the model-based case study (e.g., Perlow and Repenning, 2009). Our study proceeds in three steps, generating a different flavor of "theory" at each juncture (Sutton and Staw, 1995). In the first step, we report an inductive study of two software development projects within a single firm that displayed persistent and growing differences in performance, despite sharing many similar features and repeated attempts to improve the performance of the lower-performing product. In the second step, we use both the case and existing theory to induce a simple dynamic model that elaborates on the mechanisms that underlie capability erosion in our setting and yields a logically consistent, formal theory of the phenomenon. Analyzing the model yields internally valid inferences, well-defined boundary conditions for those inferences, and testable propositions (Harrison *et al.,* 2007). In the final phase, we build on the insights from the formal model to outline a more general, text-based theory of capability erosion and its implications. Our results are afflicted by the usual limitations of theories inspired by small samples, but our use of multiple methods should enhance internal consistency and better position future researchers to test the explanatory power of our theory.

The cases motivating the modeling work were selected inside Sigma, a global telecommunication firm, using an intrafirm polar type research design (Yin, 2009). With Sigma's support, we identified Alpha and Beta, two multiple release software products that showed significant differences in development outcomes despite comparable technical complexity, resource loading, and incentive structures. We selected cases in technologically stable markets to avoid confounding of exogenous technological shocks with capability erosion. The comparison of these two product teams and their practices provided an opportunity to model how differing approaches to sustaining PD capability could yield different performance levels, despite similar exogenous conditions and the absence of technological shocks.

Data collection

At the study's outset (summer 2004), the first author spent three months on site at Sigma conducting 48 semi-structured interviews, each lasting between 30 and 90 minutes. The interviews included members of all the areas involved in product development within each product team: services, sales, architects and systems engineers, developers, testers, customer service staff, sales support, marketing personnel, and senior managers. Much of the initial effort focused on clearly delineating what was actually done to develop new products (routines constituting the PD capability) and how that had changed over time. To that end, the first author also travelled to two other sites, attended team

DOI: 10.1002/smj

meetings, and reviewed the available archival data, mostly from Alpha's case, including planning and review files for different releases of the product, internal assessments conducted at the end of major releases, and quantitative data on code development and bug-fixing, human resources, and problems reported by customers. Additional interviews were later conducted (until spring 2005) to evaluate and expand emerging model constructs and relationships. These efforts included 22 additional interviews and a group session that elicited ideas from 26 experienced members of the firm.

Analysis

We analyzed the data in two phases. First, following standard practice for qualitative research (e.g., Glaser and Strauss, 1967), we focused on identifying emerging categories and causal relationships that informed the development of the simulation model. In each interview, we searched for PD practices, how they changed over time, and how they related to major performance metrics (quality, cost, and schedule) and looked for corroborations or disagreements across interviews. Based on these practices and the observations on the site, we generated several hypotheses to explain observed differences in performance outcomes across multiple releases of Alpha and Beta. Some of the identified processes confirmed the existence of dynamics identified in prior research on firefighting (Repenning, 2001); others, however, revealed new mechanisms. The complete set of these dynamic hypotheses appears in File S1. We then narrowed the list of hypotheses based on the most prominent themes found in the interviews and product histories and focused our analysis on the mechanisms that extended beyond those in prior research.

In the second phase, we integrated the hypotheses that emerged from the first phase into a single cohesive theory via the development of a system dynamics simulation model (Forrester, 1961; Gary, 2005; Sterman, 2000). The modeling process entailed first building a detailed model of software development in Alpha and Beta, validated by the historical performance data, and then extracting the main dynamics in a simpler, more stylized model that captured the essential elements of our emerging theory. This generic model is discussed in detail in this paper (see Rahmandad, 2005, for a full report of the detailed model and Rahmandad and Weiss, 2009, for an intermediate-size model focusing on concurrent software development). Model formulations were developed based on case data and the relevant modeling literature (Abdel-Hamid and Madnick, 1991; Ford and Sterman, 1998). We used the archival data and expert judgment to parameterize and validate the detailed model. The parsimony of the generic model, however, comes at the cost of simplified formulations and lack of calibration to numerical data. While we conceptually separate the two phases of inductive research and modeling, in practice the modeling and fieldwork informed each other in a continuous dialogue.

The simulation model served three purposes in our study. First, by flagging missing and inconsistent components in our initial set of hypotheses, the formal model enforced the internal consistency of our theory. Second, the use of simulation offset the cognitive limitations normally associated with drawing inferences from dynamics theories (see Sterman, 1994, for a summary of these limitations). Finally, via extensive simulation experiments, the model enabled the derivation of testable predictions that extend our theory beyond the context of the case studies in which it is developed.

VARIANCE IN SOFTWARE DEVELOPMENT AT SIGMA

The espoused development process at Sigma

We first outline Sigma's prescribed software development process, which all product teams were supposed to follow, and then elaborate on the differing performance of the two projects we studied. Software is typically developed in three phases (design, development, testing), which can be followed serially (e.g., waterfall) or iteratively (e.g., spiral). Sigma's development process was largely waterfall with the occasional iterative element. The practices that constitute an effective software development capability are now well established (Jones, 2000), and Sigma's prescribed process closely followed accepted best practice for waterfall development. In the concept design phase, the main features to be added to the next release are chosen, the product architecture is designed, and general requirements for developing different pieces of code are established. Subsequently, a more detailed outline of the requirements is developed that describes the inputs, outputs, and performance requirements for the new

DOI: 10.1002/smj

or modified code. Next, software engineers (developers) develop the code based on these requirements. The quality of the development depends on several factors, including the developers' skills, complexity of the software, adherence to good practice (e.g., writing detailed requirements and doing code reviews), the quality of the code being built upon, and the quality of the software architecture (MacCormack *et al.,* 2003). Preliminary tests (unit tests) are often run in this stage to insure the quality of the new chunks of code before they are integrated together and sent to the quality assurance stage. Quality assurance entails testing the code to find problems that stop the software from functioning as desired. Often these tests reveal problems in the code that are referred back to developers for rework. This rework cycle continues until the software passes the tests and is released. Due to the multitude of ways a product can be used in the field, not all possible errors are discovered during testing. Therefore, there are always some bugs in the released software.

Customers (typically large businesses in Sigma's case) buy and install the software and require service from the company. The service department follows up on problems reported by customers and refers bugs to the development organization. Bugs, if they significantly detract from customer satisfaction, are fixed at the customer site through an activity known as current engineering (CE). Noncritical bugs are fixed in a subsequent release of the software, an activity known as bug-fixing.

The development, launch, and service activities discussed above focus on a single release (version) of the software, but there are typically multiple releases under development, resulting in two key interconnections across the different releases. First, subsequent versions of the software build on their predecessors and hence carry over problems in code and architecture if those problems are not remedied. Second, because they draw off a common resource pool of developers, resources must be shared among new development, fixing the architecture, bug-fixing, and CE. Therefore, by influencing the level of CE and bug-fixing, the quality of past releases impacts the resources available for new development.

Overview of two cases

The modeling was inspired by case studies of two projects, Alpha and Beta. Alpha is a customer relationship management (CRM) software for call centers. While initially produced by a small start-up firm, Alpha was acquired by Sigma in 1996, and the team working on it has fluctuated around 120 full-time employees. We focused our data-gathering effort on Alpha's history after 2000. Product Beta is part of a telecommunication switch that is developed largely independently of its parent product, but is tested, launched, and sold together with that parent. The majority of Beta's over-80-person R&D team focuses on developing software.

Over the study period, Beta's development capability conforms to the above prescription, emphasizing early review, root-cause analysis and extensive automation of testing. As a result, it often faces fewer defects than many other products. When a field problem appears, the team does a root-cause analysis to determine its sources, thus identifying and eliminating a potential failure mode in the underlying process. While in other teams it is not unusual for 30 percent of the development resources to be engaged in current engineering activities, in Beta this number has always remained under 10 percent. Beta has also delivered most of its releases on time. In two cases, Beta had to delay a release to remain in sync with its parent product, which was delayed.

Alpha's capability has been changing significantly over time. Whereas early releases were seen to follow the best practices outlined above, over time Alpha's development process lost its rigor. As a result, while initial releases of Alpha met with significant success, led the market, and made the product a strategic asset for Sigma, since 2003, due to increasing delays in delivery and low quality, Alpha has ceded its leadership position. Alpha's performance deteriorated in each of the three categories in which software development is typically measured: schedule, quality, and productivity. Measured either in absolute terms or as a percent of the original schedule, Alpha's timeliness worsened or did not improve over the six releases we studied (Figure 2a). Quality, typically defined by customer experience of the software, is measured by the number of serious bugs reported per customer in the first few months of installation of the software. As shown in Figure 2(b), the number of bugs in Alpha grew on average over the period of study. Finally, the productivity of the team can be measured by the number of lines of code developed per individual involved with the R&D process. As shown in Figure 2(c), Alpha's productivity declined

DOI: 10.1002/smj

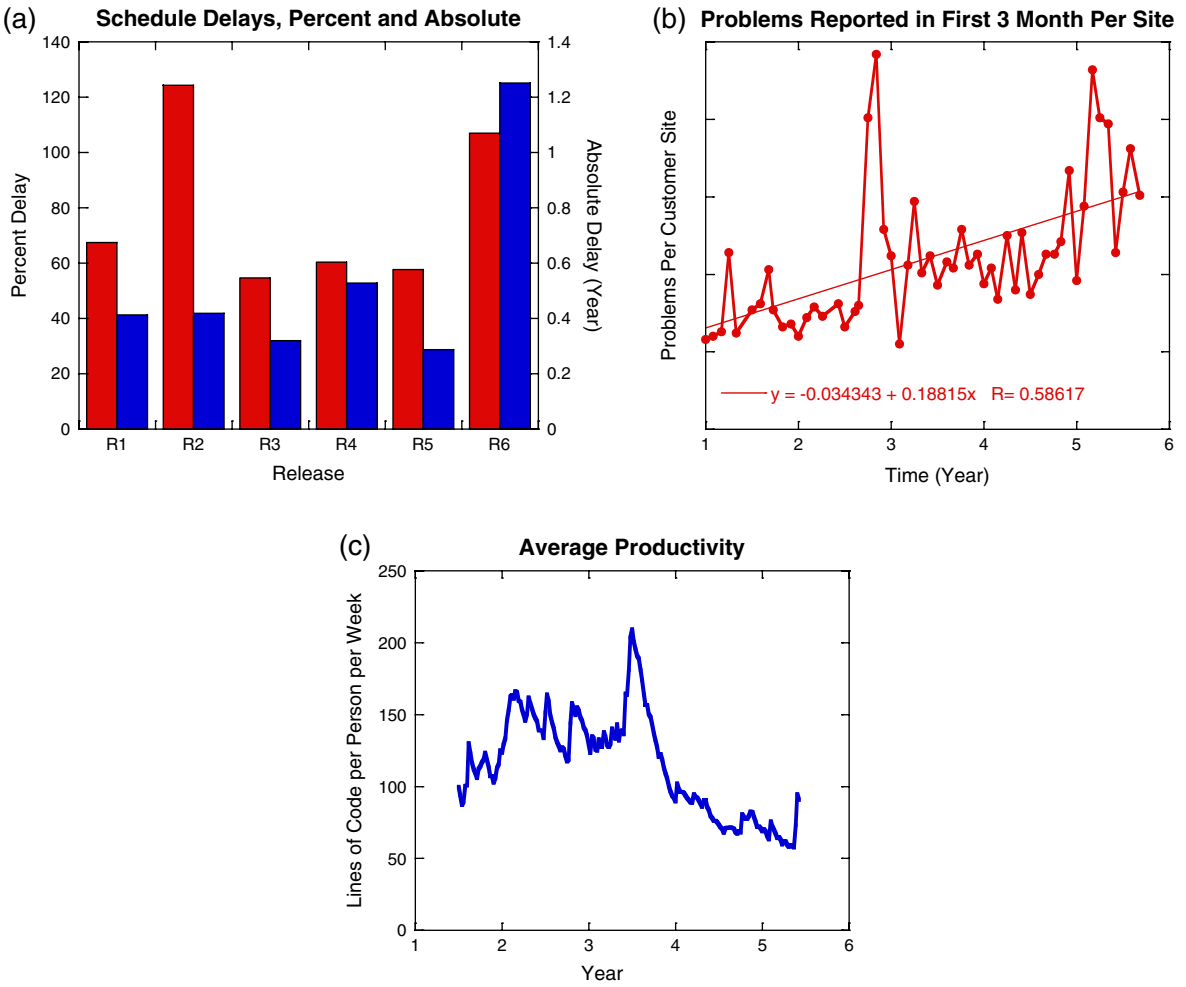

Figure 2. Alpha's performance on (a) percentage delay from original schedule (left) and absolute delay (right); (b) quality in bugs reported in first three month of installation (both axes are zero based, scale hidden for confidentiality reasons); and (c) Alpha's productivity, in new lines of code per person per week

significantly over the second half of the study period (2000–2005).

Although Alpha and Beta shared many attributes, including size, resources, and organizational and incentive structures, and both had access to similar knowledge bases, their product development practices and results diverged. The commonalities across the cases helped us in developing a generic model of the contributors to PD capability dynamics; the variations assisted modeling work by providing insights into different modes of operation that the common structure could generate. Next, we discuss the simulation model, how it relates to the two cases, and how its analysis informs the role of managers in managing PD capability.

MODEL AND ANALYSIS

We develop our theory in two steps. First, we focus on explaining the decline observed in the Alpha PD process across multiple releases. Building on a representation of the "physics" of developing software, we show how Alpha's management approach, while intended to be rational, created a phenomenon that we call the "adaptation trap"—a set of dynamics that trapped the development team in a vicious cycle of declining productivity and increasing demands, and eroded the PD capability. Second, we show how Beta's managers successfully avoided the adaptation trap and why Alpha was unable to learn from Beta's success.

DOI: 10.1002/smj

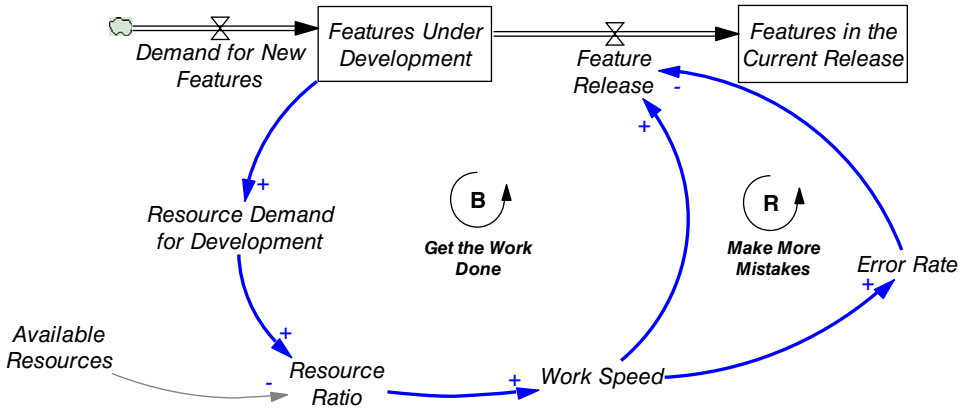

Figure 3. The basic stock and flow structure for multiple-release product development. Also included are the balancing ("get the work done") and reinforcing ("make more mistakes") loops

The rise and fall of Alpha: the adaptation trap

In contrast to Beta, Alpha's use of best practice has eroded with time. Over the period we studied, the level of requirement review declined, code review and unit testing increasingly suffered, and documentation of the code became increasingly incomplete. One experienced developer commented:

> "What I have seen is years and years of degradation—getting worse and worse over time."

In two steps, we explain how a strategically important capability managed by well-intentioned managers could reach such a state. We begin by showing how the basic physical structure of any work process, when coupled with the effects of work pressure on productivity, creates a system with a tipping threshold, a level of activity that, once exceeded, causes a sustained decline in performance. We then extend the model to capture how the dynamics change when the work process in question creates an installed base of products that can require additional attention (e.g., bugs that need to be fixed). The result is a system in which it is easy for managers to learn the wrong lessons concerning the appropriate level of resource loading, thereby inadvertently eroding the capability that they depend on.

Work pressure and tipping

Figure 3 shows the stock and flow structure and basic feedback mechanisms of our initial formulation. The stocks (boxes) represent the integration of different flows (thick valves and arrows) and help distinguish between a particular action or activity (flow) and its accumulated impact (stocks). Stock and flow networks lie at the foundation of any dynamic system (Sterman, 2000). The variable *Demand for new features*[1] represents the flow of requests for additions to the product. In both Alpha and Beta, new features were proposed by sales, services, and marketing groups and were largely controlled by higher managers rather than R&D staff (see Figure 3).

Newly proposed features accumulate in the stock of *Features under development,* remaining in this stock until they are developed, tested, and released in the market to become part of the *Features in the current release*. The flow *Feature release* captures the speed at which the PD organization develops and releases new features.

PD capability manifests in this setting as the ability to transform requests into defect-free features. This capability consists of a large number of individual and small group routines such as early emphasis on a robust architecture, detailed requirements, code reviews, automated and unit testing, and root-cause analysis of errors. Companies that successfully build and maintain this capability will sustain higher rates of feature release for a given resource base and, therefore, potentially achieve competitive advantage through some combination of more features, faster release times, higher product quality, and lower cost.

[1] The names of variables that appear in the model diagrams are italicized when they are first introduced.

DOI: 10.1002/smj

Given this representation of the system's "physics," we now turn to how the PD capability is impacted by management's decisions. While individual and team processes such as gaining experience and learning from peers are important early in the capability evolution process, our study period starts when the Alpha team had developed plenty of experience and had a well-functioning PD capability. In this period, managers' actions, and specifically the speed of work they required from their team (e.g., by increasing demand for new features), had a profound impact on the daily routines of Alpha. In tracing how that speed was set, our informants pointed to schedule pressure as a key driver (which we call "resource ratio"): the ratio of *Resource demand for development* given the current stock of *Features under development* and the *Available resources* (Figure 3). In principle, instead of requiring higher work speed, managers can add resources or cut the number of desired features to bring the demand for features in line with the team's ability to deliver them.

Alpha's options appeared to be more limited. Headcount additions could be counterproductive due to the associated learning curve and were not easily granted by senior management in a timely fashion. Similarly, feature requests were rarely eliminated, both due to the dictates of Sigma's leadership and Alpha's own desire to satisfy its customers. One of the managers explains the rationale behind the team's philosophy of attempting to respond to the majority of customer requests:

> *[The] customer is the one who is out there, and that's what you are focused on … if some customer comes, and I see I can sell to this customer if we do this thing, and there will be 7 million dollars of revenue, you would say yes.*

Given these constraints on team size and feature requests, Alpha's PD team reacted to increases in its workload largely through changes in its speed of work. Thus, when Alpha experienced a growing resource ratio, the team reacted by increasing *Work speed*. The increased work speed would, in turn, lead to an increase in the *Feature release rate*. Taken together, these links create the balancing "get the work done" loop; as the demand for new features grows, engineers work faster, release more, and thereby balance the surplus in the requested number of features. This loop gives the team a significant measure of flexibility in managing the inevitable variations in its workload. Work speed can be adjusted quickly, thus enabling the team to accommodate variations in its workload without constantly changing the delivery schedule.

Regulating variations in demand through work speed does have a potential downside. The increased speed often comes through shortcuts and less attention to detail, leading to inadequate requirement development, incomplete documentation, lack of code review, and poor unit testing (e.g., MacCormack *et al.,* 2003). Moreover, stress induced by sustained pressure can reduce individual productivity and increase absenteeism and turnover, further limiting output. If sustained over time, such changes in daily activities turn into changes in individual routines, and then group routines and organizational culture regarding quality standards. Thus, a sustained, high resource ratio can erode PD capability and hurt performance. In the words of a technical manager in Alpha:

> *There is … a lot of pressure put on people to say how can you do this as fast as you can, and people fall back on the ways that they used to do it, so instead of going through all this process stuff [they would say] "I know what to code, I can just code it up."*

In our model, as the resource ratio (i.e., work pressure) grows, the error rate (broadly defined to include anything that limits net productivity) increases, thereby slowing the net completion rate. As the completion rate slows, the stock of pending features grows (assuming a constant inflow of feature requests) thus feeding back to create even more work pressure. This new feedback, the "make more mistakes" loop, is reinforcing and amplifies any given disturbance. When working in the downward direction— a growing error rate, declining net output, and mounting resource ratio—the make more mistakes loop acts as a vicious cycle.

The interplay of these two processes dictates how the net rate of feature release evolves with changes in the resource ratio and thus represents a key assumption in our formulation. Consistent with psychological research (Mandler, 1984), and previous research in product development (Taylor and Ford, 2006) and software development (Abdel-Hamid and Madnick, 1991), our discussions with study

DOI: 10.1002/smj

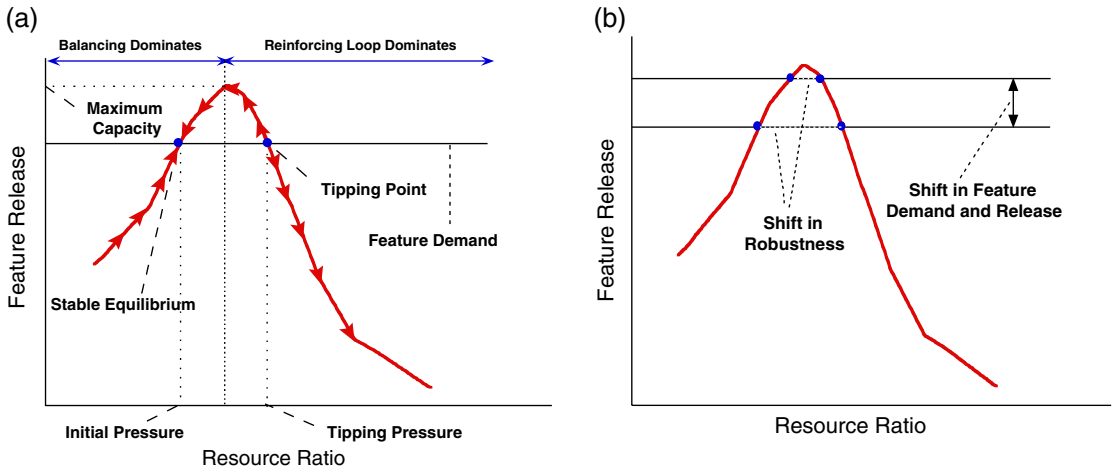

Figure 4. (a) Inverse-U shape curve of feature release as a function of resource ratio. (b) Increasing feature demand increases feature release (up to maximum capacity); it also reduces robustness (the distance between stable equilibrium and tipping point)

participants pointed to an inverted U-shape relationship between the resource ratio and the net output. At modest levels, a growing resource ratio improves net productivity as the positive effect resulting from working faster offsets any increase in the error rate. However, with increased pressure, the diseconomies of speeding up (Dierickx and Cool, 1989) dominate, causing the net effect to become negative.

Building on these formulations, Figure 4(a) shows a characterization of the system's dynamics. We begin with the inverted U-shape curve that relates the resource ratio to the rate of feature release and shows both the system's equilibria (steady states) and the dynamics around those equilibria. The system has two equilibria, each of which occurs at the intersection of the rate of feature demand and the inverted U. These equilibria represent the points at which the resource ratio induces just enough work effort to meet the current level of feature demand. Once the system reaches either of these points, it will be in steady state (i.e., equilibrium) with the rate of feature release equaling the rate of feature demand. Although both points represent equilibria, the dynamics around them are very different. The equilibrium that appears on the upwardly sloping portion of the inverted U is stable, meaning that small perturbations from it are offset by the system's dynamics. For example, a small increase in the resource ratio (perhaps due to a temporary increase by management in feature demand) would cause the feature completion rate to increase above the steady-state feature demand rate. That would be manifesting as a temporary excursion off the stable equilibrium and up the inverted U. Thus, the stock of features pending will reduce, bringing the system back into balance and into using the high-quality practices. Formally, in this region, the balancing get the work done loop dominates the system's dynamics and functions exactly as managers desire, constantly working to bring the level of outstanding work and the available resources into balance.

In contrast, the equilibrium that occurs in the downwardly sloping portion of the inverted U is unstable, implying that any deviation off that equilibrium, rather than being offset, will be reinforced. As indicated by the arrows on the inverted U, the system is unlikely to ever settle at this point since even the smallest perturbation will push the system on to a new trajectory, away from the unstable equilibrium. Due to these dynamics, the unstable equilibrium is often called a "tipping point," capturing the idea that once an unstable equilibrium is crossed, the system's dynamics can change significantly.

Consistent with this intuition, the tipping point plays a key role in determining the dynamics of the system we study. Figure 5 shows an experiment in which the feature demand rate has been temporarily increased, e.g., due to requests of customers or the management. In the first experiment (Figure 5a), the shock is not sufficient to push the system over the tipping threshold and, consequently, the system

DOI: 10.1002/smj

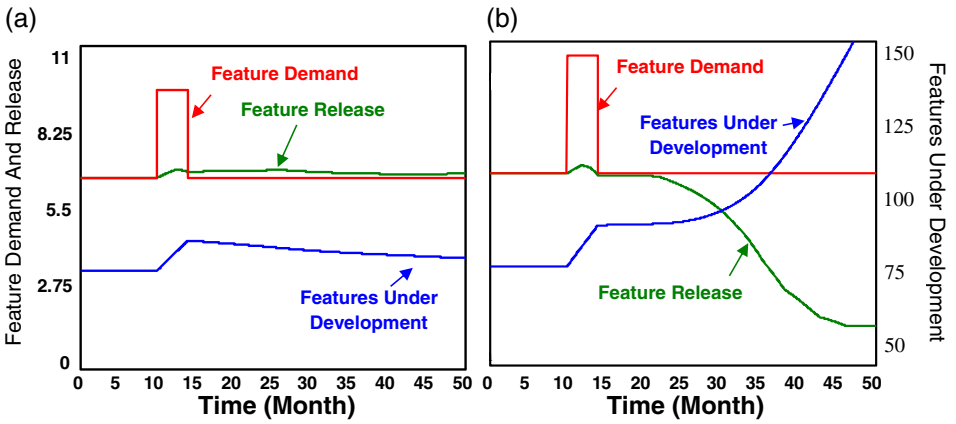

Figure 5. The reaction of simulated PD system to two slightly different demand shocks: (a) The system recovers through the balancing "get the work done" loop. (b) Tipping point is crossed and the reinforcing "make more mistakes" loop dominates, thus performance deteriorates

is able to absorb the shock and return to its initial performance level (as measured by the feature release rate). In the second experiment (Figure 5b), in contrast, the increase is large enough to push the system over its tipping threshold. Here, rather than returning to the initial equilibrium, the system gets stuck in a vicious cycle of growing feature demand, mounting resource ratio, and a declining rate of feature release as the organization increasingly moves away from good (long-term) practices. As the experiment indicates, when the system crosses its tipping threshold, performance declines significantly and does not recover. Importantly, the decline in performance is due to changes in day-to-day PD processes that compromise long-term quality and productivity, gradually changing PD team's routines and, in short, eroding the PD capability.

Three key features emerge from analysis of the model so far. First, due to the existence of the tipping threshold, the organization's response to temporary shocks is nonlinear. Shocks that do not push the system over the tipping threshold are offset and only require a temporary departure from good development practice; absent an additional intervention, shocks that do exceed that limit cause a permanent decline in performance and capability. Second, the size of the shock required to push the system over the tipping threshold is determined by the distance between the stable equilibrium and the tipping point. When the distance is large, the system can absorb a substantial shock; when it is small, a modest perturbation can push performance over the edge. Third, the distance between the two equilibria is a function of the steady state rate of feature demand. As shown in Figure 4(b), as the rate of feature demand (horizontal line) requested by customers or management rises, the two equilibria move up the inverted U and the distance between them shrinks. At very high feature demand rates, both equilibria disappear and the system's dynamics are dominated by a downward capability-eroding spiral. Thus, managers in this system face a clear trade-off between performance and robustness: As they demand more features, performance continues to improve (as long as they stay within the inverted U). However, as feature demand rises, the size of the shock required to push the system over its tipping threshold declines. This is a general result that holds as long as mounting work pressure eventually leads to a decline in productivity.

Alpha's performance clearly declined over the period of study, and a vicious cycle of accumulating outstanding work, mounting work pressure, and declining performance was a major factor—one team member described it as "ever increasing pressure to deliver higher quality in shorter intervals with fewer resources—it seems never good enough." Moreover, the interviews strongly suggest that the declining performance resulted from changes in the way the work was executed. Several developers noted that, due to schedule pressure, the Alpha team increasingly abandoned the elements of good software practice such as documentation in favor of getting the work done more quickly. One developer reported:

> *Code review is not formal here … in my view it should be formal … that would be great,*

DOI: 10.1002/smj

but honestly I can't imagine doing that here because it takes too much time … I would love to do it, because that is an obvious way to improve the quality, but [for] the time it would take to do it.

The effect on the team's morale was predictable:

… people are frustrated … everybody is under pressure, and specially as it gets closer and closer, we get upper management say I want you to work weekends, I want you to work seventy hours a week, and I want everybody to work from home till it is dawn, I don't want to come any hour of the day and find the place empty, and people say gee I am working my ass off and I am not getting the raise, they cancelled my vacation, what do I get out of it, when it is done there is another release right behind it, and I have to go through the same thing again. You end up in survival instinct, so it is like hang out and survive … all this would definitely affect the quality.

Alpha's capability clearly suffered the consequences of this increasingly ad hoc and high-pressure mode of execution. While it started well, Alpha's quality declined significantly, and later releases were plagued with bugs requiring additional resources for remediation. One might have expected the Alpha team to recognize that they were tipping into a vicious cycle and to have reduced the emphasis on speed accordingly. Our interviews pointed to two complementary reasons why such response was not adopted by Alpha. At one level, the management's mental frame for analyzing and responding to schedule pressure assumed a positive, albeit diminishing, relationship between required speed and actual output. The simple decision rule of asking for more to get more was intuitive and easy to act upon, which partly explains its persistence. A second mechanism for persistence of this heuristic relates to how the managers learn in such settings, which we elaborate on in the following section.

The adaptation trap

The impact of errors in development goes beyond the immediate rework they require when discovered. Some errors escaped the quality assurance process and remained in the software shipped to customers. When discovered by customers, these errors would be referred back to the R&D department. If the problem interfered with an important customer need, R&D would assign engineers to fix the problem, usually through quick, customized patches at the customer site. Such current engineering (CE) could take a significant share of R&D resources: Alpha allocated 30–50 percent of development resources to CE work; Beta, in contrast, kept this number below 10 percent.

Current engineering introduces two important wrinkles to our analysis. First, CE requires additional resources. Second, those resource requirements can only be known with a significant delay. Customers must buy, install, and use the new release before CE resource needs are revealed. In this section, we show how these two features make the challenge of learning to appropriately balance resources and demands far more difficult, and thus the capability erosion dynamics more likely.

Figure 6 shows the structure of our expanded model.[2] The flow of feature release now accumulates in a stock of *Features in the current release,* representing the installed base of code released to Sigma's customers. A new flow, *Defect introduction,* now runs in parallel to feature release, and captures the stream of defects in new features that are not identified by in-house testing. The defect flow accumulates in the stock of *Defects in the current release,* representing the stock of defects in software currently being used by customers.

Defects in the field are eventually discovered and create additional demand for PD resources via current engineering. The demand for CE increases the schedule pressure and thus the error rate. These new links add an additional reinforcing feedback to the model, the Current Engineering loop. As the stock of defects grows and customers request additional current engineering, developers face even more demands on their time and must work faster to meet their delivery targets. As we discussed in the previous section, the increased work speed can result in an even greater error rate, causing the stock of defects in the field to grow further. Note that the strength (or gain) of this loop is potentially high as current engineering is often extremely costly relative to fixing problems before they are released

[2] Detailed model equations, consistent with model reporting guidelines (Rahmandad and Sterman 2012) are available in File S1.

DOI: 10.1002/smj

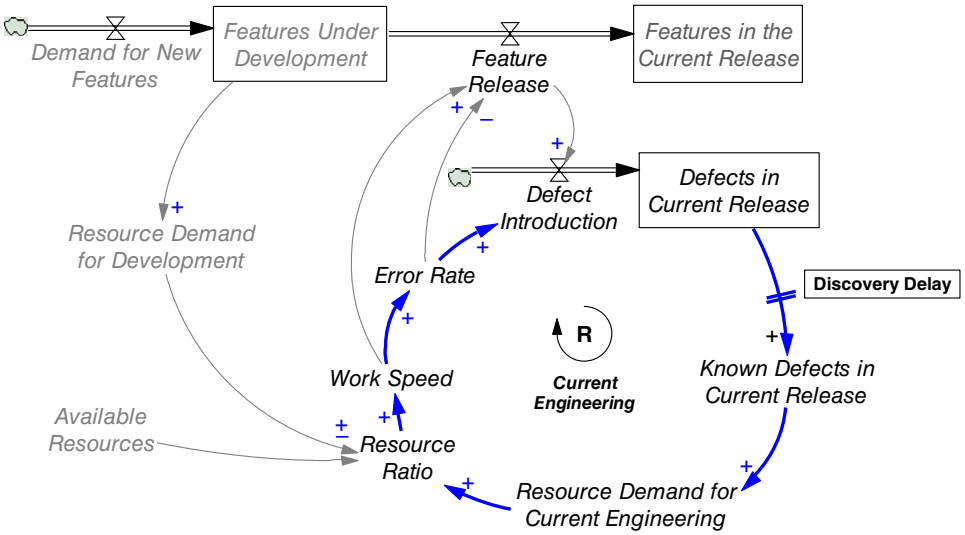

Figure 6. The structure of the model with the addition of products in the field. The new structures are highlighted through darker variable names and thick causal links

(Boehm, 1981) and is usually resolving the problem of a single customer. As one developer reported: "When somebody goes to fix a defect, they can write just as much code [as new development]."

While Alpha's PD team had little impact on feature demand rate (thus treating feature demand as exogenous in Figure 3), regulating this demand rate (as well as project deadline and resources) was the main lever managers had at their disposal for getting the best possible result from the PD organization. The addition of current engineering and the delays associated with perceiving its full impact increases the challenge facing those managers for two reasons. First, CE requires additional resources and thus reduces the capacity of the organization to develop new features. This shifts the inverse U-shape graph in Figure 4 down. Accounting for the magnitude of the shift creates a second challenge. Consider the decision to increase the rate of feature demand: Due to the delay in perceiving the increased CE demand, in the short run, the system behaves as though no additional errors (from the additional features) make it to the customer sites. Therefore, the delay increases the apparent capacity of the system. In fact, as the fraction of errors that goes undiscovered grows, so too does the apparent capacity of the system to do additional work (since resources do not appear to be needed for rework) (see Figure 7). In other words, in the short run, the system will behave as though it has more capacity than is actually available.

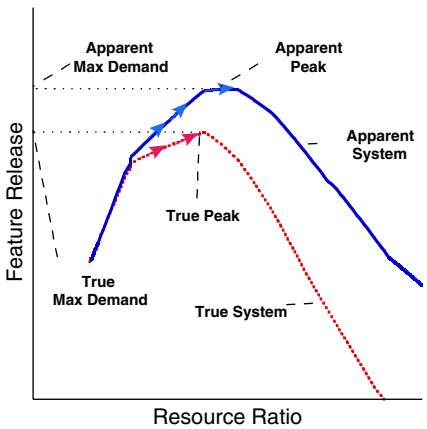

Figure 7. The curves for apparent (solid) and true (dotted) feature release vs. resource ratio. In the apparent curve, the demand for CE is kept at its short-term value despite adaptive increase in feature demand. The sustainable path comes from the same experience, where the CE resource demand is given based on the long-term level of error rate and feature release

This feature of the system—that its short-term behavior differs significantly from the long-run performance—makes learning difficult. Recall that the distance between the stable equilibrium and the tipping threshold is determined by the feature demand. Increasing the rate of feature demand makes the system more susceptible to a descent into destructive firefighting. Thus, the task facing managers is to trade off the higher level of performance that comes with a higher feature demand against

DOI: 10.1002/smj

the decline in the system's robustness to temporary shocks.

Such a decision is not particularly amenable to formal analysis. The organization's true capacity is difficult to estimate as is the workload associated with a given set of features. Managers are thus likely to engage in local search, progressively increasing the workload through more feature demand or tighter deadlines, and monitoring the performance of the system to determine whether the increases yield a higher rate of feature release. Such local search processes are central to how organizational routines adapt and create better performance (Nelson and Winter, 1982). Tightening the schedule to get the maximum feasible performance was a common strategy in Sigma. A program manager comments:

> *The thinking that [a higher manager] had, and we think we agree with this, is that if your target is X and everybody is marching for that target, you fill the time for that target. If you are ahead of the game, people are probably going to relax a little bit: because we are ahead of the game, we don't need to work so hard, we can take a longer lunch, we can do whatever; and guess what? Murphy's law says you now won't make that target. So […] you shoot for X minus something, you drive for X minus something so that you build in on Murphy's law to a point.*

Localized search coupled with a system that displays different short- and long-run dynamics leads to the notion of the "adaptation trap." Consider Figure 7 and imagine a manager attempting to set the appropriate balance of work and resources by incrementally increasing the rate of feature demand. In the short run, the system will behave as described by the upper inverted U (which we label the apparent system). Such local search will lead to a level of loading somewhere near the peak of the apparent curve. However, if the true long-run behavior of the system is described by the lower curve, then the adaptation process will have expanded feature demand beyond the sustainable capacity of the PD organization. When the additional CE work does eventually start to appear, the PD organization will find itself past its tipping threshold and on the slippery slope of firefighting. The tight schedule and high feature demand, which initially seemed so effective, will cause the rate of feature release to fall below feature demand. That leads to continued growth in the stock of features to be developed, and thereby ensures that pressure stays high and that the organization remains in the firefighting mode. Thus, in contrast to early experiments in which an exogenous shock was required to push the system over the edge (and previous analyses of tipping behavior in organizations such as Repenning, 2002, and Rudolph and Repenning, 2002), the adaptation trap notion suggests that such a decline can occur endogenously, resulting from the well-intentioned efforts of managers to optimize their system. Due to delays inherent in getting feedback, managers do not receive complete information about the results of their experiments, leading them to systematically overload their development teams.

The existence of the adaptation trap depends on managers not fully accounting for the delays in the system. Were they to fully account for delays in assessing the consequence of each incremental increase, they would be less likely to overshoot. At Sigma, the calculation of expected resource needs for CE was far less complicated: development teams were expected to dedicate no more than 10 percent of their resources to CE activities. This rule was integral to resource allocation across projects. We found the application of this heuristic was not challenged in the presence of contrary data, let alone modified in recognition of delays involved in perceiving CE needs. Experimental research, in which subjects have repeatedly been shown to both underestimate time delays and fail to properly account for their impact on a system's dynamics (e.g., Diehl and Sterman, 1995; Sterman, 1989), further supports the core adaptation trap mechanism.

Beta's success in avoiding capability erosion

The previous discussion identifies the cognitive heuristics and consequent dynamics that can result in capability erosion, thus raising the question of why Beta did not fall prey to the dynamics that were so damaging to Alpha. The first-order answer is simple; in contrast to Alpha's emphasis on flexibility and immediate efficiency, Beta was not willing to adjust its PD routines when the demand for new features exceeded its ability to deliver them; the new features simply had to wait. A manager explains:

> *In [Beta] … we have lots and lots of huge pressures … [but] we don't skip process,*

DOI: 10.1002/smj

because that is the tone we set … I have had people who skip process and I have had them fired, that it is not a platitude, I insist.

Interestingly, though, we heard similar statements from managers of the Alpha team, and they made repeated attempts to improve their use of best practice. Yet, despite their best intentions, the Alpha team appeared incapable of making the difficult decisions necessary to manage their resources effectively. Comparing the two projects suggests that, whereas both projects "talked a good game," only Beta was able to translate the desire to match resources to demand into specific operational practices. Three such practices appeared to be critical in avoiding the adaptation trap dynamics that proved so damaging to Alpha.

First, in contrast to Alpha, Beta dedicated a separate team, comprising about 10 percent of its developers, to current engineering. The fungibility of resources among the different phases of development is a key source of the dynamics we study. The positive feedback at the heart of our analysis becomes dominant when scarce development resources are pulled away from high productivity early-phase development work and dedicated to low productivity current engineering work. Beta's policy of fixing the allocation of resources to low productivity CE work greatly weakened this feedback and made their system far less prone to tipping. Note, however, that this policy restricted Beta's flexibility and short-term effectiveness. If CE demand exceeded 10 percent, then limiting the allocation of resources to CE could result in numerous unresolved customer complaints and negatively affect demand for the product; if CE demand was less than 10 percent, valuable engineering resources would be wasted.

Second, the Beta leadership enforced a "no late feature" rule. While the Alpha team frequently accepted changes in requirements far into the design phase—thus necessitating rework and further process shortcuts—Beta's managers resisted the pressure to add features once design started. The ability to match demands with resources allowed Beta to execute most of its work following best practice (i.e., few shortcuts) resulting in a low error rate and a small amount of CE. Again, note that this policy restricted Beta's ability to respond to its market. If a valued customer made a late feature request, Beta was not able to meet it until the next release.

Finally, Beta's approach to learning differed significantly from that employed by Alpha. Beta often capitalized on the learning opportunity provided by the defects it did discover. Whereas Alpha reacted to a new field problem by trying to fix it quickly so that they could get back to their backlog of new features, the Beta team frequently took the time to trace an error back to a flaw in their development process and then adjusted that process accordingly. Results of those analyses could impact personnel review and incentives, particularly if developers were found to have departed from best practice, thus strongly communicating the behaviors that Beta valued from its members. While the Alpha team members were often implicitly rewarded for getting things done quickly, the Beta members were explicitly penalized for such (apparently) time-saving departures.

Each of Beta's interventions appear, at least *ex post*, to be smart but also somewhat obvious ways to insure that the development team continued to maintain its capability. The relative simplicity of Beta's practices and the consequent results—Beta was regarded as one of the best managed projects in Sigma—raises the question of why did Alpha not follow suit? The first part of the answer is simply that it probably wasn't as simple as it looked after the fact. A long line of experimental and theoretical research highlights the challenges of learning in dynamically complex settings (Brehmer, 1992; Rahmandad, 2008; Repenning and Sterman, 2002), and the dynamic complexity arising from the tipping point and the adaptation trap would make it hard for any manager to learn the right lessons from their experience and avoid capability erosion. But, the dynamic complexity of software development notwithstanding, Beta managed to figure it out, and Alpha did not.

Beta's success in avoiding erosion appears to turn on a combination of luck and a willingness to learn from experience. By chance, some of the key managers in Beta had prior experience with capability erosion dynamics. These managers report that those experiences significantly influenced their belief in the utility of adhering to good practice. In contrast, members of the Alpha team reported few such formative experiences. One of the Beta managers recalled:

Early in my career … when I was in an organization that did wireless … we were under tremendous market pressure to get the product out … and you just do anything you

DOI: 10.1002/smj

> *can to get anything out, not one comment was put in a lot of code, not one thing was inspected, not one thing was documented, you know, it was classic, and it never worked, and still to this date it is in the field, and it has problems … and that's where I really cut my teeth and learned my lesson.*

Beta was also lucky in another way. Two times during the history we studied, Beta's parent product experienced a significant delay even though the Beta team was on track to hit the original schedule. Those delays provided the Beta team with time to reflect, improve their processes, and prepare for the next releases. These delays pulled Beta farther below its tipping threshold, allowing them to increase the effectiveness of their work. One of the Beta managers reflected on how this externally imposed delay allowed the team to reinforce its capability:

> *How we got here? The answer is in investing in [the] future. And from the [Beta's] perspective we got lucky. If I go back to the first release [of Beta], … they [the parent product] were running into many more schedule problems than we were … this allowed us to go back and actually do a lot of things that we would have preferred to have done at the beginning but we never got the time to do … So we benefited by that additional time to really add a lot of foundational components.*

The delays not only helped Beta place more focus on the high-productivity front end of the development process, but the success of those efforts further reinforced the linkage between good practices and good performance.

In contrast to Beta's ability to learn from the experience of its members and capitalize on its good fortune, three factors likely conspired to prevent Alpha from learning a similar lesson. First, the project was staffed by a younger, less experienced team, many of whom had never been associated with a major failure. Second, the project was initially quite successful, thus reinforcing the team's belief that they were on the right track and did not need to revisit their approach to developing software. Finally, note how the system's dynamics caused the cost of adopting Beta's practice to rise over time. Beta dedicated a fixed team to current engineering, about 10 percent of its resources, but as Alpha's performance declined its CE demand often fluctuated to as high as 50 percent of its total activity. Creating a dedicated CE team would have dramatically reduced the team's ability to generate new features and threatened the viability of the project. Similarly, the growing number of quality problems eroded Alpha's bargaining position with customers, making enforcing a no late feature rule increasingly infeasible. Finally, with hundreds of errors reported from the field, conducting root-cause analysis was increasingly costly, undermining incentives for high-quality work. Thus, even if Alpha did eventually come to realize the wisdom of Beta's approach, as the project continued, capitalizing on those insights became increasingly infeasible.

CAPABILITY EROSION AND STRATEGY

Explaining the existence of persistent performance differentials in similarly positioned firms remains a challenge facing strategy scholars (Gibbons and Henderson, 2010). A growing body of literature tackles this question by detailing the difficulties that organizations face in finding their way to the optimal configuration of resources and capabilities required by their chosen context. Research by Levinthal (1997), Rivkin (2001), and others shows that performance landscapes can be quite "rugged," complicating the discovery of optimal configurations. Moreover, from low exploration (March, 1991) to hot stove effect (Denrell and March, 2001), and dominance of local search resulting in competency traps (Levitt and March, 1988), existing research documents multiple mechanisms through which learning can be inefficient in exploring these rugged landscapes. While powerful in many situations, this paradigm is less well suited to explaining cases in which key elements of the optimal configuration are well documented and easily accessible but somehow remain unused. In many industries, the practices associated with success are both well understood and well established, and a growing line of research shows that adding even simple capabilities can lead to substantial improvements in performance (Bloom *et al.,* 2013). In the language of rugged landscapes, these studies suggest that there is visibly higher ground in close proximity, and yet many firms are not successful in their efforts to maintain those positions.

DOI: 10.1002/smj

Following the seminal work of Dierickx and Cool (1989), reintroducing erosion offers the possibility of a more complete theory of differential performance rooted in different capability trajectories. The first contribution of our study is to show that capabilities can erode on a strategically relevant time scale. Alpha was a highly successful product and initially viewed as central to Sigma's future growth and success; less than a decade later, it was put on hold due to poor performance. Capitalizing on the observation that erosion can influence competitive outcomes requires a theory that details both the conditions under which it is likely and what separates those firms that avoid it from those that do not. While we are far from a complete account, the second contribution of our study is to both identify a potential erosion mechanism and a set of testable conditions under which that mechanism is likely to exist.

The erosion mechanism emerging from our study starts with the notion of capability as a locally stable configuration of interaction patterns among organizational members that accomplishes certain tasks. These patterns are embedded in individual memories, organizational routines, and the physical structure of tasks. Capability erosion is then understood as a set of endogenous dynamics that take an organization from an efficient capability configuration to an inferior one. Prior theory acknowledges that organizations will often temporarily deviate from such configurations due to environmental shocks, turnover, or inefficient organizational memory systems (Argote, 2012). Existing theory implicitly assumes, however, that with time such departures will be offset and therefore have little competitive significance. Our study indicates that, under the competitive pressures firms commonly face, three features can combine to create a system in which such minor departures, rather than being offset, are amplified, and thus have the potential to yield a competitively inferior position.

The first requirement is the existence of an unfavorable temporal trade-off when departing from the optimal configuration. In the setting we study, software engineers could improve short-run productivity by skimping on review, testing, and documentation, but this decision caused subsequent quality problems. This type of trade-off has been identified in a variety of situations, including maintenance (Carroll *et al.,* 1997; Zuashkiani *et al.,* 2011), manufacturing (Repenning *et al.*, 2002), new product development (Repenning, 2001), and financial services (Oliva and Sterman, 2001), but its strength is likely to vary widely across settings. In the case we study, the trade-off was acute. Good software practice dictates a variety of activities that can easily be skipped in favor of "… just writing the code." However, code written without attention to the overall architecture, continuous review and testing, or clear documentation often contains a large number of bugs, which must eventually be fixed. A variety of studies clearly show that, in the long run, it is far more efficient to do it right the first time (Boehm, 1981).

Not every configuration, however, may be subject to the same trade-off. For example, while studies suggest that preventive maintenance is useful for many types of sophisticated manufacturing equipment (and thus subject to an unfavorable temporal trade-off), the trade-off is often nonexistent for easily replaced parts like light bulbs—it is optimal to just replace them when they burn out. When there is no possibility of boosting current throughput at an implicitly unfavorable cost, then the erosion mechanism we study does not exist. Thus, our first proposition is that, all else being equal, the stronger the unfavorable temporal trade-off, the more likely that erosion will play a role in determining competitive outcomes.

The second requirement is resource fungiblity, meaning that the same resources are used to address both sides of the temporal trade-off. In the Alpha project, resource fungibility was high as engineers could move easily from new work to fixing bugs. The combination of the unfavorable temporal trade-off and fungibility creates the self-reinforcing feedback process at the heart of our analysis. In the Alpha project, when engineers skimped on good development practice, thus departing from the optimal capability configuration, they created bugs they themselves eventually had to fix; as the resources required for bug-fixing grew, the engineers had even less time to work on new features, thus necessitating further departures from good practice and pushing the organization further from the optimal capability configuration. Much like the first proposition, software development is a strong case in point: resources are often perfectly fungible, with development teams working on everything from developing new features to fixing issues at specific customer sites.

Fungibility may be less present in other contexts and thus make the dynamics we discussed less likely. For example, heavy manufacturing, such as

DOI: 10.1002/smj

car production, probably represents an intermediate point on the fungibility continuum. When problems appear in the field, engineers need to develop a fix and insure that the underlying defect is eliminated in future production. In contrast to Alpha and Beta, however, engineers in this context rarely go fix individual automobiles. Even further along the fungibility continuum, many development environments do not offer the possibility of rectifying past mistakes and thus present no trade-off. Fashion goods, for example, often miss the mark and do not sell well, but once a product is released, designers (unlike software engineers) cannot issue "updates" or "bug fixes" to rectify their earlier missteps. We would not expect to see the erosion mechanism we study in this setting. Our second proposition is thus: all else being equal, as the fungibility of resources across the unfavorable temporal trade-off increases, erosion will play a more significant role in determining competitive position.

The final condition concerns the delay between the "better" and the "worse" part of the temporal trade-off. Organizational research at a variety of levels highlights time delays as a critical variable. At the micro-level, delays between action and outcome have been shown to hamper learning (Denrell, Fang, and Levinthal, 2004; Rahmandad, 2008; Rahmandad, Repenning, and Sterman, 2009), and both their duration and impact are often underestimated (Sterman, 1989). At the organizational level, delays have been implicated in a variety of pathologies (Azoulay, Repenning, and Zuckerman, 2010; Hall, 1976; Perlow, Okhuysen, and Repenning, 2002; Repenning *et al.*, 2002). In the setting we study, the delay between departing from an optimal configuration (abandoning good development practice) and realizing its full consequence (bugs reported in the field) greatly increases the complexity of the manager's task. They must both trade off an immediate benefit (more code) against a future cost that will not be revealed for weeks or even months, and anticipate any additional resources that the initial departure might eventually require. The adaptation trap results because managers search for the optimum resource utilization faster than they receive feedback about their past decisions. As the delay between the better and the worse parts increases, so too does the self-reinforcing process that erodes capability and drives the organization away from the optimal configuration.

Sigma probably represents an intermediate case on the length of possible delays between better and worse behaviors. Development managers at Sigma were able to see the quality of their teams' work over the course of 6–12 months. The delays are much longer in other industries. In the auto industry, for example, it can take several years to develop a new vehicle and then at least one more year before the true quality of that vehicle has been revealed (consider Toyota's turn of fortune in 2009, likely the result of product development problems many years before). On the other side of the continuum, website developers can often get feedback on their work in a matter of days or even hours. The mechanism we study is thus likely to be more relevant in explaining differential performance in the auto industry than in website development. Our third proposition is that, all else being equal, as the delay between better and worse grows, erosion plays a more significant role in explaining outcomes.

Our analysis thus suggests that the influence of capability erosion on competitive outcomes is contingent upon the nature of the configurations that the industry in question requires. For configurations characterized by unfavorable temporal trade-offs, highly fungible resources, and long delays in realizing the consequences of departures from optimality, rates of capability erosion can be high and therefore contribute to differential competitive outcomes. In contrast, for configurations that are not characterized by strong trade-offs, where resources are not fungible, and/or where managers received rapid and complete feedback on the costs and benefits of departures, the erosion mechanisms identified in our study are less likely to play a significant role.

Implications

Beyond positioning future researchers to test the impact of erosion on competitive performance, our study offers three additional implications for the ongoing search on the sources of differential performance. First, the existing literature recognizes that a capability can eventually become a liability. Established capabilities focus managers' attention on a limited set of cues and may blind them to disruptive changes in the environment (Porac, Thomas, and Badenfuller, 1989; Tripsas, 2009). Firms not locked into those capabilities may find it easier to identify the need for change and adjust their orientation towards the distant, more fruitful, capability bundles required by the new environment. In a similar vein, some capability configurations may be more difficult to maintain and therefore more prone to

DOI: 10.1002/smj

erosion than others. Existing theory is largely silent on the relative fragility of different capability configurations, but there is good reason to believe that they are not all created equally. Vastly different forgetting rates across different production processes (Benkard, 2000; Darr *et al.,* 1995; Thompson, 2007) and variation in safety incident rates across organizations (Perrow, 1999; Roberts, 1990; Sagan, 1993; Weick, Sutcliffe, and Obstfeld, 1999) could be manifestations of different capability erosion rates and not empirical anomalies.

This conceptualization opens several avenues for future inquiry. For example, several scholars have documented the benefits of the so-called high-commitment work practices (Berg, Kalleberg, and Appelbaum, 2003; Gittell, Seidner, and Wimbush, 2010; Ichniowski and Shaw, 2003), a suite of practices that sometimes combine to produce exemplary performance. These gains are driven in part by the willingness of employees to go beyond the "letter of the contract" in return for above-average wages and/or job security. Simple game theoretic models, however, suggest that deviations by management from such "relational contracts" could cause that configuration to unravel. Consistent with this prediction, there are several examples of firms that struggled to maintain this configuration (Rubinstein and Kochan, 2001). Similarly, scholars have amply documented how the choice of technology shapes the organizational processes that surround it (e.g., Barley, 1986; Black, Carlile, and Repenning, 2004), and technology choices clearly influence the optimal capability configuration. Future research may be profitably focused on the question of whether some choices are more stable than others. For example, automating production tasks might initially be seen as removing the human element and therefore yielding a configuration less prone to erosion. However, sophisticated automation often requires more sophisticated maintenance and thus strengthens the unfavorable temporal trade-offs discussed above. Therefore, the net outcome is unclear: is erosion more, or less, likely in high technology settings? A more nuanced understanding of erosion dynamics is needed to shed light on the role of capability resiliency in determining competitive outcomes as technology mediates the transformation of different industries. Studying erosion dynamics may also inform related phenomena. For example, safety researchers continue to debate whether tightly coupled systems will always be prone to failure due to unforeseen interactions among system components (Perrow, 1999; Sagan, 1993), or can such modes be mitigated by collective mindfulness that allows early detection of failure modes (Roberts, 1990; Weick *et al.,* 1999)? A systems perspective for understanding safety structures in organizations, the vicious loops potentially embedded in them, the dynamics that erode those safety capabilities, and the management's role in controlling these systems could be essential for bridging the different perspectives on organizational safety (Leveson *et al.,* 2009).

Our second implication concerns the role of managers in offsetting erosion. Managers play an increasingly central role in the current accounts of organizational strategy, and a growing body of research focuses on the role of managerial cognition in perceiving and reacting to environmental change through the investment in and deployment of alternative capabilities (Eggers and Kaplan, 2009; Nadkarni and Barr, 2008). Existing capabilities are often implicated in the failure to make such adjustments as past success narrows the search for new configurations and creates inertia. While strategic change and renewal (Agarwal and Helfat, 2009) are essential for understanding firm heterogeneity, erosion may also place a premium on managerial ability. The situation we study is dynamically complex (i.e., characterized by multiple interacting components, significant nonlinearities, and appreciable time delays) and much experimental research demonstrates the inability of many human subjects to understand and control such environment (for summaries, see Cronin, Gonzalez, and Sterman, 2009; Sterman, 1994, 2000). There is thus good reason to believe that, even without a significant environmental shock, managers may differ in their ability to avoid erosion. Beta's management did so quite successfully, Alpha's did not.

Focusing the attention on the ability (or lack thereof) to manage the dynamics of erosion opens a new path for pursuing the question of persistent performance differentials in seemingly similar enterprises: What separates those managers who avoid erosion from those who do not? It may be that the ability to manage such dynamics is an innate character trait. Alternatively, experience and training may offer more explanatory power. On this score, the Beta project offers a tantalizing prospect. Learning in complex dynamic systems is often most effective when managers have experienced relevant cause and effect relationships in extreme, far-from-the-equilibrium conditions

DOI: 10.1002/smj

(Sterman, 2000). When queried as to the sources of their ongoing success, Beta's leader referenced experience with a project that failed spectacularly and the necessity of learning the value of good practice "the hard way" (i.e., through failure). Alpha's managers, in contrast, because they were younger and started outside the company, had fewer such experiences. Thus, a significant past failure, because it clearly revealed the links between departing from the optimal configuration and the subsequent performance decline, may have been central to the Beta managers' current success. Recent research supports this view, suggesting that the transfer of experience across similar situations is critical in understanding complex dynamics (Gary, Wood, and Pillinger, 2012).

Note, however, that most organizations will actively select against managers who have been associated with such extreme failures. Certainly, as was the case with Beta, some will survive and their hard-won skills will eventually allow them to thrive. However, this is likely to be the exception rather than the rule. The utility of "performance management"—a current euphemism for selecting out lower performers—has become an almost unchallengeable assumption in modern management practice (e.g., Spence, 1976; Welch and Welch, 2005). And, while such selection may insure that the remaining managers have a high propensity for effort, our analysis suggests that it is less clear whether the survivors have the best understanding of the systems that they are managing. Future research could thus be profitably directed at testing the hypothesis that past experience with significant erosion may actually equip managers to avoid it in the future. If this hypothesis is borne out, then the ability to avoid capability erosion is likely to be a competency that is both valuable and rare since those with the experiences that create such competency will often be selected out of the population.

The final implication of our work is methodological. Whereas the formal models in the strategy literature are often inherited from economics or biology (e.g., NK models such as in Levinthal, 1997) and are thus quite abstract, in this study we build a formal dynamic model grounded in the field observations of capability dynamics. Our approach offers several benefits. We are both able to connect elements of our theory directly to the challenges faced by real managers and the characterization of the system's dynamics provided by our model enables us to map the results of experimental studies of cognition in dynamic systems into actual organizational practice. In doing so, we generate hypotheses concerning which capabilities will be difficult to sustain. Formal models grounded in case data help capture how important features of the technology setting interact with managerial cognition and learning to create strategically relevant mechanisms such as the tipping and adaptation trap. Simulation further enables formal analysis of the resulting temporal complexity, something that would not be feasible using qualitative methods (March, 2001). This approach allows us to uncover capability erosion as a relevant theoretical construct that has been absent from the previous research and may offer a promising venue for deepening our understanding of the sources of competitive advantage (Gary *et al.,* 2008). Both abstract models and rich case-specific analyses have contributed to our understanding of individual and firm-level behavior. By providing a more rigorous link between these two approaches, our method offers the possibility of a more unified account of strategic behavior.

Limitations and future research

Our findings come with the usual caveats associated with inductive studies, and there are two particularly significant threats to the validity of our findings. First, though Alpha and Beta were similar on many dimensions, we cannot rule out every alternative explanation for the differences we observed. Differences in the underlying technology and the capabilities of the supporting engineering teams may have contributed. Retrospective bias may have also influenced how informants recalled past experiences in the two cases. Second, studying a single firm restricts the generalizability of any finding, particularly in our case as we use an intrafirm comparison to theorize about interfirm differentials. We address some of these problems by using dynamic modeling to organize the structures and causal mechanisms observed in the field into an internally consistent model that captures the behaviors we observed. More importantly, using a formal model to organize our findings allows us to derive precise and testable predictions and thereby enables future scholars to evaluate the utility of our theory in explaining observed differences in competitive performance. The model developed in this study is very simple and does not include

DOI: 10.1002/smj

potentially relevant factors such as changes in development resources, schedule modification, feature cancellation, product architecture, and market response to quality and timeliness. More detailed models should be used to provide operational advice. Another fruitful extension is to capture interaction between capability erosion and market competition. For example, firms with eroding capabilities often lose market share, which may reduce future demand pressure and allow for their recovery. On the other hand, they may also lose critical resources and face further downward pressure (Rahmandad, 2012). Finally, this research is also limited by focusing on a single capability—product development. Identifying key dynamics that can erode different capabilities in different contexts can lead to a productive research program with both theoretical and managerial implications.

ACKNOWLEDGEMENTS

Special thanks to Martine Devos, David Weiss, Randy Hackbarth, John Palframan, and Audris Mockus and several members of Sigma organization for their support of, and contribution to, this research. We are also grateful for the valuable feedback of anonymous reviewers, Michael Cusumano, Bob Gibbons, Rebecca Henderson, Shayne Gary, Jan Rivkin, John Sterman, Jeroen Struben, Lourdes Sosa, Gokhan Dogan, Ezra Zuckerman, Kate Kellogg, and several seminar participants at MIT, Stanford, Wharton, and the Academy of Management.

REFERENCES

Abdel-Hamid T, Madnick SE. 1991. *Software Project Dynamics: An Integrated Approach*. Prentice-Hall: Englewood Cliffs, NJ.

Agarwal R, Helfat CE. 2009. Strategic renewal of organizations. *Organization Science* **20**(2): 281–293.

Argote L. 2012. *Organizational Learning: Creating, Retaining and Transferring Knowledge*. Springer: New York.

Argote L, Beckman SL, Epple D. 1990. The persistence and transfer of learning in industrial settings. *Management Science* **36**(2): 140–154.

Azoulay P, Repenning NP, Zuckerman EW. 2010. Nasty, brutish, and short: embeddedness failure in the pharmaceutical industry. *Administrative Science Quarterly* **55**(3): 472–507.

Barley SR. 1986. Technology as an occasion for structuring: evidence from observations of CT Scanners and Social Order of Radiology Departments. *Administrative Science Quarterly* **31**: 78–108.

Benkard CL. 2000. Learning and forgetting: the dynamics of aircraft production. *American Economic Review* **90**(4): 1034–1054.

Berg P, Kalleberg AL, Appelbaum E. 2003. Balancing work and family: the role of high-commitment environments. *Industrial Relations* **42**(2): 168–188.

Black LJ, Carlile PR, Repenning NP. 2004. A dynamic theory of expertise and occupational boundaries in new technology implementation: building on barley's study of CT scanning. *Administrative Science Quarterly* **49**(4): 572–607.

Bloom N, Eifert B, Mahajan A, McKenzie D, Roberts J. 2013. Does management matter? Evidence from India. *Quarterly Journal of Economics* **128**(1): 1–51.

Bloom N, Van Reenen J. 2007. Measuring and explaining management practices across firms and countries. *Quarterly Journal of Economics* **122**(4): 1351–1408.

Boehm BW. 1981. *Software Engineering Economics*. Prentice–Hall: Englewood Cliffs, NJ.

Brehmer B. 1992. Dynamic decision making: human control of complex systems. *Acta Psychologica* **8**(1): 83–90.

Brown SL, Eisenhardt KM. 1995. Product development – Past research, present findings, and future–directions. *Academy of Management Review* **20**(2): 343–378.

Carroll J, Sterman JD, Markus AR. 1997. Playing the maintenance game: how mental models drive organization decisions. In *Debating Rationality: Nonrational Elements of Organizational Decision Making*, Stern RN, Halpern JJ (eds). ILR Press: Ithaca, NY; 99–121.

Chao RO, Kavadias S, Gaimon C. 2009. Revenue driven resource allocation: funding authority, incentives, and new product development portfolio management. *Management Science* **55**(9): 1556–1569.

Chew WB, Bresnahan T, Clark K. 1990. Measurment, coordination, and learning in a multiplant network. In *Measures for Manufacturing Excellence*, Kaplan R (ed). Harvard Business School Press: Boston, MA; 129–162.

Cronin MA, Gonzalez C, Sterman JD. 2009. Why don't well–educated adults understand accumulation? A challenge to researchers, educators, and citizens. *Organizational Behavior and Human Decision Processes* **108**(1): 116–130.

Cusumano MA, Selby RW. 1995. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*. Free Press: New York.

Darr ED, Argote L, Epple D. 1995. The acquisition, transfer, and depreciation of knowledge in service organizations: productivity in franchises. *Management Science* **41**(11): 1750–1762.

Denrell J, Fang C, Levinthal DA. 2004. From T–Mazes to labyrinths: learning from model–based feedback. *Management Science* **50**(10): 1366–1378.

Denrell J, Fang C, Winter SG. 2003. The economics of strategic opportunity. *Strategic Management Journal* **24**(10): 977–990.

Denrell J, March JG. 2001. Adaptation as information restriction: the hot stove effect. *Organization Science* **12**(5): 523–538.

DOI: 10.1002/smj

Diehl E, Sterman J. 1995. Effects of feedback complexity on dynamic decision making. *Organizational Behavior and Human Decision Processes* **62**(2): 198–215.

Dierickx I, Cool K. 1989. Asset stock accumulation and sustainability of competitive advantage. *Management Science* **35**(12): 1504–1511.

Easton GS, Jarrell SL. 1998. The effects of total quality management on corporate performance: an empirical investigation. *Journal of Business* **71**(2): 253–307.

Eggers JP, Kaplan S. 2009. Cognition and renewal: comparing CEO and organizational effects on incumbent adaptation to technical change. *Organization Science* **20**(2): 461–477.

Eisenhardt KM, Martin JA. 2000. Dynamic capabilities: what are they? *Strategic Management Journal* **21**(10–11): 1105–1121.

Felin T, Foss NJ, Heimeriks KH, Madsen TL. 2012. Microfoundations of routines and capabilities: individuals, processes, and structure. *Journal of Management Studies* **49**(8): 1351–1374.

Ford DN, Sterman JD. 1998. Dynamic modeling of product development processes. *System Dynamics Review* **14**(1): 31–68.

Forrester JW. 1961. *Industrial Dynamics*. The MIT Press: Cambridge, MA.

Foss NJ, Heimeriks KH, Winter SG, Zollo M. 2012. A hegelian dialogue on the micro–foundations of organizational routines and capabilities. *European Management Review* **9**(4): 173–197.

Gary MS. 2005. Implementation strategy and performance outcomes in related diversification. *Strategic Management Journal* **26**(7): 643–664.

Gary MS, Kunc M, Morecroft JDW, Rockart SF. 2008. System dynamics and strategy. *System Dynamics Review* **24**(4): 407–429.

Gary MS, Wood RE, Pillinger T. 2012. Enhancing mental models, analogical transfer, and performance in strategic decision making. *Strategic Management Journal* **33**(11): 1229–1246.

Gavetti G. 2005. Cognition and hierarchy: rethinking the microfoundations of capabilities' development. *Organization Science* **16**(6): 599–617.

Gibbons R, Henderson R. 2010. Relational contracts and the origins of organizational capabilities. In *The Handbook of Organizational Economics*, Gibbons R, Roberts J (eds). Princeton University Press: Princeton, NJ; 680–731

Gittell JH, Seidner R, Wimbush J. 2010. A relational model of how high–performance work systems work. *Organization Science* **21**(2): 490–506.

Glaser BG, Strauss AL. 1967. *The Discovery of Grounded Theory; Strategies for Qualitative Research*. Aldine Publishing Company: Chicago, IL.

Grant RM. 2010. *Contemporary Strategy Analysis*. John Wiley & Sons: Hoboken, NJ.

Hall RI. 1976. A system pathololgy of an organization: the rise and fall of the old saturday evening post. *Administrative Science Quarterly* **21**(2): 185–211.

Harrison JR, Lin Z, Carroll GR, Carley KM. 2007. Simulation modeling in organizational and management research. *Academy of Management Review* **32**(4): 1229–1245.

Helfat CE. 2000. Guest editor's introduction to the special issue: the evolution of firm capabilities. *Strategic Management Journal* **21**(10–11): 955–959.

Helfat CE, Peteraf MA. 2003. The dynamic resource–based view: capability lifecycles. *Strategic Management Journal* **24**(10): 997–1010.

Henderson R, Cockburn I. 1994. Measuring competence – Exploring firm effects in pharmaceutical research. *Strategic Management Journal* **15**: 63–84.

Hodgson GM, Knudsen T. 2004. The firm as an interactor: firms as vehicles for habits and routines. *Journal of Evolutionary Economics* **14**(3): 281–307.

Hoopes DG, Madsen TL. 2008. A capability–based view of competitive heterogeneity. *Industrial and Corporate Change* **17**(3): 393–426.

Ichniowski C, Shaw K. 2003. Beyond incentive pay: insiders' estimates of the value of complementary human resource management practices. *Journal of Economic Perspectives* **17**(1): 155–180.

Ichniowski C, Shaw K, Prennushi G. 1997. The effects of human resource management practices on productivity: a study of steel finishing lines. *American Economic Review* **87**(3): 291–313.

Jones C. 2000. *Software Assessments, Benchmarks, and Best Practices*. Addison–Wesley: Boston, MA.

Klein KJ, Sorra JS. 1996. The challenge of innovation implementation. *Academy of Management Review* **21**(4): 1055–1080.

Krishnan V, Ulrich KT. 2001. Product development decisions: a review of the literature. *Management Science* **47**(1): 1–21.

Leiblein MJ. 2011. What do resource– and capability–based theories propose? *Journal of Management* **37**(4): 909–932.

Lenox MJ, Rockart SF, Lewin AY. 2006. Interdependency, competition, and the distribution of firm and industry profits. *Management Science* **52**(5): 757–772.

Lenox MJ, Rockart SF, Lewin AY. 2010. Does interdependency affect firm and industry profitability? An empirical test. *Strategic Management Journal* **31**(2): 121–139.

Leveson N, Dulac N, Marais K, Carroll J. 2009. Moving beyond normal accidents and high reliability organizations: a systems approach to safety in complex systems. *Organization Studies* **30**(2–3): 227–249.

Levinthal DA. 1997. Adaptation on rugged landscapes. *Management Science* **43**(7): 934–950.

Levinthal D. 2000. Organizational capabilities in complex worlds. In *The Nature and Dynamics of Organizational Capabilities*, Dosi G, Nelson R, Winter S (eds). Oxford University Press: New York; 363–379.

Levinthal D, Rerup C. 2006. Crossing an apparent chasm: bridging mindful and less–mindful perspectives on organizational learning. *Organization Science* **17**(4): 502–513.

Levitt B, March JG. 1988. Organizational learning. *Annual Review of Sociology* **14**: 319–340.

Lyneis JM, Ford DN. 2007. System dynamics applied to project management: a survey, assessment, and directions for future research. *System Dynamics Review* **23**(2–3): 157–189.

MacCormack A, Kemerer CF, Cusumano M, Crandall B. 2003. Trade–offs between productivity and quality in selecting software development practices. *IEEE Software* **20**(5): 78–85.

Mandler G. 1984. *Mind and Body: Psychology of Emotion and Stress*. W.W. Norton: New York.

March JG. 1991. Exploration and exploitation in organizational learning. *Organization Science* **2**(1): 71–87.

March JG. 2001. Foreword. In *Dynamics of Organizations: Computational Modeling and Organization Theories*, Lomi A, Larsen ER (eds). The MIT Press: Cambridge, MA; ix–xvii.

Mauri AJ, Michaels MP. 1998. Firm and industry effects within strategic management: an empirical examination. *Strategic Management Journal* **19**(3): 211–219.

Moløkken K, Jørgensen M. 2003. A review of surveys on software effort estimation. In *Proceedings of International Symposium on Empirical Software Engineering*, ACM-IEEE, Rome, Italy; 223–230.

Morecroft JDW. 1997. The rise and fall of people express: a dynamic resource–based view. *Proceedings of the 15th International System Dynamics Conference: "Systems Approach to Learning and Education into the 21st Century"* (Volume **2**) Y. Barlas, V. G. Diker and S. Polat. Istanbul, Turkey, Bogazici University Printing Office. 579–586.

Nadkarni S, Barr PS. 2008. Environmental context, managerial cognition, and strategic action: an integrated view. *Strategic Management Journal* **29**(13): 1395–1427.

Nelson R, Winter SG. 1973. Toward an evolutionary theory of economic capabilities. *American Economic Review* **63**(2): 440–449.

Nelson RR, Winter SG. 1982. *An Evolutionary Theory of Economic Change*. Belknap Press of Harvard University Press: Cambridge, MA.

Oliva R, Sterman JD. 2001. Cutting corners and working overtime: quality erosion in the service industry. *Management Science* **47**(7): 894–914.

Parmigiani A, Howard-Grenville J. 2011. Routines revisited: exploring the capabilities and practice perspectives. *Academy of Management Annals* **5**: 413–453.

Perlow LA, Okhuysen GA, Repenning NP. 2002. The speed trap: exploring the relationship between decision making and temporal context. *Academy of Management Journal* **45**(5): 931–955.

Perlow LA, Repenning NP. 2009. The dynamics of silencing conflict. *Research in Organizational Behavior* **29**: 195–223.

Perrow C. 1999. *Normal Accidents: Living with High–risk Technologies*. Princeton University Press: Princeton, NJ.

Pfeffer J, Sutton RI, NetLibrary Inc.. 2000. *The Knowing–doing Gap: How Smart Companies Turn Knowledge into Action*, xv. Harvard Business School Press: Boston, MA.

Porac JF, Thomas H, Badenfuller C. 1989. Competitive groups as cognitive communities – the case of Scottish Knitwear Manufacturers. *Journal of Management Studies* **26**(4): 397–416.

Priem RL, Butler JE. 2001. Is the resource–based "view" a useful perspective for strategic management research? *Academy of Management Review* **26**(1): 22–40.

Rahmandad H. 2005. Three essays on modeling dynamic organizational processes. PhD., Sloan School of Management, Massachusetts Institute of Technology: Cambridge, MA.

Rahmandad H. 2008. Effect of delays on complexity of organizational learning. *Management Science* **54**(7): 1297–1312.

Rahmandad H. 2012. Impact of growth opportunities and competition on firm–Level capability development trade–offs. *Organization Science* **23**(1): 138–154.

Rahmandad H, Repenning N, Sterman J. 2009. Effects of feedback delay on learning. *System Dynamics Review* **25**(4): 309–338.

Rahmandad H, Sterman JD. 2012. Reporting guidelines for simulation-based research in social sciences. *System Dynamics Review* **28**(4): 396–411.

Rahmandad H, Weiss D. 2009. Dynamics of concurrent software development. *System Dynamics Review* **25**(3): 224–249.

Repenning NP. 2000. A dynamic model of resource allocation in multi–project research and development systems. *System Dynamics Review* **16**(3): 173–212.

Repenning NP. 2001. Understanding fire fighting in new product development. *Journal of Product Innovation Management* **18**: 285–300.

Repenning NP. 2002. A simulation–based approach to understanding the dynamics of innovation implementation. *Organization Science* **13**(2): 109–127.

Repenning NP, Sterman JD. 2002. Capability traps and self–confirming attribution errors in the dynamics of process improvement. *Administrative Science Quarterly* **47**(2): 265–295.

Rivkin JW. 2001. Reproducing knowledge: replication without imitation at moderate complexity. *Organization Science* **12**(3): 274–293.

Rivkin JW, Siggelkow N. 2003. Balancing search and stability: interdependencies among elements of organizational design. *Management Science* **49**(3): 290–311.

Roberts KH. 1990. Some characteristics of one type of high reliability organization. *Organization Science* **1**(2): 160–176.

Roberts PW, Dowling GR. 2002. Corporate reputation and sustained superior financial performance. *Strategic Management Journal* **23**(12): 1077–1093.

Rubinstein SA, Kochan TA. 2001. *Learning from Saturn: Possibilities for Corporate Governance and Employee Relations*. ILR Press: Ithaca, NY.

Rudolph JW, Repenning NP. 2002. Disaster dynamics: understanding the role of quantity in organizational collapse. *Administrative Science Quarterly* **47**: 1–30.

Rumelt RP. 1991. How much does industry matter. *Strategic Management Journal* **12**(3): 167–185.

Sagan SD. 1993. *The Limits of Safety: Organizations, Accidents, and Nuclear Weapons*. Princeton University Press: Princeton, NJ.

Salter WEG. 1960. *Productivity and Technical Change*. Cambridge University Press: Cambridge, UK.

Schreyoegg G, Kliesch-Eberl M. 2007. How dynamic can organizational capabilities be? Towards a dual–process

DOI: 10.1002/smj

model of capability dynamization. *Strategic Management Journal* **28**(9): 913–933.
Spanos YE, Lioukas S. 2001. An examination into the causal logic of rent generation: contrasting Porter's competitive strategy framework and the resource–based perspective. *Strategic Management Journal* **22**(10): 907–934.
Spence M. 1976. Competition in salaries, credentials, and signaling prerequisites for jobs. *Quarterly Journal of Economics* **90**(1): 51–74.
Sterman JD. 1989. Modeling managerial behavior: misperceptions of feedback in a dynamic decision making experiment. *Management Science* **35**(3): 321–339.
Sterman JD. 1994. Learning in and about complex systems. *System Dynamics Review* **10**(2–3): 91–330.
Sterman J. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. McGraw–Hill/Irwin: Boston, MA.
Sterman JK, Repenning NP, Kofman F. 1997. Unanticipated side effects of successful quality programs: exploring a paradox of organizational improvement. *Management Science* **43**: 503–521.
Sutton RI, Staw BM. 1995. What theory is not. *Administrative Science Quarterly* **40**(3): 371–384.
Syverson C. 2011. What determines productivity? *Journal of Economic Literature* **49**(2): 326–365.
Taylor T, Ford DN. 2006. Tipping point failure and robustness in single development projects. *System Dynamics Review* **22**(1): 51–71.
Teece DJ. 2007. Explicating dynamic capabilities: the nature and microfoundations of (sustainable) enterprise performance. *Strategic Management Journal* **28**(13): 1319–1350.
Teece DJ, Pisano G, Shuen A. 1997. Dynamic capabilities and strategic management. *Strategic Management Journal* **18**(7): 509–533.
Thompson P. 2007. How much did the liberty shipbuilders forget? *Management Science* **53**(6): 908–918.
Tripsas M. 2009. Technology, identity, and inertia through the lens of "The Digital Photography Company". *Organization Science* **20**(2): 441–460.
Weick KE, Sutcliffe KM, Obstfeld D. 1999. Organizing for high reliability: processes of collective mindfulness. *Research in Organizational Behavior* **21**: 81–123.
Welch J, Welch S. 2005. *Winning*. HarperBusiness Publishers: New York.
Williamson OE. 1999. Strategy research: governance and competence perspectives. *Strategic Management Journal* **20**(12): 1087–1108.
Winter SG. 2003. Understanding dynamic capabilities. *Strategic Management Journal* **24**(10): 991–995.
Yin RK. 2009. *Case Study Research: Design and Methods*. Sage Publications: Los Angeles, CA.
Zollo M, Winter SG. 2002. Deliberate learning and the evolution of dynamic capabilities. *Organization Science* **13**(3): 339–351.
Zuashkiani A, Rahmandad H, Jardine A. 2011. Mapping the dynamics of overall equipment effectiveness to enhance asset management. *Journal of Quality in Maintenance Engineering* **17**(1): 74–92.

SUPPORTING INFORMATION

Additional supporting information may be found in the online version of this article:

File S1. Capability erosion dynamics.

DOI: 10.1002/smj