Developing Products on "Internet Time": The Anatomy of a Flexible Development Process

Alan MacCormack • Roberto Verganti • Marco Iansiti
Harvard Business School, Boston, Massachusetts 02163
Politecnico di Milano, Milan, Italy
Harvard Business School, Boston, Massachusetts 02163
amaccormack@hbs.edu • roberto.verganti@polimi.it • miansiti@hbs.edu

Uncertain and dynamic environments present fundamental challenges to managers of the new product development process. Between successive product generations, significant evolutions can occur in both the customer needs a product must address and the technologies it employs to satisfy these needs. Even within a single development project, firms must respond to new information, or risk developing a product that is obsolete the day it is launched. This paper examines the characteristics of an effective development process in one such environment—the Internet software industry. Using data on 29 completed development projects we show that in this industry, constructs that support a more *flexible* development process are associated with better-performing projects. This flexible process is characterized by the ability to generate and respond to new information for a longer proportion of a development cycle. The constructs that support such a process are greater investments in architectural design, earlier feedback on product performance from the market, and the use of a development team with greater amounts of "generational" experience. Our results suggest that investments in architectural design play a dual role in a flexible process: First, through the need to select an architecture that maximizes product performance and, second, through the need to select an architecture that facilitates development process flexibility. We provide examples from our fieldwork to support this view.
(*New Product Development; Software Development; Innovation; Flexibility; Internet*)

1. Introduction

Over the last two decades, a rich stream of studies has broadened our understanding of how to effectively manage new product development projects (e.g., Katz and Allen 1982, Clark and Fujimoto 1991, Cusumano and Nobeoka 1992, Cooper 1995, Iansiti 1997). These have demonstrated that a wide range of decisions regarding organization structure, team composition, and process are associated with development performance in terms of lead time, cost, and quality. Many of these studies have been undertaken in environments where the target market and the technologies to be employed in a product are relatively well understood. As a result, the models derived from them portray development as a sequential process of design followed by execution. Effective projects are characterized by a structure that minimizes changes to the product design once the execution stage has begun (Cooper 1990).

Uncertain and dynamic environments,[1] however, present fundamental challenges to these accepted

[1] By uncertain, we mean environments in which future evolutions in markets and/or technologies are hard to predict. By dynamic, we mean environments in which these evolutions occur rapidly.

0025-1909/01/4701/0133$5.00
1526-5501 electronic ISSN

models of the new product development process. In such environments, significant changes can occur in both the customer needs a product must address and the technologies it employs to satisfy these needs between successive product generations. Therefore, in each project there is a requirement to understand how these changes should be incorporated into the design. An even greater challenge comes from the need to respond to new or changing information *during* a development project. Without this ability, managers risk developing a product that is targeted at a market where customer requirements have long since evolved.

Previous authors have developed theoretical models which demonstrate the value of greater flexibility in a design process faced with uncertainty (Krishnan et al. 1997, Bhattacharya et al. 1998). Most of these models, however, rely on the trade off of benefits from delaying commitment to certain parts of a design against the costs that such delays consequently impose on a project. As a result, they employ a static optimization perspective, with an assumption that the costs of various actions are out of the immediate control of the development team. This study, by contrast, attempts to examine the underlying mechanisms through which firms *directly* influence the flexibility of their development processes (and hence the costs assumed to be fixed in many theoretical models). Building on recent exploratory work in "high-velocity" environments (Eisenhardt and Tabrizi 1995, Iansiti and MacCormack 1997), we identify several constructs that support a more flexible process and show that these constructs are good predictors of project performance in an industry where development teams face extreme levels of uncertainty.

The remainder of this paper is divided into four sections. In the next section, we describe the prevailing model of product development advocated by much academic research and discuss the challenges this model faces in uncertain and dynamic environments. This motivates our study of several constructs that support a more flexible process. We then introduce the context for our study—the Internet software industry—and describe the methods used to gather data on our sample of projects. We next report our empirical results, discussing the statistical evidence linking process choices to a measure of project performance. We conclude by discussing how these results inform our understanding of how a more flexible process is achieved in practice.

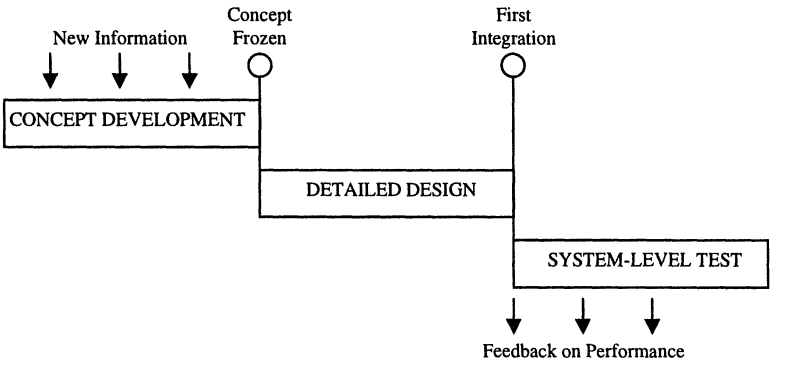
2. Models of the Product Development Process

Many models of product development are based on the premise that design activities are best divided into a number of sequential project "stages" separated by milestones called "gates" (e.g., Cooper 1990, Ulrich and Eppinger 1995). During each stage, alternative design decisions are explored and certain options chosen over others that are found to be inferior. On arrival at each gate, managers evaluate progress, decide whether the project is to proceed, and if so agree the foundation on which work in subsequent stages will build. As a project moves through successive stages and gates, the design evolves in increasing levels of detail, from high-level representations of the overall concept to the details of each and every component.

While the stage-gate process has been shown to be effective in stable environments (Cooper and Kleinschmidt 1996), its value has been questioned in uncertain and dynamic environments (Bhattacharya et al. 1998, Iansiti and MacCormack 1997). The challenges that these environments pose can be illustrated by considering a simple model of development consisting of three separate stages: concept development (where the overall concept is defined and the product architecture developed), detailed design (where individual modules are designed and tested), and system-level testing (where these modules are integrated into a complete system and tested). In a stage-gate process, these stages are performed sequentially: The product concept is defined and frozen *prior* to the start of detailed design, and the functionality this specifies in each module is completed *prior* to the start of system-level test (see Figure 1).

The first challenge this model faces is that it assumes all information about potential design choices is known or can be discovered during concept development. After this stage has ended, there is

Figure 1 A Stage-Gate Model of Product Development



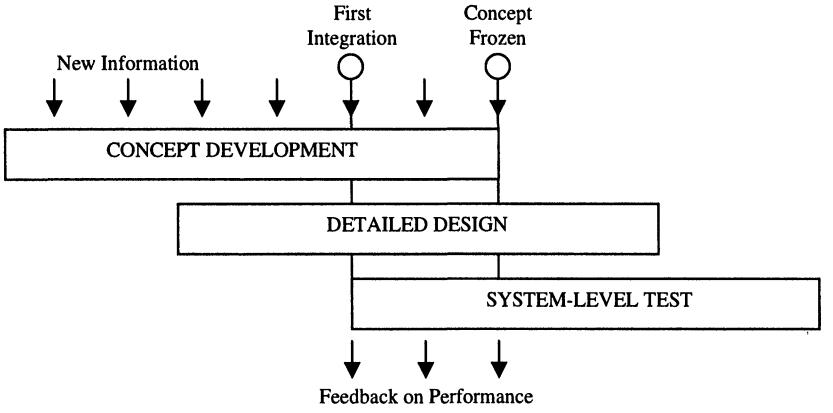
little flexibility to change the overall design. In uncertain and dynamic environments, however, firms cannot predict every potential design choice up-front. Instead, new information about market needs and technical opportunities is expected to emerge as a project proceeds. There is, therefore, a greater need to keep the product concept open to change, maintaining design flexibility for a longer proportion of development. To achieve this, Stages 1 and 2 must overlap, implying that detailed design must start *before* the specification is complete.

The second challenge this model faces is that feedback on how the product performs as a system is not obtained until late in a project, when the functionality in each module has been fully developed. In uncertain environments, however, such an approach involves significant risk. Such uncertainties are present, for example, when a "breakthrough" product concept is proposed (Wheelwright and Clark 1992), given that in such a project the ability to satisfy customer needs remains largely unknown until customers interact with the product in the end-use context. The fact that such breakthroughs often require a new product architecture heightens this uncertainty, given that the use of unfamiliar components and interfaces increases the probability of problematic interactions. Therefore, in such environments there is a need to gain feedback from "early" system-level tests, implying that Stages 2 and 3 must overlap (hence, the first test of system-level performance occurs *before* the modules from which the system is built are complete). Indeed, in the most flexible of processes, Stages 1 and 3 will also overlap, thereby allowing feedback from system-level tests to have a direct impact on the evolution of the product concept (see Figure 2).

Recent studies have begun to explore more flexible models of development characterized by the overlapping of development stages (e.g. Krishnan et al. 1997). These models are founded upon a process that emphasizes the ability to generate and respond to new information for as long as possible during development. Rather than a sequential stage-gate process, development becomes an "evolutionary" process of learning and adaptation (Tushman and O'Reilly 1997). Activities proceed in an iterative manner, the feedback gained in one cycle of experimentation being used to guide activities in the next (Eisenhardt and Tabrizi 1995).

The environment where flexible models of development have been most widely advocated has been in software development (e.g., "Incremental," Wong 1984; "Spiral," Boehm 1988; "Rapid Prototyping," Connell and Shafer 1989). These models have been proposed to address flaws in the traditional "waterfall" model of development, a stage-gate-type model that emerged from efforts to control the management of large software projects (Royce 1970). They typically employ an iterative process, founded upon the construction of a series of prototypes used to gather feedback on whether the design meets customer requirements. Indeed, variations on these approaches have been observed in a number of commercial software firms (Cusumano and Selby 1995, Iansiti and MacCormack 1997). However, despite the literature describing more flexible models for developing software, there is as yet little empirical validation that

Figure 2 A More Flexible Model of Product Development



they result in better-performing projects. One aim of this study is to address this shortfall.

Constructs Supporting a More Flexible Process

We consider three constructs that support a more flexible process: greater investments in architectural design, earlier feedback on a product's system-level performance, and a development team with greater amounts of "generational" experience.

The requirement to overlap development stages in a flexible process creates three major challenges for the design of the product architecture. The first of these is the need to start detailed design work *prior* to the product architecture being fully defined. Achieving this objective requires those parts of a design that are uncertain or likely to change to be buffered from those parts expected to remain stable. The second challenge is the need to integrate the system *prior* to the completion of all of its component modules. This requires partitioning design work in such a way that critical elements of system performance can be demonstrated even when some of the individual modules are only partially complete. The final challenge is the need to respond to new information arising during the later stages of a project. This requires a product architecture that can accept the addition of new functionality as a project progresses without requiring major changes to other parts of the system.[2]

In stable environments where customer requirements and component technologies are known with confidence at the beginning of a project, the challenges described above are not present. The primary focus for architectural design efforts is therefore aimed quite naturally at maximizing product performance. In uncertain and dynamic environments, however, these challenges suggest significant benefits will be gained from an architecture that can also facilitate greater flexibility (Parnas 1972, Ulrich 1995, Sanchez and Mahoney 1996). The need for flexibility therefore creates an additional design criterion that the product architecture must satisfy. These two objectives (i.e., maximizing product performance and facilitating process flexibility) are often incompatible from a design standpoint; hence, the selection of the "optimal" product architecture becomes a more complex problem (Ulrich 1995). As a consequence, we assume that firms adopting a more flexible process will allocate greater resources to the design of the product architecture to resolve these tensions.

HYPOTHESIS 1. *In uncertain and dynamic environments, greater investments in architectural design will be associated with better performing projects.*

While greater investments in architectural design help facilitate a more flexible process, they do not guarantee that a project takes maximum advantage of such flexibility. To do this, the development process must encompass mechanisms for both generating and responding to new information as it proceeds. We discuss each of these objectives below.

The need to generate new information requires a search process focused on the major sources of uncertainty facing a project, whether these are related to

[2] The software engineering field describes such architectures as "highly modular" and/or "loosely coupled".

the underlying technologies in a product or the market context in which this product operates. The value of search in processes of organizational renewal has been documented in previous research (Levinthal and March 1981, Nelson and Winter 1982, March 1991). In a stage-gate model of development, however, search typically occurs only during the early project stages. Once detailed design begins, it is no longer an active part of the process. A more flexible process, by contrast, requires that search occur throughout development. The objective is to generate early feedback on how the product performs as a "system," an essential activity given that in uncertain and dynamic environments, changes to the product architecture are more likely to be required.[3]

Gaining early feedback on system-level performance requires the adoption of an architecture in which the system's main components can be integrated at an early stage (as discussed above). However, it also requires that a project establish intensive links with the target market through which feedback on the system's external interactions with its operating environment can be obtained, and hence its ability to meet customer needs. Recent research demonstrates the value of such links, especially to the lead users of new products and services (Von Hippel 1986, 1988). Where information about the user environment is tacit or "sticky" (i.e., not easily captured by traditional market research techniques) there is value in mechanisms which facilitate the release of early product versions to users when a project retains the ability to change the design. These early versions provide a conduit for generating information on system-level performance that can be used to improve the design as the project proceeds.

HYPOTHESIS 2. *In uncertain and dynamic environments, earlier feedback on a product's system-level performance will be associated with better performing projects.*

[3] The need for a new product architecture can stem from changes in the relative performance of different component technologies which render older architectures obsolete (e.g., Henderson and Clark 1990) or from changes in customer requirements which dictate a new set of performance trade-offs (e.g., Christensen and Rosenbloom 1995).

The need to respond to the new information generated through search requires that a team makes changes to the existing design quickly and at low cost (Thomke 1997). This objective is facilitated through the selection of a product architecture which minimizes the impact of design changes on the rest of the system (as discussed above). However, it also requires the ability to rapidly refine the new information generated through search and integrate the results into the evolving design. The academic literature suggests several mechanisms through which this can be achieved. For example, recent studies have demonstrated that a larger capacity for experimentation can help a team gain rapid feedback on the performance of alternative solutions, and, subsequently, to integrate this information into the design (Iansiti 1997, Thomke 1998). However, there is still a need to identify the appropriate set of experiments to conduct and, subsequently, to interpret the results as they emerge. Herein lies the role of experience.

While the value of experience is clear in environments that are stable, its value is often questioned in uncertain and dynamic environments, given that in such settings, knowledge of specific technologies and applications can rapidly become obsolete. Several authors suggest that experience can be a *disadvantage* in such environments, given that inertia and rigidity in problem solving may lead to the selection of inappropriate solutions for novel problems (e.g., Katz and Allen 1982). Still, this appears to be an overly broad conclusion, and one that leads to unappealing managerial prescriptions. The question must be asked, therefore, whether there are specific types of experience that have value in uncertain and dynamic environments.

The main challenge that highly dynamic and uncertain environments generate for development teams is the need to learn about new technical solutions and their potential applications. Experience therefore, has value to the degree that it facilitates these activities, allowing a team to frame and direct an effective experimentation strategy to resolve the uncertainties that arise during a project (Thomke et al. 1998, Verganti 1999). Experience also has value to the degree that it allows teams to integrate the new information generated from these experiments into a

coherent system-level design (Iansiti 1997). We assert that these useful forms of experience come from having previously worked on multiple different project generations.[4] Completing multiple different projects over different generations is likely to involve dealing with different/changing contexts and, hence, results in a more abstract approach to development (i.e., one which evolves through the lessons learned in applying it to each new project). It also requires that several cycles of learning have been completed at the *system-level*, ensuring feedback on the impact of design choices has been received both locally (i.e., through developing individual modules) and globally (i.e., through seeing how these modules perform as a system). We call this type of experience "generational" experience, differentiating it from the "context-specific" form of experience whose value is subject to obsolescence in uncertain and dynamic environments. Such a distinction is consistent with work in organization theory, which shows that learning occurs at multiple levels of abstraction (Fiol and Lyles 1985, Argyris 1977, McKee 1992). Lower-level learning results in knowledge that can be directly applied to a specific context, whereas higher-level learning results in a deeper knowledge of the process of problem solving, especially with respect to analyzing new frames of reference.

HYPOTHESIS 3. *In uncertain and dynamic environments, development teams with greater amounts of generational experience will be associated with better-performing projects.*

The arguments above describe how in uncertain and dynamic environments, we expect constructs which underlie a more flexible process to be associated with better-performing projects. We need to be specific, however, about the dimensions of project performance that are expected to vary, as these dictate the focus for our empirical study. The aim of a flexible process is to provide a closer match between the design of a product, and the evolving requirements that this product must address in the target market. We therefore expect such a process will result in a better-performing product *as perceived by a customer* (i.e., when compared to the equivalent product developed using a less flexible process). Given that the adoption of a more flexible process may require the use of additional resources, however, it is also important to capture data on the resource productivity of projects. This will allow us to assess the magnitude of any potential trade-offs that might exist when using such a process.

3. The Empirical Setting

The analysis in this paper is based on data captured during a two-year study of product development practices in the Internet software industry (see MacCormack 1998 for a detailed discussion). During this period, this industry presented tremendous levels of uncertainty to developers of new products. The industry was "created" in 1993, when the development of a graphical interface to the Internet (at that time, a communication network used mainly by scientists, academics, and the military) established the "World Wide Web." Since then, its growth has been dramatic—from less than 200 web servers in June 1993, there were over 200,000 as of January 1997 (Reid 1997). This growth has spurred the creation of hundreds of new firms and applications and a variety of alternative technical standards.

The unit of the analysis used in this study is the project. Our research was conducted in two stages. In the first stage, we conducted interviews with project managers at several companies, both to understand the specific nature of the development process in this industry and to identify how we could operationalize measures of the constructs supporting a more flexible process. In the second stage, we collected data through the use of a survey instrument distributed to project managers at a sample of firms identified through a review of industry journals. In most cases, we visited participating firms to collect additional descriptive data on projects. Our final sample consists of 29 completed projects from 17 firms.[5] These

[4] We use the word "generations" to discriminate between major new platform releases and minor derivative/incremental releases, which typically do not involve reexamining the architecture of the product.

[5] We approached 39 firms to participate in the survey phase. We followed up with phone calls and e-mails to solicit responses,

projects cover a broad range of applications, including products, services, and development tools targeted at both commercial and consumer users. Given this heterogeneity, we rely on an innovative method for assessing comparative levels of product performance through the use of an expert panel.

The Measure of Performance: Product Quality

To evaluate the relative performance of products in our sample, we rely upon the assessments of a panel of experts gathered using a two-round Delphi process (Dalkey and Helmer 1963). Similar methods have been used in other studies where the performance of products is not directly comparable (Clark and Fujimoto 1991). The panel of experts consisted of 14 industry observers from a variety of journals and websites that review Internet software products. Experts were asked to rate the overall quality of products *relative to other competitive products which targeted similar customer needs at the time the product was launched*. Overall quality was defined as a combination of product features (i.e., the range of features the product contained), technical performance (i.e., how fast/easy-to-use the product was), and reliability (i.e., the extent to which the product operated without failure).[6] Assessments were given on a Likert scale ranging from 1–7, 4 indicating that the product was at parity with competitive offerings. Experts were asked to indicate their level of knowledge of the product (also on a 1–7 scale) and list up to three competitive products against which their comparisons were made.

Variation among experts' assessments of the same product can be attributed to several factors: a lack of knowledge of the product, a different understanding of the market segment the product competes in, ambiguity in the scale for assessment, and actual variation in experts' subjective judgments. The second round of the Delphi process was aimed at eliminating variability because of the first three of these. For each product, we provided panelists with the top three competitive products against which knowledgeable experts (i.e., those scoring 4 or higher) had made their assessments. We also provided the mean scores for each individual product, and the mean scores overall. Panelists were asked to review these data and submit a new evaluation for each product. As compared to the first round, the second round of the Delphi generated a convergence in experts' opinions on individual products while creating additional variance in the spread of mean performance outcomes across products.[7] In the following analysis, we use the mean score provided by knowledgeable experts as our dependent variable.[8]

Control Variable: Resources. We capture a measure of the development and test resources allocated to each project to control for the potential impact of resources on product quality.[9] This measure is adjusted to reflect the fact that the complexity of products in the sample varies. We adjust for this differing complexity by modeling the resources allocated to each project as a function of both the lines of new code developed in each project and a dummy which indicates whether the project was to develop a service (see Appendix A).[10] The residuals from this model are used as our control.

eventually collecting survey data on 29 projects from 17 firms (a response rate of 43%). We collected additional descriptive data for 22 of the 29 individual projects through face-to-face interviews conducted with the project manager. The sample contains many of the "defining products" of the Internet age, including Netscape's Navigator 3.0 and 4.0 browsers, Microsoft's Explorer 3.0 and 4.0 browsers, Yahoo!'s My Yahoo! service, etc.

[6] Our aim was to understand how a user would rate the performance of the product, not how a software engineer would. We therefore excluded some aspects of software performance (e.g., maintainability) cited in the literature.

[7] A third round with two pilot products showed that experts where not willing to change their assessments further and that iteration of the process would therefore not generate additional convergence (Linstone and Turoff 1975).

[8] The highest variance in experts' judgements of product quality came from experts who rated their knowledge of the product as equal to 3 or lower. We therefore decided to focus only on experts scoring 4 or higher.

[9] All else equal, one might expect a trade-off to exist between resource productivity and product quality.

[10] Projects to develop new services (e.g., websites) were observed to achieve higher productivity than other projects in terms of lines of code per person-day because of their use of different programming languages and commands.

Measures of the Development Process

We capture four measures of the development process: investments in architectural design, early market feedback, early technical feedback, and generational experience.

Investments in Architectural Design. In Internet software development projects, the product architecture must, on the one hand, optimize the operating performance of the product (in terms of memory usage, speed, etc.), yet on the other, it must facilitate making changes to the product design as new information emerges as a project proceeds. We assume that the resources allocated to architectural design activities, relative to those allocated to development and test activities, reflect the emphasis given to resolving these potentially conflicting objectives. To capture the relative levels of investment in architectural design, we asked project managers to break down the resources devoted to each project in four categories: project management, architecture design, development, and test. We then calculated the ratio of architecture design resources to development and test resources. Given that the size of products in the sample varies significantly, we adjusted this ratio to control for scale effects.[11] This was done by modeling the ratio as a function of the lines of new code developed in each project (see Appendix B). In our analysis, we use the residuals from this model as our measure of investments in architectural design, controlled for scale.

Early Market and Technical Feedback. Three milestones stand out as particularly important in Internet software development projects, given that each signifies a point at which new information becomes available to the project team. These milestones are: the point at which the first prototype is presented to customers, the point at which the various component modules are first integrated into a partially working system, and the point at which the first beta release is made available to customers.[12]

The first prototype is typically shown to customers during the concept development stage. It represents the first time that the project team receives feedback from customers on the overall product concept. Given the uncertainty surrounding the requirements that users have of new applications in this industry, this is a critical activity. Early feedback from such prototypes allows a team to make modifications to the design of the product architecture during the period it has the most flexibility to do so—when detailed design work has not yet progressed significantly.

The next major milestone in a project is system integration, when the various component modules in the design are first integrated into a partially working version of the system. This milestone typically occurs part way through the detailed design stage, representing the first point at which the project team receives feedback on how modules in the design interact and, consequently, how well they perform as a system. Given the uncertainty surrounding the nature of the product architecture in this environment, early feedback on system-level performance is an essential part of development. Problematic interactions, which are hard to predict ex ante, often emerge for the first time, requiring that the team revisit design decisions made earlier in the project. Given that this first version of the system is often used as the foundation on which subsequent functionality is built, the importance of achieving this milestone early is magnified.[13]

The final milestone marking the generation of new information about the design is the first beta release, which occurs sometime after the first system integration. This point represents the first time that the project team receives feedback from customers on the functioning of the product in the end-use application context (even though the product is typically not functionally complete). Given the uncertainty

[11] All else equal, one might expect larger projects to allocate *proportionately* fewer resources to architecture design.

[12] We define a prototype as a version of the product which cannot function as a system (e.g., a mock-up of the user interface). We define a beta version as a version of the product that contains at least part of all the core component modules (even though these modules may be functionally incomplete), are hence can function as a system. With this definition, the first beta version can never be released before the first system integration has occurred.

[13] Changes and additions to this first version are made through regular system "builds," during which new code is integrated into the design and regression tests run to ensure these new additions work well with the main code base.

surrounding potential interactions between the product and its operating environment, early feedback from beta versions is an important activity in this industry.[14] This feedback allows a team to "synchronize" the emerging design with the evolving needs of the market, generating new information on both the performance of existing features and on the potential for new features which could significantly enhance the product offering.

The milestones described above all refer to points at which new information on product performance becomes available to a project team. We expect that projects in which they occur earlier (thereby having greater influence on continuing development activities) will outperform others. The way in which we operationalize this concept is to capture the percentage of the product's functionality that has been developed when each milestone is reached.[15,16] We assert that projects reaching these milestones with *less* of the product's functionality having been completed will outperform others.[17] Given that this measure is likely to be sensitive to the size of a project, we control for scale effects by modeling the percentage of product functionality developed at each milestone as a function of the lines of new code developed in each project (see Appendix B). In our analysis, we use the residuals from these models as our measures for how early each milestone occurs, controlled for scale.

The measures of product functionality developed at each milestone turn out to be correlated (see Appendix D).[18] For modeling purposes, one would normally proceed by developing a composite variable to capture the common variance across all three. In this study, however, we wanted to preserve the conceptual distinction between the milestones which signify new information arising from customers (first prototype and first beta) and the system integration milestone, where the new information generated is primarily of a technical nature, relating to interactions between different modules. We therefore developed a composite variable based on the measures of product functionality developed at both first prototype and first beta (referred to as early market feedback), and retained the measure of product functionality developed at system integration (referred to as early technical feedback) for use in our analysis.[19]

Generational Experience. A flexible process requires that as development proceeds, changes to the evolving design can be made quickly and at low cost. In our theoretical motivation, we argued that one of the mechanisms through which this objective can be achieved was through a development team which has greater amounts of generational experience. We captured the amount of generational experience within each development team by asking project managers to assess the proportion of team members whose past experience with software development projects fell into the following four categories: no previous experience, one generation of experience, two generations of experience, and greater than two generations of experience. In our analysis, we use the final category, the proportion of team members with greater than two generations of experience, as our indicator

[14] Projects may release multiple beta versions prior to launch. The most significant milestone however, is the first beta release, which marks the start of the period when customers can actively contribute to development activities.

[15] Using the percentage of a project's overall lead time which has elapsed at each milestone would be misleading, given that projects that experience delays in later stages would appear to reach these milestones earlier.

[16] The functionality contained in the product at launch is defined as 100%. We asked project managers to assess the percentage of functionality that existed in the product at each of the three major milestones, assuming that the functionality when the project started was 0%.

[17] One may ask, why not reach a milestone with no functionality? Given the way we define two of these milestones however, this is not possible. By definition, the first system integration and the first beta release require a working version of the product to be available, even if this version is incomplete.

[18] This is to be expected, given the way we define these milestones. A beta version cannot be released until the system has been integrated; hence, an early beta release requires an early system integration. Note, however, the reverse is not true. Early system integration does not dictate that a beta version is released soon thereafter.

[19] The composite measure for early market feedback was constructed by taking the first component from a principal component analysis of the two sets of residuals obtained when predicting product functionality at first prototype and first beta release with the logarithm of new code.

Table 1 Descriptive Statistics

Variable	Description	Mean	S. Dev.	Min	Max
Quality	Mean experts assessment of quality relative to competitive products[a]	4.46	0.49	3.50	5.52
Code	Size of application (lines of code)	376K	459K	5K	1.5Mn
New Code	Amount of new code developed in project (lines of code)	155K	205K	2K	750K
Resources	Total development and test resources consumed in project (person-days)	5369	10771	274	43200
Architecture Design Effort	Ratio of architecture design resources to development and test resources	0.26	0.20	0	0.67
Market Feedback: 1. First Prototype	Percentage of product functionality included in the first prototype	41%	24%	10%	90%
Tech. Feedback: First Integration	Percentage of product functionality included in the first system integration	61%	22%	20%	100%
Market Feedback: 2. First Beta	Percentage of product functionality included in the first beta release	77%	19%	34%	100%
Generational Experience	Proportion of team with greater than two generations of project experience	52%	35%	0%	100%

[a] = Only knowledgeable experts (i.e., experts with confidence on their assessment ≥ 4) are considered

of generational experience.[20] Note the emphasis here on the number of project generations, rather than on years of experience. We therefore discriminate between a developer that has worked on a single technology/project for many years and a developer that has worked on several complete projects, even if each lasted only a few months. The latter, in our view, has a better experience base for developing products in an uncertain and dynamic environment.

Descriptive Statistics

Table 1 summarizes the data on our sample of projects. To make the table more meaningful, measures of the development process are reported here with their original values, rather than their values after adjusting for scale/complexity effects (descriptive statistics and a correlation table for the adjusted variables are given in the appendices). Note in particular the wide variation in the percentage of product functionality completed at each milestone. For example, one project did not show a prototype to customers until almost all of the product's functionality had been developed. In contrast, another distributed a working version of the system to customers as a beta release when only 34% of the functionality was complete. We expect the latter project would have had a significant advantage in integrating new information on evolving market requirements into the product design as development progressed.

4. Empirical Analysis

Our modeling approach involved the use of OLS regression to predict the mean quality of products in the sample (as rated by knowledgeable experts), using a control for resources and measures of the development process.[21] We tested the robustness of results using several other modeling approaches which account for both the variation in experts'

[20] Note that these experiences are not limited to the previous version of this specific product. Managers were asked to include experiences obtained through the completion of *any* software project. Given that most Internet software products in 1996–1998 were first or second generation, most of the experience we capture was necessarily gained with different products. This measure is therefore not connected to the maturity of products in this industry.

[21] We conducted a number of tests to establish whether the characteristics of the specific project being undertaken influenced the process measures that we captured. Specifically, we categorized projects based upon whether they were real-time systems, applications, or websites, and also captured the percentage of new code developed in each project. None of these variables were associated with significant differences in any of our process measures.

Table 2 Models Predicting Product Quality with Single Process Measures

MODEL	I	II	III	IV	V	VI	VII	VIII
Constant	4.417****	4.410****	4.440****	4.438****	4.457****	4.456****	4.303****	4.193****
	(0.085)	(0.081)	(0.073)	(0.071)	(0.084)	(0.084)	(0.163)	(0.180)
Control: resources[a]		0.304* ($t = 1.84$)		0.244* ($t = 1.71$)		0.150 ($t = 0.89$)		0.260 ($t = 1.33$)
Architecture design effort[b]	1.378**	1.732***						
	(0.589)	(0.594)						
Market feedback: First proto/beta[b]			−0.310****	−0.356****				
			(0.076)	(0.079)				
Tech. feedback: First integration[b]					−0.011**	−0.012**		
					(0.005)	(0.005)		
Generational experience							0.002	0.004
							(0.003)	(0.003)
R-squared (adj)	15.2%	22.8%	35.6%	39.9%	14.8%	14.2%	0%	2.2%
F-ratio	5.47	4.70	16.50	10.30	5.88	3.32	0.80	1.29
Df	24	23	27	26	27	26	25	24

****$p < 0.1\%$, ***$p < 1\%$, **$p < 5\%$, *$p < 10\%$ NB: Unless stated, numbers in brackets are standard errors.
[a] = adjusted for product complexity; [b] = adjusted for scale effect.

assessments of quality, and the differing confidence levels of experts.[22] We are satisfied the results are robust to these different analytical methods; hence, for simplicity, we report only the OLS results here.[23]

Table 2 describes a series of models predicting product quality using single measures of the development process. We report results both with and without the use of resources as a control. Note that a model predicting product quality using resources alone is not significant. Our hypothesis is that if such a relationship exists, it might emerge only after the variance attributable to differences in the development process has been explained.

Models I and II test the association between product quality and investments in architectural design. In both these models, this variable is significant, indicating that projects which allocate proportionately greater resources to the design of the product architecture tend to result in higher quality products. In Model I, this variable explains over 15% of the variance in outcomes. In Model II, which includes a control for resources, over 22% of the variance in outcomes is explained. In this latter model, a weak trade-off between quality and resources is apparent, and the significance of investments in architectural design is noticeably stronger.

Models III and IV test the association between product quality and early market feedback. In both these models this variable is significant, indicating that projects in which greater proportions of functionality are developed after feedback is received from the market tend to result in higher-quality products. In Model III, this variable explains over 35% of the variance in outcomes. In Model IV, which includes a control for resources, over 39% of the variance in outcomes is explained. In this latter model, a weak trade-off between quality and resources is apparent. Models V and VI test the association between product quality and early technical feedback. In both these models this variable is significant, indicating that projects in which greater proportions of functionality are developed after the first system integration tend to result in higher-quality products. In Model V, this

[22] Other modeling approaches included: OLS regression using a weighted mean quality score obtained by weighting expert ratings by level of confidence, WLS regression using data on the standard error of quality ratings to weight observations, OLS regression using data from each expert as an independent outcome ($n = 204$), WLS regression using data from each expert as an independent outcome, and weighting expert rating by level of confidence ($n = 204$).

[23] In our regression models, significance levels are reported for two-tailed tests.

variable explains over 14% of the variance in outcomes. In Model VI, which includes a control for resources, similar results are seen.

Note that the explanatory power of early technical feedback is significantly lower than that of early market feedback, despite the high correlation between these variables (see Appendix D). Our interpretation of this result is that even though there is significant uncertainty in this environment, these uncertainties are manageable from a technical standpoint, given that the nature of the underlying technology (i.e., software code) is relatively well understood by firms. What is not well understood, however, is the requirements that *customers* have of products, both in terms of the overall architecture which best captures their desired performance trade-offs, and the end-use context in which the product must operate. While the development process must possess mechanisms to identify problematic interactions between technical components, the most troublesome interactions are in fact more likely to arise between the product and its operating environment. With such a view, an early integration of the system is only the first step in establishing a link to this environment through which a project can begin receiving feedback.

Models VII and VIII test the association between product quality and generational experience. In neither of these models is this variable significant. This result was surprising, given that our fieldwork had indicated that this type of experience was extremely valuable in a flexible process. We heard many examples of how team members' experiences from previous projects helped them to rapidly interpret new information and subsequently add new functionality to a design as a project progressed. This suggested, however, that the benefits of generational experience might manifest themselves primarily through a reduction in the resources required to develop a product (as opposed to an increase in product quality).[24] This assertion is supported by the correlation between resources and generational experience (see Appendix D). Indeed, variations in the proportion of team members that have completed more than two previous project generations explains over 24% of the variance in resources. These results suggest generational experience might be significant in a model predicting product quality if a trade-off between quality and resources is also present (i.e., the benefit can then be taken either as a resource saving or an increase in quality). We therefore continue to include this indicator as a predictor in more complex models in case such a relationship is found.[25]

Table 3 describes a series of more complex models predicting product quality using multiple measures of the development process. We report all results using resources as a control. We do not include early technical feedback in these models given the high correlation this variable has with early market feedback, which is the better predictor.

In all these models, the control for resources emerges as a significant predictor of product quality. After adjusting relative performance for differences in the development process, higher quality products are associated with the use of additional resources. Note also that generational experience is associated with product quality in those models where the relationship between quality and resources is most significant. Our final model, which contains all three measures of the development process and the control, explains 54.2% of the variance in product quality.

5. Discussion

The results described above highlight the importance of generating new information on performance through establishing early links to the market. Variations in the percentage of functionality added to the product after first prototype and first beta predict more than one-third of the variance in product quality. This is an intriguing result. It suggests that in a flexible process, development teams should focus, above all, on getting an early (and by definition, incomplete) version of the product into customers' hands at the first opportunity. Thereafter, teams must work with these customers to "coevolve" the design,

[24] The productivity of different software developers (e.g., as measured by lines of code per person-day) has been shown to vary widely, sometimes by an order of magnitude (Cusumano 1991).

[25] We also captured data on the average experience of team members in years. We found no relationship between this measure of experience and either product quality or resources.

Table 3 Models Predicting Product Quality with Multiple Process Measures

MODEL	IX	X	XI	XII
Constant	4.423****	4.070****	4.304****	4.176****
	(0.067)	(0.149)	(0.151)	(0.133)
Control:	0.369**	0.545***	0.341**	0.532***
Resources[a]	($t = 2.68$)	($t = 3.13$)	($t = 2.11$)	($t = 3.54$)
Architecture	1.326**	2.097****		1.655***
design effort[b]	(0.505)	(0.549)		(0.496)
Market feedback:	−0.270***		−0.297***	−0.225***
First proto/beta[b]	(0.079)		(0.083)	(0.077)
Generational		0.007**	0.002	0.005**
experience		(0.003)	(0.003)	(0.002)
R-squared (adj)	47.3%	38.4%	34.3%	54.2%
F-ratio	8.47	6.19	5.52	8.41
Df	22	22	23	21

****$p < 0.1\%$, ***$p < 1\%$, **$p < 5\%$, *$p < 10\%$ NB: Unless stated, numbers in brackets are standard errors.
[a] = adjusted for product complexity; [b] = adjusted for scale effect.

gathering feedback on the performance of existing features, while being responsive to requests for additional functionality.

Achieving such tight integration between an emerging design and its application context, however, is not trivial. Releasing beta versions of a system to customers when significant parts of the functionality remain incomplete creates a major architectural challenge. Work must be partitioned and prioritized in such a way that when only part of the functionality is developed, a few core modules can be integrated to provide an early working version of the system. While this version will have many features missing, it must contain the essence of the product, providing a baseline to which customers can react. Further challenges are created by the need to respond to new information later in the process, as the design nears completion. This requires an architecture which can accept new functionality in a way that minimizes changes to the rest of the system. Without such explicit architectural choices in the early stages of a project, the potential for responding to new information in the later stages is likely to be severely limited.

This discussion highlights the fact that investments in architectural design play a dual role in a flexible process. They are required both to evaluate and select between various architectural options, and also to facilitate the flexibility of the development process itself. Support for this view comes from the correlation between investments in architectural design and the percentage of product functionality developed after the first system integration (see Appendix D). This relationship suggests that projects that dedicate greater resources to architectural design tend to be able to integrate the product at an earlier stage of development.

Our field observations also support the view that investments in architectural design can facilitate a more flexible process. A program manager for Microsoft's Internet Explorer 3.0 project commented,

> ... the most important aspect of the project was that we developed the product architecture in a way that separate component teams could feed into the project. The idea was to build a good core infrastructure, and have the rest of the team add components on top of it. In fact, at the first integration, all we had was the core infrastructure. Most other features were missing.

Similarly, a manager at Altavista explained that their

> ... architectural design efforts are structured to give priority not to performance, but to independence. We create interfaces to buffer the impact of uncertainty—when one module changes, the others are therefore insulated. If we were trying to optimize the size and efficiency [of a design] we would not do this, but optimizing a design typically makes it more complex and subsequently very difficult to change.

The final topic our results illuminate is the subtle role of experience in projects which face uncertain and dynamic environments. Our hypothesis was that a team with greater generational experience would have a better ability to integrate new information into the design during later project stages, thus improving the flexibility of the development process. Indeed, support for this view comes from the correlation observed between our measure of generational experience and the percentage of product functionality developed after the first prototype and beta release (see Appendix D). However, while generational experience is a significant predictor of quality in our final model, the manner in which this effect emerges is indirect. Our analysis shows that generational experience is associated with the use of fewer resources to complete a project. Its association with product quality only arises in models where a relationship between quality and resources is also present (i.e., in more complex models). Our interpretation of this result is that in uncertain and dynamic environments, this type of experience is not useful in identifying the requirements of customers, given that these requirements are continually evolving. This experience is useful, however, in framing and directing the experimentation strategy employed in a way that the ongoing process of integrating new information into the design is performed in the most *efficient* manner. Given a trade-off between product quality and resources, this gain in efficiency can be taken either as a resource saving or as a quality gain.

6. Conclusion

This paper has demonstrated that several constructs which support a more flexible development process are associated with better-performing projects in a highly uncertain and dynamic environment. This flexible process is characterized by the ability to generate and respond to new information for a longer proportion of a development cycle. The constructs that support such a process are greater investments in architectural design, earlier feedback on product performance from the market, and the use of a development team with greater amounts of generational experience. In combination, these constructs explain over 50% of the variation in product quality across a sample of completed projects in the Internet software industry.

From a practitioner's perspective, implementing a more flexible process requires thinking about the product development process with a different mindset, given significant amounts of the perceived wisdom about good practice can run counter to the goals of such a process. Take, for example, the notion that a large number of design changes in the late stages is a sign of a poor project. This is not true in the environment we studied. Many excellent projects (as judged by the quality of the final product) made major changes to the design at late stages. Critically, however, these projects possessed a process which allowed them to do this.

An example of the change in mindset required in a flexible process comes from our fieldwork at one firm. The firm volunteered data on two projects, one which they identified as a "successful" project, and the other which they regarded to have been "poorly executed." When we analyzed the results, however, we found quite the reverse. The successful project scored lower on product quality and consumed relatively greater resources (adjusted for complexity) than its supposedly inferior partner. To identify why this misconception arose, we reviewed our notes of the interviews with each project manager. The "poorly executed" project had involved a process of continual change, as new developments emerged in the market and competitive products were launched with features which became competitive necessities. By contrast, the "successful" project was run in a very structured fashion, starting with a carefully optimized product architecture, and followed by an approach to development in which there was no flexibility to change this initial design. From the point of view of the senior manager who had selected these projects, the first appeared to be extremely chaotic in nature, given that the specification changed continually throughout development. By contrast, the second followed a more controlled process, arriving at product launch with a design which closely mirrored the initial specification. Through the lens of traditional development practices, the latter appeared to be a much superior process. The resulting design, however, being "frozen

in time" early in the project, was not well received by the time it reached the market.

For the academy, this research provides empirical validation of the value of more flexible models of development that have been proposed in the literature (e.g., Boehm 1988, Iansiti and MacCormack 1997). In particular, we have demonstrated the usefulness of a flexible model in an environment which presents extremely high levels of uncertainty to development teams. We note, however, that it is possible that the constructs we argue support a flexible process may also be associated with project performance in more stable environments.[26] Further work is needed to determine if this is the case. Indeed, fully exploring the details of a *contingent* view of product development process design would require a multi-industry study to identify which parameters should be regarded as universal "best practices," and which should be varied to suit the specifics of each particular context. This current study represents the first step along just such a path.

[26]We expect that in more stable environments, the explanatory power of these constructs would diminish, relative both to their impact in more uncertain environments and to the explanatory power of other variables that have been shown to predict project success in stable settings.

Acknowledgments

The authors gratefully acknowledge the industry experts who helped assess the quality of products in this research: Keith Allison, Whit Andrews, Christofer Barr, Rik Drummond, James Farley, Mark Frauenfelder, Carlo Ghezzi, Logan Harbaugh, Justin Hibbard, Kevin Jones, Eric Lundquist, Melanie McMullen, Shawn Rogers, Gus Venditto. We are also very grateful for the comments of Rocco Mosconi and two anonymous reviewers.

Appendix A Controlling Project Resources for Product Complexity

In order to control project resources for product complexity, we capture the number of uncommented lines of new code developed in each project. Although more advanced measures of software complexity have been proposed in the software engineering literature (e.g., Halstead and Marciniak 1994) these metrics are rarely computed by managers. We first compute the resources (in person-days) that would have been required in each project to develop an application of standard size (100,000 lines of code). Since software productivity exhibits scale effects (Banker and Kemerer 1989), we then model the relationship between standardized resources and size as follows:

$$\text{STDRESOURCES} = a^*(\text{NEWCODE})^{-b^*(1+c^*\text{SERVICE})}$$

Where:

STDRESOURCES:	person-days required to develop 100'000 lines of new code;
NEWCODE:	lines of uncommented code developed;
SERVICE:	dummy variable indicating project is to develop a service;
a, b, c:	parameters to be estimated.

Taking logarithms of each side we get:

$$\ln(\text{STDRESOURCES}) = \ln(a) - b^*(1 + c^*\text{SERVICE})^* \ln(\text{NEWCODE}), \text{ or}$$
$$\ln(\text{STDRESOURCES}) = d - b^* \ln(\text{NEWCODE}) - e^* \text{SERVICE}^* \ln(\text{NEWCODE});$$

where d, e are parameters to be estimated. The model predicts much of the variation in standardized resources ($F = 72.99$, $adj\ R^2 = 83.7\%$, $p < 0.01\%$). We use the residuals from this model as our measure of the excess of resources (or the lack of resources) allocated to a project, after controlling for its complexity.

Appendix B Controlling Measures of the Development Process for Scale Effects

In order to control for scale, we model several measures of the development process as a function of the logarithm of the lines of new code developed. In our analysis, we use the residuals from these models as our adjusted measure.

Table 4 Models Predicting Measures of the Development Process

MODEL	Architecture Design Effort	Functionality at First Prototype	Functionality at First Integration	Functionality at First Beta
Constant	1.027****	93.53***	133.502****	136.519****
	(0.172)	(25.59)	(21.21)	(19.02)
Log (lines of new code)	−0.072****	−4.848**	−6.733***	−5.388***
	(0.016)	(2.329)	(1.930)	(1.731)
R-squared (adj)	43.1%	10.3%	27.8%	23.1%
t-statistic	−4.55	−2.08	−3.49	−3.11
Df	24	27	27	27

****$p < 0.1\%$, ***$p < 1\%$, **$p < 5\%$, *$p < 10\%$

Appendix C Descriptive Statistics for Adjusted Variables

Table 5 Descriptive Statistics for Adjusted Variables

Variable[a]	Description	S. Dev.	Min	Max
Resources	Total development and test resources consumed in project adjusted for product complexity (Person-days/100'000 lines of new code)	0.54	−0.91	0.76
Architecture Design Effort	Ratio of architecture design resources to development and test resources adjusted for scale effect	0.14	−0.31	0.28
Market Feedback: First Proto/Beta[b]	Percentage of product functionality included in the first prototype and beta release, adjusted for scale effect (composite measure)	1	−2.33	1.76
Tech. Feedback: First Integration	Percentage of product functionality included in the first system integration adjusted for scale effect	19%	−30%	39%

[a]Measures have mean zero, given that they are residuals or a composite formed from residuals.
[b]The statistics for market feedback are not comparable to that of technical feedback given that the former is a composite measure, and has therefore been normalized (i.e., has mean zero and variance one).

Appendix D Correlation Table

The table below contains correlations for all measures in their final form (i.e., after adjusting for scale and complexity). We report the measures of percentage functionality completed at each of the three milestones separately for completeness. We also report correlations for the composite measure of early market feedback.

Table 6 Correlation Table for Measures of the Development Process

	Quality	Resources	Architect. Design Effort	Market Feedback: 1. Prototype	Tech. Feedback: Integration	Market Feedback: 2. Beta	Generation. Experience
Quality	1.000						
Resources	0.025	1.000					
Architecture Design Effort	0.431**	−0.323	1.000				
Market Feedback: 1. First Prototype	−0.415**	0.439**	−0.141	1.000			
Tech. Feedback: First Integration	−0.423**	0.294	−0.336*	0.618****	1.000		
Market Feedback: 2. First Beta	−0.636****	0.132	−0.279	0.459**	0.481***	1.000	
Generational Experience	0.176	−0.526***	−0.054	−0.384**	−0.223	−0.176	1.000
Market Feedback: First Proto/Beta	−0.616****	0.341*	−0.248	Not Applicable	0.643****	Not Applicable	−0.334*

****$p < 0.1\%$, ***$p < 1\%$, **$p < 5\%$, *$p < 10\%$

References

Argyris, C. 1977. Double-loop learning in organizations. *Harvard Bus. Rev.* **55**(Sep–Oct) 115–125.

Banker, R.D., C.F. Kemerer. 1989. Scale economies in new software development. *IEEE Trans. Software Engnrg.* **15**(10) 1199–1206.

Bhattacharya, S., V. Krishnan, V. Mahajan. 1998. Managing new product definition in highly dynamic environments. *Management Sci.* **44**(11 Part 2) S50–S64.

Boehm, B. 1988. A spiral model of software development and enhancement. *IEEE Comput.* **21** 61–72.

Christensen, C.M., R. Rosenbloom. 1995. Explaining the attacker's advantage: Technological paradigms, organizational dynamics, and the value network. *Res. Policy* **24** 233–257.

Clark, K.B., T. Fujimoto. 1991. *Product Development Performance.* HBS Press, Boston, MA.

Connell, J.L., L. Shafer. 1989. *Structured Rapid Prototyping: An Evolutionary Approach to Software Development.* Yourdon Press, Englewood Cliffs, NJ.

Cooper, R.G. 1990. Stage-gate systems: A new tool for managing new products. *Bus. Horizons* **33**(3) 44–54.

——. 1995. Developing new products on time, in time. *Res.-Tech. Management* **38**(5) 49–57.

——, E.J. Kleinschmidt. 1996. Winning businesses in product development: The critical success factors. *Res.-Tech. Management* **39**(4) 18–29.

Cusumano, M.A. 1991. *Japan's Software Factories.* Oxford University Press, New York.

——, K. Nobeoka 1992. Strategy, structure and performance in product development: Observations from the auto industry. *Res. Policy* **22** 265–293.

——, R. Selby. 1995. *Microsoft Secrets.* Free Press, New York.

Dalkey, N., O. Helmer. 1963. An experimental application of the delphi method to the use of experts. *Management Sci.* **9**(3) 458–467.

Eisenhardt, K.M., B.N. Tabrizi. 1995. Accelerating adaptive processes: Product innovation in the global computer industry. *Admin. Sci. Quart.* **40** 84–110.

Fiol, C.M., M.A. Lyles. 1985. Organizational learning. *Acad. Management Rev.* **10** 803–813.

Halstead, M., J.J. Marciniak. 1994. *Encyclopedia of Software Engineering.* John Wiley & Sons, New York.

Henderson, R., K.B. Clark. 1990. Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Admin. Sci. Quart.* **35** 9–30.

Iansiti, M. 1997. *Technology Integration: Making Critical Choices in a Dynamic World.* HBS Press, Boston, MA.

——, MacCormack. 1997. Developing products on Internet time. *Harvard Bus. Rev.* **75**(Sep–Oct) 108–117.

Katz, R., T.J. Allen. 1982. Investigating the not-invented-here (NIH) syndrome: A look at the performance, tenure and communication patterns of 50 R&D project groups. *R&D Management* **12**(1) 7–19.

Krishnan, V., S.D. Eppinger, D.E. Whitney. 1997. A model based framework to overlap product development activities. *Management Sci.* **43**(4) 437–451.

Levinthal, D., J.G. March. 1981. A model of adaptive organizational search. *J. Econom. Behavior & Organ.* **2** 307–333.

Linstone, H.A., M. Turoff, eds. 1975. *The Delphi Method: Techniques and Applications.* Addison-Wesley, Reading, MA.

MacCormack, A. 1998. Managing Adaptation: An Empirical Study of Product Development in Rapidly Changing Environments. Unpublished doctoral dissertation, Harvard University, Boston, MA.

March, J. 1991. Exploration and exploitation in organizational learning. *Organ. Sci.* **2**(1) 71–87.

McKee, D. 1992. An organizational learning approach to product innovation. *J. Product Innovation Management* **9** 232–245.

Nelson, R., S. Winter. 1982. *An Evolutionary Theory of Economic Change.* Harvard University Press, Cambridge, MA.

Parnas, D.L. 1972. On the criteria to be used in decomposing systems into modules. *Comm. ACM* **15** 1053–1058.

Reid, R.H. 1997. *Architects of the Web.* John Wiley and Sons, New York.

Royce, W.W. 1970. Managing the development of large software systems: Concepts and techniques. *Procedures* WESCON, Western Electric Show and Convention, Los Angeles. Reprinted 1989 in *Proc. 11th Int. Conf. of Software Engnrg.* Pittsburgh, PA.

Sanchez, R., J.T. Mahoney. 1996. Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management J.* **17** 63–76.

Thomke, S. 1997. The role of flexibility in the development of new products: An empirical study. *Res. Policy* **26** 105–119.

——. 1998. Managing experimentation in the design of new products and processes. *Management Sci.* **44**(6) 743–762.

——, E.A. von Hippel, R.R. Franke. 1998. Modes of experimentation: An innovation process variable. *Res. Policy* **27** 315–332.

Tushman, M.L., C.A. O'Reilly. 1997. *Winning Through Innovation.* HBS Press, Boston, MA.

Ulrich, K.T. 1995. The role of product architecture in the manufacturing firm. *Res. Policy* **24** 419–440.

——, S.D. Eppinger. 1995. *Product Design and Development.* McGraw-Hill, New York.

Verganti, R. 1999. Planned flexibility: Linking anticipation and reaction in product development projects. *J. Product Innovation Management* **16**(4) 363–376.

Von Hippel, E. 1986. Lead users: A source of novel product concepts. *Management Sci.* **32**(7) 791–805.

——. 1988. *The Sources of Innovation.* Oxford University Press, New York.

Wheelwright, S.C., K.B. Clark. 1992. *Revolutionizing Product Development.* Free Press, New York.

Wong, C. 1984. A successful software development. *IEEE Trans. Software Engnrg.* **SE-10**(3) 714–727.

Accepted by Karl Ulrich; received April 18, 1999. This paper was with the authors 8 months for 2 revisions.