Product Development Process Design: Improving Development Response to Market, Technical, and Regulatory Risks

By:
Darian W. Unger

B.S., Engineering
Swarthmore College, 1995

B.A., Political Science,
Swarthmore College, 1995

S.M., Technology and Policy
MIT, 1999

S.M., Civil & Environmental Engineering
MIT, 1999

Submitted to the Engineering Systems Division in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Engineering Systems Management and Policy
at the
Massachusetts Institute of Technology
June, 2003.

Signature of Author.....
MIT Engineering Systems Division
Technology Management and Policy Program
May 2, 2003

Certified by.....
Prof. Steven Eppinger
General Motors LFM Professor of Management and Engineering Systems
Co-Director, MIT Leaders for Manufacturing Program and System Design and Management Program
Advisor and Chairman of Doctoral Committee

Certified by.....
Prof. Christopher Magee
Professor of the Practice of Mechanical Engineering and Engineering Systems
Director, MIT Center for Innovation in Product Development
Committee Member

Certified by.....
Dr. Daniel Whitney
Senior Research Scientist, MIT Center for Technology Policy and Industrial Development
Committee Member

Certified and accepted by.....
Prof. Daniel Hastings
Professor of Engineering Systems and Aeronautics and Astronautics
Director of the MIT Technology Management and Policy Program
Chair, Committee on Graduate Studies and Committee Member

Product Development Process Design:
Improving Development Response to Market, Technical, and Regulatory Risks

By: Darian W. Unger

Submitted to the Engineering Systems Division in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Engineering Systems Management and Policy
May 2003

ABSTRACT

Engineering companies frequently face product development challenges. Competitive pressures, industrial or societal innovations, and government regulations are some of the many factors that drive the need for new or better products. Companies respond to these drivers and changing needs by developing new products and employing product development processes (PDPs) to coherently manage the risks inherent in their development. Well-designed PDPs reduce development time, create better products, generate profit, and increase market share. In contrast, poorly-designed PDPs can severely harm both product lines and the companies that manufacture them. Many companies seek guidance in making important PDP design decisions.

This thesis introduces PDPs as risk management frameworks. The research investigates the relationship between PDPs and risk management and seeks to help companies improve PDP design. It begins by discussing the drivers and risks of product development and then describes different PDPs. The traditional stage gate process is compared with the modified waterfall process, evolutionary prototyping, evolutionary delivery, design to schedule/budget process, the spiral process, and several other PDP variations. The research then proposes several iteration- and review-based metrics by which PDPs can be more effectively identified and compared.

Ten company case studies exemplify a wide variety of actual PDPs, demonstrate the utility of iteration and review metrics in distinguishing PDPs, and illustrate how different processes manage different risks. Case study findings indicate that software development companies face rapidly-changing markets, generally perform quick integrations and tests, and are likely to employ flexible PDPs. In contrast, manufacturing companies that face greater integration difficulties and technical risks are likely to employ more rigid PDPs. Integration and risk are both instrumental in determining the applicability of different PDPs. The research employs case study lessons to propose a method for improved PDP design based on risk and integration. To demonstrate the method, it is applied to one company.

The thesis concludes that PDPs vary more than previously documented; that the proposed metrics are useful in distinguishing PDPs, their different integrations, and their different risk management methods; and that companies facing different risks can more thoughtfully tailor their PDP designs to suit their own unique circumstances.

Thesis Supervisor: Steven Eppinger, Ph.D.
General Motors LFM Professor of Management and Engineering Systems
MIT Sloan School of Management and School of Engineering
Co-Director, MIT System Design and Manufacturing Program and Leaders for Manufacturing Program

Doctoral Committee: Steven Eppinger, Ph.D.
Professor of Management and Engineering Systems
MIT Sloan School of Management and School of Engineering
Director, MIT System Design and Manufacturing Program and Leaders for Manufacturing Program

Daniel Hastings, Ph.D.
Professor of Engineering Systems and Aeronautics & Astronautics
MIT School of Engineering
Director, Technology Management and Policy Program

Christopher Magee, Ph.D.
Professor of the Practice of Mechanical Engineering and Engineering Systems
MIT School of Engineering
Director, Center for Innovation in Product Development

Daniel Whitney, Ph.D.
Senior Research Scientist
MIT Center for Technology Policy and Industrial Development

ACKNOWLEDGMENTS

I would like to thank the many people who helped me during my time at MIT. In particular, I would like to thank Steve Eppinger for being not just my advisor, but my mentor as well. I have been grateful for his support, guidance, and friendship ever since he brought me on board at the MIT Center for Innovation in Product Development. I also thank Daniel Whitney for his helpful and sharpening suggestions, Daniel Hastings for his assistance and confidence, and Chris Magee for his excellent guidance and conversation.

I also offer thanks to my previous advisors and professors at MIT who helped me along my way, including Howard Herzog, Elisabeth Drake, John Heywood, Richard Tabors, Richard de Neufville, Maurice Holmes, Malcolm Weiss, and Arnoldo Hax. I gratefully acknowledge the support of the MIT Energy Laboratory/Laboratory for Energy and the Environment, the MIT Center for Innovation in Product Development, and the Singapore-MIT Alliance. I also met with, questioned, and interviewed many people during this study and thank them all for sharing information, data, their opinions and their time.

Many thanks and love also go to

- My wonderful parents, of whom I'm so proud and who simultaneously encouraged and delayed this dissertation with their many loving phone calls.
- Felek and Vera, who are my second loving and amazingly supportive family in Boston.
- Noam, who joined us in Boston to my delight and made me realize how much fun it is to live in the same town as my brother.
- My friends and roommates, Scott, Mike, and Sylvia, for providing laughs and often quizzical looks during stressed days – and for making academia seem like paradise by demonstrating the horrors of early-morning hours in the working world.
- Jill, for being amazingly cute and for loving me back.

I am grateful to the MIT Sloan School of Management for providing free food in times of adversity. It was tasty. Finally, I acknowledge the support of the American beef industry in providing me with the many pounds of meat I used to host barbecues during my years at MIT.

TABLE OF CONTENTS

1. INTRODUCTION

"It is not the employer who pays wages – he only handles the money. It is the product that pays wages." – Henry Ford

Successful product development is critical to industrial performance. Rapid and innovative product development (PD) can provide critical competitive advantages to firms. (Rosenau & Moran, 1993; Ulrich & Eppinger, 2000) The pressure to improve products and PD processes is evident in the words of the Vice President of R&D of Grace Performance Chemicals:

> Today, more than ever, the only way for any...industrial organization to stay competitive is to be more creative, more innovative, and faster than the competition. We need to continuously introduce better and less expensive products and technologies. (Jachimowicz, 2000)

Despite the importance of PD, technology management in this field leaves considerable room for improvement. Companies currently have difficulty choosing from a long and complex menu of PD processes; if they design a process poorly, they may endanger the success of their products, their competitiveness, and possibly their survival. There are currently no established criteria for comparing, selecting or designing ideal PD processes; nor is any single process ideal for all circumstances and companies.

This thesis exhibits and explains various PD processes (PDPs) and aims to help companies better design their own processes. Using existing literature and research, this thesis proposes risk management characteristics to describe and compare different PDPs. It then uses ten case studies to examine the variation among PDPs and to demonstrate the usefulness of the proposed metrics. Finally, it uses the lessons of the case studies to suggest a framework for improved PDP design and selection based on risk.

This research introduces PDPs as risk management structures. Although PD is valuable as a source of competitive advantage, it is risky. PD processes must address different kinds of risk: they must overcome various technical challenges, meet changing market needs and keep on time and on budget. PDPs manage risk, but this thesis demonstrates that they do so in dramatically different fashions.

The study is divided into three parts. Part I (Chapters 2, 3, and 4) explains the context and drivers of PD, displays a menu of PDPs that companies use in developing new products, and defines key risk management characteristics that all PDPs share. Part I is based mainly on previous research and field assessments, but also includes several new metrics for defining PDPs. Part II (Chapters 5, 6 and 7) explains the conduct, results, and findings of ten company case studies. The empirical case studies showcase a wide variety of actual PDPs and demonstrate that risk management parameters provide a useful means of comparison among them. Part III (Chapters 8 and 9) extrapolates from the case study lessons to propose a framework for improved PDP design and selection. The framework is applied at one company to demonstrate its potential usefulness. The study concludes that PDPs vary more than previously documented, that risk management metrics are useful in distinguishing PDPs, and that PDP design and selection can be more thoughtfully tailored for companies facing different risks.

The study begins with Chapter 2, which examines the context and drivers of PD. Existing literature demonstrates that driving forces behind innovation and PD can include customers, competitors, and government action. Customers can change market demand, forcing companies to improve or change their products. Competitors can prompt the need for PD by developing lower-cost or better-selling alternatives. Government agencies can set new technology requirements or use other regulatory tools to prompt companies to incorporate changes into their products, especially in the energy and manufacturing arenas where product externalities are common. Together, these upstream drivers create a series of risks which companies must face while developing new products. Product integrations, models, tests or prototypes help some companies to address development risks early in development. A contextual case study of gas turbine development history demonstrates many of the drivers cited earlier in the chapter while simultaneously providing the context for the PDP analysis of turbine development in Chapter 6. The chapter concludes by linking these upstream drivers to PD risks and concepts and defining the problem of PDP comparison.

Chapter 3 showcases a variety of common PDPs. The traditional stage gate process is compared and contrasted with the modified waterfall process, evolutionary prototyping, evolutionary delivery, design to schedule/budget process, the spiral process, and several other PD variations. Although the processes have several steps in common, they progress through those steps in very different ways. The advantages and disadvantages of each process are described; no single process is suitable for all PD circumstances.

Chapter 4 proposes metrics by which the potential of different PDPs to manage risk can be compared and contrasted. Development risks can include technical, market, budget, and schedule risks. To manage these risks, PDPs employ design reviews, which uphold standards or mark milestones, and iterations, which are changes and feedback loops between design groups or project phases. All PDPs use reviews and iterations, but the manner of reviews and iterations varies dramatically. They may differ in frequency, rigidity, and several other characteristics. Thus, reviews and iterations – incorporating specifications, milestones, integrations, and tests – are advanced as useful metrics for distinguishing PDPs.

In Part II, Chapter 5 details the methodology of the empirical research in this study. Elements of reverse case study analysis (RSCA) and grounded theory building (GTB) methodologies were used to develop a series of company case studies. Interviews and questionnaires were designed to gather both qualitative and quantitative PDP data. The groundwork is laid for empirical case studies representing an array of industries and different environments.

Chapter 6 includes ten company case studies and comparative findings. Interviews, questionnaire responses, and process document excerpts are used as evidence in the description and categorization of each company's PDP. The metrics discussed in Chapter 4 are used to describe and distinguish the different company PDPs. All of the companies face unique risk profiles but use at least one form of iteration or design review to manage their risks. The PDPs of most companies match one of the previously-described

theoretical processes closely. For example, some companies are found to follow stage gate processes, while another uses an evolutionary delivery approach.

Chapter 7 draws lessons from the case studies and demonstrates that the metrics proposed earlier are indeed useful descriptive tools for PDP analysis. The lessons also illustrate how effectively different companies' processes manage various types of risk. Threats to research validity are addressed, and potential alternative criteria in describing or designing PDPs are discussed. The results suggest that there is more to PDP description than mere risks and metrics. Integrations or prototypes play an important role in determining the effectiveness and applicability of different processes

Chapter 8 uses the case study results and lessons to propose a framework for improved PDP design and selection. Using risk management concepts and the parameters supported by the case studies, a PDP design method is suggested. This PDP design method is applied at a company seeking to improve its PDP. The method's application and ease of understanding within the company supports its utility.

The study concludes that for a PDP to be effective, it should be designed to manage company-specific development risks and include integrations to provide early feedback whenever practical. Current PDP selection is inconsistent and PD literature – if it even recognizes a variety of choices – offers only limited guidance regarding the strengths and weaknesses of different PDPs. This research improves the understanding and implementation of PD in three ways. First, it demonstrates how PDPs differ by defining and comparing key characteristics. Second, it offers lessons on how PDP iterations and reviews manage risks differently. Finally, it proposes a PDP design method that companies can use to improve their management of development risks. Together, these contributions aim to improve the academic understanding of management processes and to improve business response to the changing regulations and market needs that drive product development.

2. THE CONTEXT OF PRODUCT DEVELOPMENT: INDUSTRY, POLICY, AND SOURCES OF INNOVATION

God has made man upright, but they have sought out many inventions.
– Ecclesiastes, 7:29

2.1 Introduction

Product development is both a cause and effect of industrial and social progress. This section explains the context and drivers of product development because of their importance in shaping development risks and the ability of companies to build prototypes or integrated models of their products. These risks, in turn, influence the product development processes that companies employ.

Figure 2.1 shows a technology strategy chain linking drivers of innovation to actual product development. The chain places PD in the context of overall technology strategy, where PD is the last and often most critical part. Drivers of innovation include profit potential and spurs external to market forces, such as government action. Companies respond to the innovation challenges in a variety of ways, depending on their organizations, the type of innovation necessary, and their ability to create and capture value from the innovation. The resulting risks feed into the product development cycle.

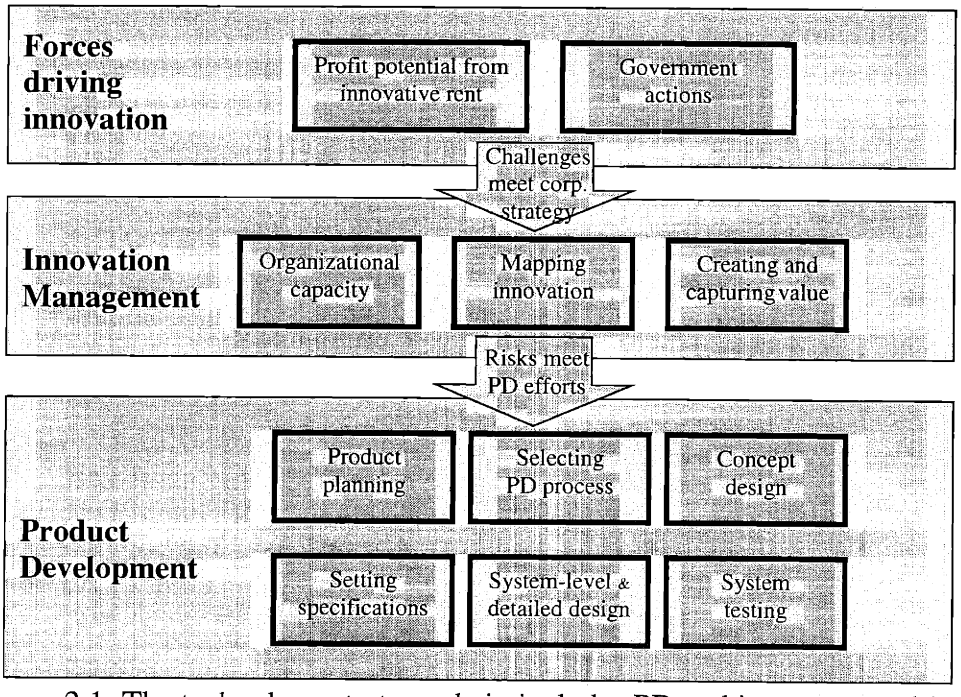


Figure 2.1: The technology strategy chain includes PD and its upstream drivers

Understanding PD in the context of general technology strategy is important because PD does not occur in vacuum. This chapter provides the context and background necessary to understand what shapes development risks and how PDPs then address those risks. Section 2.2 introduces the forces driving innovation. Section 2.3 explains key areas of innovation management, providing the link between innovation and PD. Section 2.4 describes a case study of the gas turbine industry to illustrate how these upstream factors shape PD risks. Finally, Section 2.5 introduces basic PD concepts, provides a literature review of the field, and defines the problem of PDP comparison and design that this thesis addresses.

2.2 Forces driving technological innovation

Previous research and literature suggest that a variety of forces drive technological innovation. Here, innovation is defined as the application of an invention, though not necessarily in the form of a final product. This section examines the two categories of drivers shown in Figure 2.1 – profit potential and government actions. Together, they provide companies with the basic incentives to innovate and develop products.

2.2.1 Profit potential from innovative rent

In most cases, market dynamics and profit potential are the root causes of PD. If a potential product will have market value, the company that develops it can reasonably hope to create it and capture some of the value as profit, which spurs development efforts. (Henderson, 1994)

An excellent way to pinpoint the sources of innovation is to follow profit potential and the trail of "innovative rent" to see who profits most from an innovation. The profit motive for innovation frequently affects product manufacturers. Von Hippel (1988) demonstrates that lead users and suppliers can also play a large role in innovation and PD because they are frequently familiar with either the applicable technologies or acutely tuned to market needs. Furthermore, the development of a new product can sometimes mean increased profits for the user or supplier as well as (or instead of) the original

manufacturer. A variety of actors may innovate and develop new products, but this fundamental economic incentive drives them all. (Wheelwright & Clark, 1992)

2.2.2 Government influence

Government actions also influence innovation and PD, usually when a product exhibits market externalities, is subject to standards, or is likely to lead to a question of legal liability. This occurs frequently, as in the case of power plant emission regulations or automobile safety standards. Whenever these regulations change, manufacturers must develop new products to meet the new standards. (McGrath, 2001) This section briefly describes three major ways in which government can impact PD: technology-forcing regulations, direct R&D spending, and judicial imposition of liability.

Technology-forcing regulations

Regulation is by far the most powerful government impact on innovation and PD. Regulations can be helpful or burdensome, and may either induce or stifle innovation. (Nelson, 1995)

Environmental regulation is a common driver of innovation. For example, air quality laws may prompt the innovation, development and use of catalysts to reduce power plant emissions. However, different types of regulation may produce drastically different PD results. One type of clean air standard could force an industry to develop a new product as part of a long-term cleanup process. Another standard could instead stress feasibility by forcing immediate compliance based on the Best Available Control Technology (BACT). The first standard prompts immediate innovation and PD in the longer run. The BACT standards, on the other hand, promote the diffusion and adoption of current technologies, leading to less innovation but more immediate product output based on existing technology. The regulators must decide for each circumstance whether diffusion or development of technology is the worthier goal. (Ashford & Caldart, 1996) This is analogous to the natural tension inherent in patents, where the promise of a patent grant may serve as an incentive to innovate in the beginning. Once an invention is patented, however, the one-time inducement can sometimes become a barrier to further

development, especially if it creates a safe monopoly for the patent holder or licensee, thus reducing the need to update or respond to a competitive threat.

Product development may also be influenced by whether regulations are performance-based or technology-based. Returning to the same example of an environmental regulation, an agency can opt either to require a specific device – such as a specific catalyst – or to promulgate a performance standard defining the parts-per-million allowances at the end of a pipe or stack. Both may require the development of new products, but the latter method may be a greater spur to innovation and allow for more variety in product development. (Porter & Van der Linde, 1997)

These regulatory issues are universal and manifest themselves across companies, industries, technologies, and agencies. They may affect products as diverse as computers, dolls, cars and power plants. For example, the Department of Energy regulates an incentive program called "Energy Star," which allows computer manufacturers to place a prestigious (and market share gaining) emblem on their products if they meet threshold energy efficiencies. This is an incentive for improved hardware and power-saving screens. The Consumer Products Safety Commission regulates the safety of children's toys so that some dolls must attain a certain level of fire resistance. How manufacturers and toy developers attain that resistance is up to them, leading to the development and adoption of fire-retardant materials. The Environmental Protection Agency sets both technology and performance regulations regarding automobiles and fuels. For example, a technology standard forced the development and adoption of catalytic converters in the 1970s and 1980s, while performance-based fuel economy standards allow manufacturers to develop their own means for achieving required auto efficiencies. Similar performance-based regulations are established indirectly by the Clean Air Act of 1973 (and its 1991 amendments) to limit particulates, carbon monoxide, SO_x and NO_x emissions from electric power plants. In response, power generators can switch fuels, limit production, improve machinery, or change operations. Some of these actions lead to new product development by either the generators or their suppliers. (Unger, 2001)

Direct research and development

Government may also spur innovation by directly funding R&D. The US federal government is not only the nation's largest consumer and employer; it is also the largest single investor in R&D. (Haggerty, 1973; Nelson, 1995) Most government R&D funding is devoted to basic research to fund pre-commercial science and technology. Sometimes, these projects succeed through the efforts of national laboratories or subsidized companies and become commercially viable technologies or products.

There are many examples of products stemming from both internal and external government R&D. In one instance of internal development, NASA engineers independently designed and developed a probe for Martian soil. The probe, with an array of sensors at the tip, was later licensed to the Bioluminate company for commercial use as a noninvasive detector for cancer in human tissue.[1] Earlier NASA development programs led to the creation of finite element analysis codes, such as Ansys and computation fluid dynamics tools. These tools became products because they filled the needs of companies that had to design a variety of complex parts and flows, ranging from turbomachinery to automobiles.[2] Externally, NASA operates several industrial partnership plans that provide seed money to US companies developing concepts and products for potential NASA missions. One example of this government funding is a grant distributed to Osmoteck Inc., a company that developed a direct osmosis water treatment system for purifying space station wastewater. A spinoff product extracts water from leachate in landfills to prevent soil and river contamination. (NASA, 2000) Partnerships are not restricted to any one government agency or to only small businesses. For example, the US Partnership for a New Generation of Vehicles coupled Department of Commerce funding with the big 3 US automakers. (National Research Council, 1998)

---

[1] From interview with David Lackner, Technology Commercialization Manager , NASA Ames Research Center.

[2] From interview with Lee McLurin, Combustion Turbine Development Manager, Siemens-Westinghouse Power Generation.

Judicial imposition of liability

While the legislative and executive branches regulate safety and environmental standards, the judicial branch of government may also influence innovation and PD. Even without legislative safety standards, companies must reckon with potential lawsuits on the judicially-imposed basis of strict liability. Even if a company is not negligent in developing and manufacturing a product (i.e. even if regulation-required fire retardant material is used on a doll, or even if Coca-Cola prints all its ingredients on a bottle as required) it may be held liable for certain damages. Exposure to such lawsuits can sometimes prompt companies to develop their products to increase safety or quality assurance even in the absence of regulation. (Ashford & Caldart, 1996)

The law can also force the diffusion of new technologies. A landmark case in this field was the TJ Hooper decision of 1932. (2nd Circuit Court of Appeals, 60 F.2d 737; 1932) In that case, a barge company hired a tugboat company to bring a cargo ship from one port to another. A storm hit during the transfer and the cargo ship was lost. In the ensuing lawsuit, the tugboat company claimed that the storm was "an act of God" and that it was therefore not negligent or liable for the losses. There was no radio (a relatively new invention) on the tugboat, so there was no way for the ship's captain to know about the storm in advance. The decision, written by Judge Learned Hand, revolutionized the shipping and radio industries. The ruling stated that the tugboat *should have had* a radio on board, even though there was no law specifically requiring it:

> A whole calling may have unduly lagged in the adoption of new and available devices...Courts must in the end say what is required."

Even without any related laws or regulations, the US shipping industry quickly adopted on-board radio. The liability ruling galvanized an industry and *created* a need that could only be addressed by a newly developed product. This remarkable decision is a salient example of courts mandating the diffusion of new technologies.

2.3 Innovation management

Once prompted to innovate by market forces or government actions, companies face a broad array of innovation management issues. As companies move from basic research

and innovation to actual PD, prior research suggests that companies must address their own organizational capacity, assess the new innovation on a technology map, and add customer value while capturing some of that same value for themselves. Each of these issues is discussed individually in this section.

2.3.1 Organizational capacity

Organizational capacity is critical for any company endeavoring to develop new products. (Cusumano & Nobeoka, 1999) The ability of the organization may be limited by its human resources, the skills and knowledge of its employees, its absorptive capacity, its output or manufacturing capability, and its internal control and communication. The number and skills of a company's employees can of course determine the level and type of effort that a company can invest in new PD. If internal knowledge or capacity is limited, a company can expand either physically (by growing or acquiring) or by improving absorptive capacity, which is the ability to recognize and assimilate new/external information that is critical to innovative capabilities. (Fine & Whitney, 1997; Cohen & Levinthal, 1990) Finally, companies must address their own communications and control mechanisms. For example, engineering groups within a company may be organized either along functional or product lines. The differences between how workers are organized, or "siloed," may distinguish who communicates most often with whom. If one group makes a change to a plan or product, they usually must provide notice to and receive feedback from other affected groups as part of any development process. Difficulties in this organizational necessity can lead to trouble, while smooth internal communication can lead to innovation and development success.

2.3.2 Understanding and mapping innovation

A company's view of a technological innovation may profoundly affect how it develops the technology into a product. PD risks may be entirely different depending on whether an innovation is considered to be either a significant leap or an ordinary next step. Understanding and mapping where an innovation stands in the context of others is important to any development strategy. Previous literature suggests several relevant theories on mapping innovation and development.

A common mapping device is the S-curve, similar to a learning curve, typified in Fig. 2.2, which demonstrates how a generic new technology may develop or enter a market. As engineering development continues, a new technology improves and then matures. Previous work divides the curve into three distinct stages and also distinguishes between development that rises on an S-curve and development that moves to an altogether new and higher S-curve. (Foster, 1986)

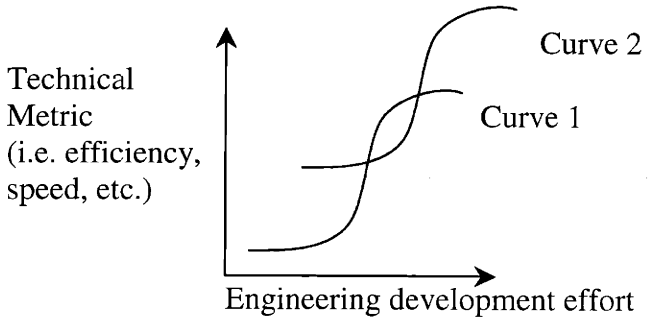


Fig 2.2: A typical technology S-curve

The S-curve is only one of many possible innovation maps. Other authors have mapped innovation and new product development in different ways. For example, Figure 2.3 categorizes types of innovations and demonstrates Henderson's case for why sometimes even minor innovations can have enormous competitive consequences. If a new product has the same basic physical components but a different linkage between those components – a seemingly minor difference – the result may be a surprisingly major change in product architecture, as shown in the bottom left quadrant. For example, a minor change in an automobile motor's calibration may cause an unexpected vibration in the hood metal – an altogether different product component. However, because neither the motor nor the hood are new, the true challenges of this seemingly minor innovation are not noticed as readily as radical innovations and developments. (Henderson and Clark, 1990) It is further argued that these architectural and radical innovations are two of several reasons why incumbent firms sometimes fail and lose their dominance with the introduction of new technologies and products. (Christensen, 1997)

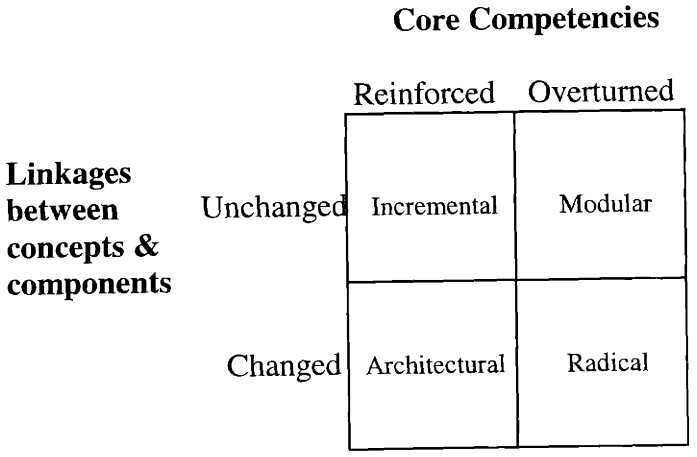


Figure 2.3: A Map of Innovation (Henderson & Clark, 1990)*

The main commonality among these various mapping schemes is that they all attempt to place new innovations and developments in context. The context is important to technology and development managers because it helps them determine both threats and opportunities in new PD. A manager may treat a development project very differently depending on, for example, whether the product is likely to be a platform for future generations of products. Alternatively, a development manager would find it useful to know the likely trajectory of the product (and similar products) in the industry in order to determine how much effort to devote to additional features versus early delivery. These categorizing schemes for innovations and developments are useful PD tools if they are accurate.

2.3.3 Creating and capturing value

Creating value is another fundamental component of innovation management and PD. Value is measured in a product's ability to address customer needs. Consumers are willing to pay more for products that are of good quality, save them time and money, or provide more features. (Soho, 1993; Griffin & Hauser, 1993; Ulrich & Eppinger, 2000)

---

* There are several other variations on these themes. Other systems map development projects into the following categories:

- Niche creation, Architectural, Regular, and Revolutionary. (Abernathy & Clark, 1985)
- Research and development, Breakthrough, Platform, Derivative, and Alliance/Partner (Wheelwright & Clark, 1992)

Products can have many benefits; each benefit provided by a product fills a customer need, and thus has worth. Developing a product that increases benefits to customers increases the worth and thus creates value.

Methods for creating value vary widely by product and industry. The vice president of R&D at ABB uses the motto "creating value through new technology," yet advanced technology is only one means of adding value. (Bayegan, 2002) Tool companies may add value by improving the industrial design of their screwdrivers to fit more comfortably in peoples' hands. Light bulb manufacturers may add value by increasing the shock resistance or longevity of their bulbs. Software developers may add value by eliminating bugs or expanding an application feature. A dictionary website developer may add value by allowing a viewer to avoid the cost of purchasing a physical dictionary.

To be successful, companies must also capture some of the value created by a new innovation. Far from the aphorism about the world beating a path to the door of a successful innovator/mousetrap-maker, the innovators themselves must make great efforts to reap the rewards of successful product introduction. Otherwise, consumers or competitors may gain the benefits instead. There are several corporate examples of successful and unsuccessful efforts to capture the value of innovation and product development. For example, Xerox developed graphical user interfaces (GUIs) but then allowed rival companies to introduce the developments in computer products. Instead of Xerox gaining, companies such as Apple, IBM, and Microsoft were able reap the benefits of GUIs in the computer market. (Cringely, 1992) On the other hand, some companies have displayed remarkable success in locking in value from their own product development. Searle succeeded in developing and marketing Nutrasweet exclusively, while Microsoft was able to set an operating system standard and profit handsomely from its dominant position. (Cusumano & Selby, 1995) The reasons for these disparate results stem from the varying abilities of companies to capture the value of innovation and development.

There are several tools for capturing the value of innovation and product development. Patents are one key to appropriability, however they are insecure because they are not always granted and because competitors can frequently circumvent them. Companies may instead guard themselves by maintaining trade secrets, but this tends to work better for processes rather than products, since processes are internal and products are usually available for all to see. Companies can also appropriate the value of new products by controlling complementary assets. These complementary assets can include manufacturing facilities, industrial partnerships, marketing contacts, internal technical knowledge, or a variety of other factors whose absence would serve as a barrier to entry to other firms. (Teece, 1987) Finally, companies may consider standards as part of their plan to profit from innovation. It generally behooves a company to control, set, or be a standard in an industry, whether that standard is an operating system in computers, a paper size for printers, a format for audio-visual recording, a mechanical alignment for turbines, or a keyboard arrangement for writing equipment. Once such standards are set, it becomes more difficult to force a change, introduce an improved design, or gain market share as an outsider. (Cusumano and Selby, 1995; David, 1995) Companies, as part of their innovation and product development, may aim to set such a dominant design or create an industrial platform as part of an effort to capture a market.

Thus, innovation management may include organizational capacity, mapping innovations, and creating and capturing value. Companies' innovation management decisions set the stage for PD and determine which risks will have to be managed by PDPs.

2.4 Development risks

"Nothing in the universe can be the same if somewhere, we do not know where, a sheep that we never saw has – yes or no? – eaten a rose."
– Antoine de Saint-Exupéry in The Little Prince

The upstream drivers of the previous sections set the stage for product development by establishing a context for company actions and risks. Risk is defined as exposure to danger or loss. The word's roots, the Latin and Spanish sailors' terms for "steep rock"

and "to go against the rocks," suggest that risk has long been considered ominously. Although rarely considered pleasant in business, taking risks is frequently necessary for success. Balancing risks and potential rewards is one of the most enduring themes of economic and developmental decisionmaking. (Foster, 2001; MacCrimmon & Wehrung; 1986; Nichols, 1994; Ansell, 1992)

Development risk is closely tied to uncertainty. Uncertainty is a prerequisite to risk, and is frequently viewed as a natural background or context over which action may be taken. (Ben-Haim, 2001) Risk usually applies to the critical action rather than the context, tends to be more quantifiable, and is something to which one exposes oneself. For example, when a sailor goes out to sea, the possibility of an ocean storm is an uncertainty, but the resultant risk is that the sailor's boat will sink in that storm. If the sailor chooses not to sail, the uncertainty of the storm remains, but the potential impact, and thus the risk, disappears. In this case, uncertainty is a natural phenomenon but risk is the function of exposure to that uncertainty. (Young & Tippins, 2001)

In the field of product development, levels of uncertainty are so variable that uncertainty and risk are frequently used interchangeably. Common parlance also frequently merges the terms "risk" and "uncertainty" because of changing circumstances. For the purposes of this research, uncertainties are the "unknowns" which are usually the context for PD action and beyond the control of developers. Risks, on the other hand, are the resultant ambiguities – usually multiplicative products of uncertainties and impacts – that companies can quantify and reduce to simple probabilities or potential costs. Although risk is a matter of choice rather than fate, the choice to expose oneself to risk is inherent with beginning a PD program. (Bernstein, 1996) Thus, risk and uncertainty remain distinct concepts only when uncertainty percentages and impact values are specifically named.

Many PD uncertainties lead to different risks of development failure: a slow or late product may miss a market opportunity and incur too many development costs; a technically challenging product might be impossible to design, may lack the expected

features, or be of poor quality; and a product with misguided specifications may not fulfill customer needs and completely miss a market niche.

Existing literature suggests several ways of categorizing PD risks and uncertainties. Some sources simply list every possible risk, ranging from schedule slip to staff turnover. (Beck, 2000) Others categorize risk in particularly overlapping and interdependent terms, such as market, market introduction, technical, manufacturing, and managerial risk. (McGrath, 2001) This research uses a traditional categorization of risk by source. A successful PD process should be able to manage or mitigate the following four major types of risk:

Technical – Risk regarding whether a new product is technologically feasible and will perform as expected. This has been colloquially termed "difficulty building to a specification" because the design specifications are clear and valid but difficult to achieve.

Schedule – Risk regarding whether a new product can be developed in the time allowed or available.

Budget – Risk regarding whether a new product can be developed with the financial resources available.

Market – Risk regarding whether a new product accurately addresses changing customer needs and product positioning with respect to dynamic competition. Unlike difficulty building "to a specification," this concern arises when an achievable specification ends up bringing the wrong product to market.

These four major risks are neither comprehensive nor entirely independent of each other. Many other factors may also present uncertainty, but they can be subsumed by the larger risks detailed above. For example, quality assurance or integration risk may sometimes be subsets of technical risk. The risks are also occasionally interdependent and overlapping. For example, "scope creep," a common problem involving feature addition during development, frequently occurs in an attempt to address market risk, but it may

increase technical, schedule, and budget risks. In another example, technical uncertainty may give rise to a lag in schedule. Alternatively, market or technical uncertainty may influence the budget by requiring additional prototypes. It is therefore impossible to completely separate the types of risks faced in PD, although the categorizations are useful in planning and assessing PDPs.

Given these four categorizations, companies can identify risk or uncertainty profiles for particular projects. For each of the four types of risk, we can rate the level of uncertainty, ranging from slight coordination issues to major unknowns (De Meyer, 2002) A sample risk profile might appear as in Figure 2.4. Risk profiles allow for a reasonable comparison among different companies' development risks. Those risks are exactly what the PDPs in the following menu attempt to manage.

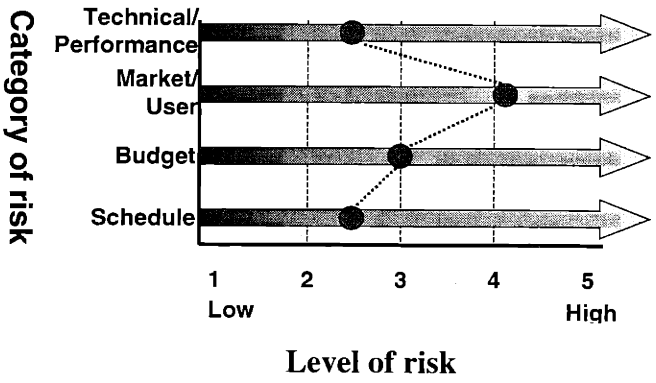


Fig. 2.4: A sample risk profile

One of the major tools for risk management includes product integration, often in the form of early prototypes, tests, simulations, or models. An integration is usually part of a planned iteration and provides information or feedback for improved design. As later sections show, the cost, time and fidelity or quality of these integrations vary widely across industries. For example, some prototypes are difficult or expensive to build because they require dyes, machining, or complex mechanical integrations. Other integrations may be easier if they are computer-based, but do not provide as much quality information or feedback if the computer models do not capture all the real-world aspects of future products. Companies must weigh the benefits and costs of integrations to ensure that they reduce more risks than they create; early integrations or prototypes are

not always practical or possible. However, information gained from integrations, tests, and feedback often improves companies' PD efforts.

2.5 A contextual case study – industrial combustion turbine PD

This section describes a case study of comparative combustion turbine development at Siemens-Westinghouse Power Generation (SWPG), General Electric (GE) Power Systems and ABB. It demonstrates important precursors to PD, including the drivers of innovation and innovation management that shaped the development risks and processes that companies use today. In doing so, it sets the stage for a further case studies (see sections 6.1 and 6.6) by providing a retrospective analysis of a power generation success story.

This case study is divided into three sections. First, a brief introduction provides the context of the combustion turbine story, explaining what the product is and why it is important. Combustion turbines are complex products that have revolutionized the power generation sector and society in general. Second, the technical drivers and enablers are explained. As families of commercial product platforms emerged from their military origins, improvements were not measured in terms of added features, but rather in terms of increasing efficiency. These improvements were achieved through a combination of material and cooling innovations, and were heavily influenced by government actions. Finally, the industry's innovation management – the step immediately preceding PD – is reviewed. It is found that S-curves are only a conditionally useful tool for understanding PD risks and that standard-setting and organizational capacity help define PD challenges.

2.5.1 Case study introduction

Industrial combustion turbines are part of a power generation success story. These relatively new and environmentally-friendly turbines are in high demand for electric generation and are experiencing a surge of market growth as they supplant coal-fired power plants. These turbines usually run on natural gas (they are sometimes called gas turbines) and are among the most important pieces of hardware in the $220 billion US power industry.

Combustion turbines hold a dominant position in new orders from turbine customers, who are generally either utility companies or independent power producers. Fig. 2.5.1 demonstrates how combustion and combined cycle turbines dominate over two-thirds of the over 21.8 gigawatts of recent US electric capacity purchases. They are rapidly penetrating the established base of coal/steam electricity generation. This section examines the path to this dominance. Although originally developed in the 1940s as military spinoffs of jet engines, they did not emerge in force until massive blackouts and shortages during the 1960s led to the demand for the provision of peak power.

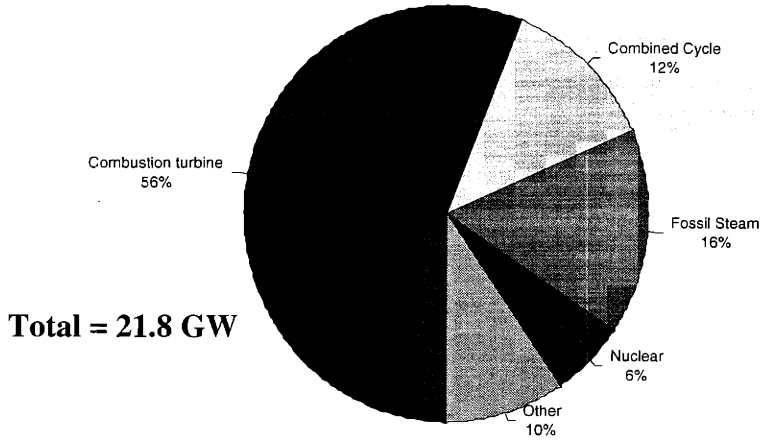


Figure 2.5.1: Combustion and combined cycle turbines dominated US capacity additions for 1996-2000 (US DOE, 1995 & 2000)

2.5.2 Factors in technical development

Combustion turbines for power generation have improved greatly from their early predecessors, as can be seen in their rising efficiencies over time in Figure 2.5.2. A doubling in efficiency has occurred for simple cycles, with the introduction of combined cycles causing a tripling in efficiency. Turbine efficiencies, along with cost and reliability, are among the most important criteria considered when power producers (customers) place orders for new plants. Therefore, the development gains in efficiency are crucial for the success of combustion turbines.

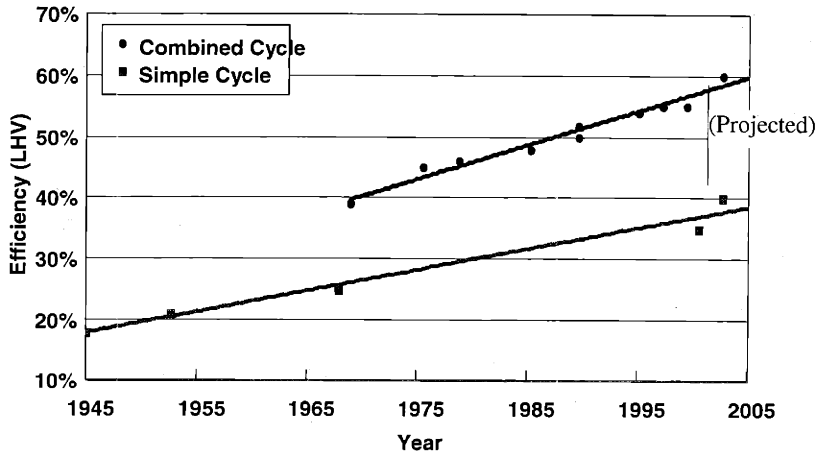


Figure 2.5.2: General efficiency increases for representative combustion turbines.

To increase efficiencies, turbine designers have worked to increase firing temperatures without damaging the turbines themselves. The advantage of high firing and rotor inlet temperatures (RITs) is that they nudge combustion turbine cycles closer to ideal Carnot thermodynamic cycles. However, firing turbines beyond the threshold temperatures of their components threatens their integrity and reliability. R&D addressing this concern has progressed along two major avenues of development: material and cooling innovations.

Material innovations

Materials used in combustion turbines have gone through many incremental improvements since the first practical turbines were developed in the 1940s. Most R&D efforts led to improved steel alloys for use in turbine vanes, blades, and inlet blocks. This R&D in turbine materials and coatings led to two important effects. First, combustion turbines were better able to withstand high temperatures. These more rugged materials allowed for hotter inlet gas to enter the turbine's first stage blades, leading to higher efficiencies. Second, material improvements led to an increase in rotor life and reliability. Gas and combined cycle plants could not have achieved popularity and larger market shares without solving problems such as premature blade cracking or component deformations. Together, the higher temperatures, higher efficiencies, and improved

reliabilities have advanced the deployment of combustion turbines in the power generation market.

Progress in combustion turbine material development often came in the form of alternative stainless steel or metal alloys that had improved heat characteristics. Different parts of combustion turbines use a variety of alloy metals, including varying quantities of cobalt, nickel, and chromium. For example, some early stationary turbine blade designs used welded structures in AISI 310, a 25-20 austenitic stainless steel that had excellent resistance to both corrosion and to oxidation at elevated temperatures, but had limited strength capabilities. Some turbojets then switched to higher strength, nickel-based alloys, but this proved unsuitable for industrial combustion turbines because they either lacked corrosion and oxidation resistance or they were too difficult to weld with enough integrity. (Bannister, 1996) In the 1960s, engineers began to design these vanes with cobalt alloys for two reasons. First, cobalt alloys have high heat tolerances and can withstand high firing temperatures and corrosion with less cracking or warping. Second, cobalt alloys tend to have favorable welding characteristics. The welding ease of this metal can be extremely important in repairing turbine vanes that crack with time and use. Thus, material improvements in blades and vanes have improved heat characteristics and increased rotor life by reducing turbine damage and allowing easier maintenance. Improved cobalt alloys are still used today to increase creep and oxidation resistance. Some turbine manufacturers have also increased their use of titanium, a stronger but expensive metal, in their combustion turbine components.[3]

Future combustion turbines may be able to make better use of ceramic materials. The introduction of heat and corrosion resistant ceramics may become possible if their brittleness can be circumvented. Advocates of ceramics hope that these advanced materials can be the next big breakthrough in combustion turbines, succeeding another major breakthrough that occurred in the 1960s in the area of turbine cooling.

[3] From interview with Lee McLurin, Combustion Turbine Development Manager, Siemens-Westinghouse Power Generation.

Cooling innovations

The introduction of cooling to combustion turbines was the most important technological breakthrough in combustion turbine development since the end of World War II. Improvements in turbine cooling also helped to advance the penetration of combustion turbines in today's power generation market. Like material advancements, cooling innovations allowed power producers to feed higher-temperature inlet gases into the turbine blade-path. Combustion turbine operation at these higher temperatures leads to higher efficiencies and makes these turbines more viable sources of electric power.

Cooling helps to increase rotor inlet temperatures by circulating air or steam through hot turbine components. This includes extremely intricate pathways, tunnels, and holes to allow for maximum heat transfer to the cooling fluid. Without this critical innovation, turbine designers would have been limited to the ordinary heat tolerances of metal alloys and coatings. As Figure 2.5.3 demonstrates, firing temperatures would have leveled off after the 1960s at their threshold levels of about 1800°F (1000°C). Instead, rotor inlet temperatures were able to increase and improve turbine efficiencies while minimizing thermal damage. These trends helped combustion turbines break into the power generation market.

Cooling was originally introduced into military turbojet engines in the early 1960s. The advance followed a relatively common trend in combustion turbine technology transfer: new developments in the turbines of military turbojets would become available to civilian aircraft about two or three years later, followed by diffusion to the power generation combustion turbine industry after about five years. (Hammond, 1973) Cooling techniques became more intricate and effective with the advance of computer codes and modeling. An example of the evolution of cooling technology can be seen in Figure 2.5.4, which shows how Westinghouse has improved its 1st row combustion turbine blade through the years.

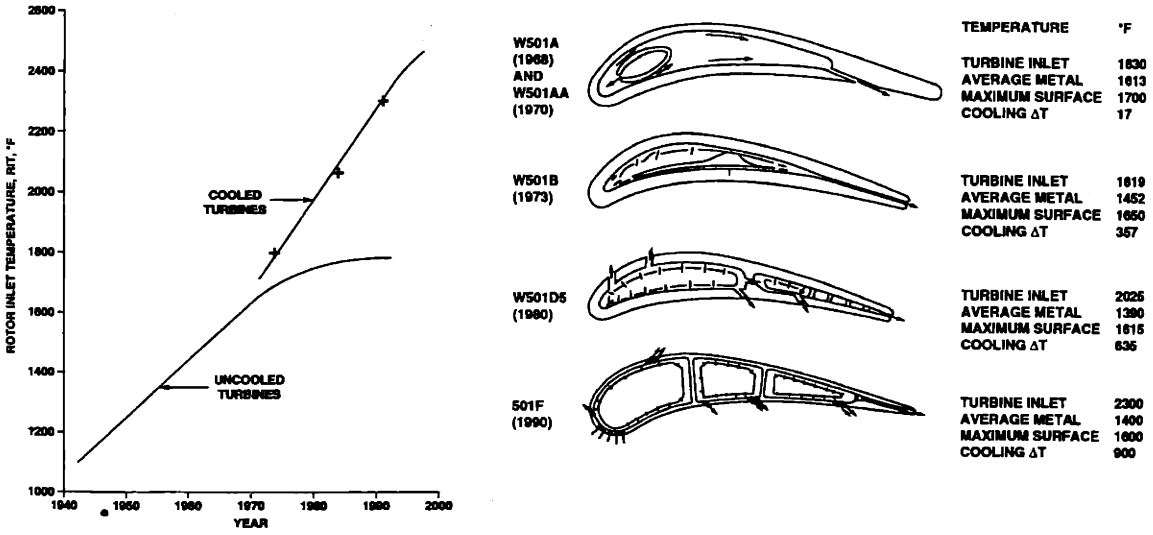


Fig. 2.5.3 (left): The introduction of cooling led to a breakthrough in inlet temperatures
Fig. 2.5.4. (right): The evolution and complexity of component cooling. (Bannister, 1996)

The role of government in innovation

In analyzing the technological development of combustion turbines, it is important to note how government involvement has contributed to R&D. The clearest involvement took place in the beginning of the combustion turbine era, when defense programs poured money into basic turbojet research. Major turbine manufacturers such as GE and Westinghouse got their start in the combustion turbine industry by then developing combustion turbines for applications in military aviation. As noted above, these turbojets became the first crude models for power generation combustion turbines and much of their technologies were transferred to industrial turbine use.

Government involvement through aviation R&D continues to this day. The US government alone has spent over $13 billion since 1940 on military jet engine development and still spends about $400 million per year on R&D efforts through companies such as GE and Pratt & Whitney. As a result of these efforts, there is a "supermarket of technology" created by the jet engine programs that is slowly utilized by power equipment companies. As noted above, this supermarket contains everything from new alloys that can withstand higher temperatures to computer codes for advanced turbine blade profiles and cooling. (Watson, 1996)

Governments have also devoted some attention to the power generation uses for combustion turbines. The 1970s saw the US Department of Energy (DOE) sponsorship of the High Temperature Turbine Technology program and the Japanese Government sponsorship of its national Moonlight project with Mitsubishi to improve turbines. A more recent initiative is the US Advanced Turbine Systems (ATS) program, which is spearheaded by Federal Energy Technology Center that includes six US turbine manufacturers, 83 universities and multiple DOE research centers. The goal is to subsidize and coordinate R&D that will lead to the next generation of efficient gas and combined cycle turbines.

In summary, technological innovation has played an important role in the advancement of combustion and combined cycle turbines. Material improvements and cooling advances have helped to increase RITs and efficiencies. Throughout this process, government purchases and sponsored R&D have spurred innovation alongside market forces.

2.5.3 Innovation management and development risks in combustion turbines

This analysis examines several aspects of innovation and development strategy, including

1) Mapping innovation: S-curve modeling of models turbine innovation
2) Creating and capturing value: Standards
3) Developmental organization

It concludes by observing how companies in this industry followed the technology strategy chain to arrive at today's PD risks and environment.

Mapping Innovation: The triumphs and pitfalls of S-curve analysis

The traditional technology S-curve provides some insights into the emergence of combustion turbine technologies, but is severely limited as an analytical product development tool. A modified S-curve is a useful descriptive tool to view the development of combustion turbine technology in retrospect; however, it is difficult to utilize the curve as a forecasting mechanism. Two charts display how S-curves can be useful but still problematic in mapping innovation and planning PD.

A capacity diagram of combustion turbine generation in the US can be seen in Fig. 2.5.5. It can be viewed either as a combination of two S-curves, with the latter beginning where the first ends, or as a single S-curve with a flattening abnormality in the middle. For reasons discussed in detail later, we assume for now that it is one curve with an abnormality imposed by external forces.

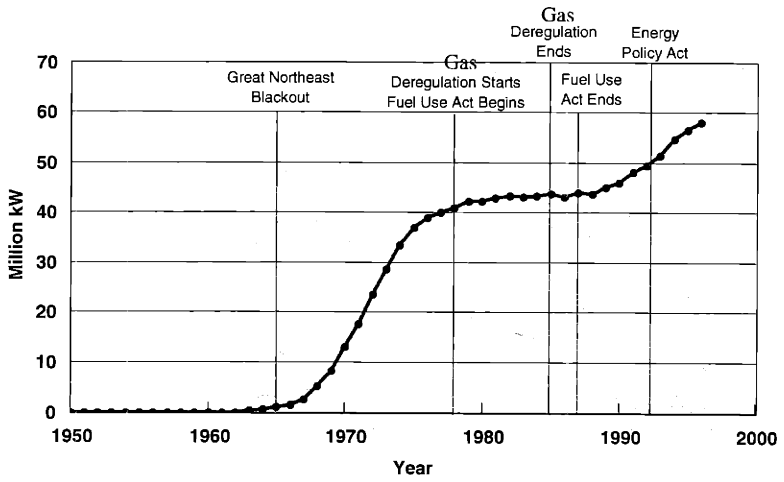


Fig. 2.5.5: The expansion of electric utility combustion turbine capability (US DOE, 1998).

Fig. 2.5.5 shows how combustion turbines gained entry into the power generation market, replacing traditional steam turbines along the way. The process began in the mid 1960s after a major blackout, after which combustion turbines served a niche market for peak or emergency power generation because of purely technical reasons: combustion turbines start up faster than steam turbines by avoiding the need to start a large boiler for steam production. The versatility and speed of combustion turbines made them popular, which in turn led design engineers and manufacturers to seek improved designs for the relatively new products. Efficiencies improved with additional R&D, spurring additional sales until an energy crisis and major changes occurred in the US fuel system. These exterior forces essentially shut down the combustion turbine industry for a decade. Later, policy decisions would eventually re-create the market for combustion turbines and once again promote new development and innovations. These external forces are discussed subsequently in this section, but the focus here is to show how the adoption of

combustion turbines can indeed resemble the S-curve, with the number of adopters growing dramatically before leveling off.

A different S-curve can be seen in Figure 2.5.6, which shows how operating temperatures increased with time for one manufacturer's specific family of turbines, in this case the Westinghouse 501 series. This figure shows how temperatures increased with the use of cooling in the early 1970s, stalled during a period of slow development during the 1980s, then increased when demand surged and development resumed using advanced computer codes in the 1990s. The curve does display S-type characteristics, but if effort replaced time as the unit of measure, the plateau and inflection points would disappear, we would appear to be in the middle of the slope, and innovations and improvements would not yet appear to be leveling off. This displays the sensitivity of this type of analysis to varying assumptions.

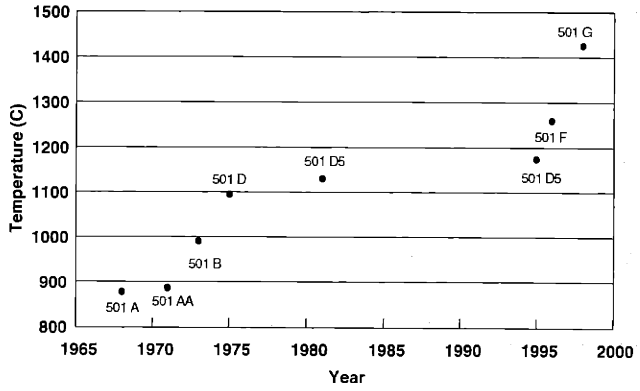


Fig. 2.5.6: Operating Temperatures of the Westinghouse 501 Combustion Turbine Family (Bannister, 1996)

As these examples demonstrate, the S-curve can act as a rough descriptive tool for managers mapping innovation, although different curves could lead to differing conclusions on where a technology stands on the S-curve.

Several other factors complicate the task of mapping innovations accurately. For example, three areas of changing government policy severely altered the S-curves above. The dates of some of their highlights can be seen on Fig. 2.5.5. First, fuel polices alternately deregulated and restricted natural gas, retarding gas turbine sales,

development, and innovation. Second, environmental regulations complicated S-curves by heaping new costs (of externalities) on coal-fired power plants and suddenly promoting new demand for cleaner combustion turbines. Finally, electric industry restructuring, marked by the Energy Policy Act, also skewed the S-curves by spurring renewed interest in and sales of combustion turbines in the 1990s. This is because combustion turbines are smaller and more modular than traditional steam turbines, thus representing smaller capital investments and quicker rates of return in a power industry that suddenly must care about its profitability in a new, competitive age.

In conclusion, S-curve analysis and innovation mapping is difficult in the power industry. As a descriptive tool, the S-curve adequately characterizes historical and technical data. However, as a prescriptive tool, different metrics and other "exterior" or policy factors skew the curves enough to hide the points of inflection. Understanding those inflection points is critical to managers' abilities to map innovations successfully, understand likely risks, and make appropriate PD decisions.

Creating and capturing value: Standards

Combustion turbines are somewhat standardized, but still vary widely by manufacturer and model. Despite some profound technical differences, major similarities exist which could identify a sort of dominant design. The dominant design is a series of burners in an annular ring around the turbine itself. American firms were able to set the dominant design standard because they had benefited from government-sponsored military research funding. GE and Westinghouse turbines therefore contained aero-derived concepts such as the "combustor-ringed" turbine. (Watson, 1996) In contrast, European manufacturers initially followed a losing route. Because they had not benefited as much from jet-related subsidized research, they viewed combustion turbines as a modification of older power generation methods that used large burners and boilers rather than series of small combustors. These companies failed to recognize a break in the S-curve. Rather than considering radical new annular designs like their American counterparts, they merely put forth designs that were much closer to traditional generation technology. Their initial turbines and combustors were essentially bolted together, with only one or two large

combustors resembling a conventional steam boiler. This branch of combustion turbine evolution withered and died, leaving the American companies to set the dominant design standard and earn consumer trust. The American methods were later adopted by other companies.

After the initial "standard" of many combustors was established, the turbine industry was able to grow independently. Standardization is not critical to the turbine industry as a whole. The only attribute which all turbines must share is the ability to turn generators at an established, synchronized speed, but this 60-hertz[4] power standard became established long before the advent of combustion turbines. Despite the lack of strong industry-wide turbine standards, individual companies try to follow their own construction and operating standards for purposes of modularity, reduced cost, ease of manufacturing, and sales.

Developmental organization

Different companies in the power turbine manufacturing industry, including GE, Westinghouse/SWPG and ABB, choose varying methods for managing and organizing their innovative workforce.

Market leader GE clearly views functional engineering organization as important for success, as can be seen in its statement:

> For us, the development of combustion turbine technology has been evolutionary, and based on teamwork. We've incorporated technology advances from our own Power Generation technology development programs, our own GE aircraft engine business, and our Corporate Research and Development Center.[5]

This demonstrates how, as part of innovation management, GE aligns its efforts by function as opposed to by product. In this case, GE combines its aircraft and power generation turbine efforts because of similar functionality, defying a product "silo." This organizational method has pitfalls as well as advantages. In the mid 1990s, GE retired many of its turbine designers at its power generation headquarters in Schenectedy, New

[4] 50 Hz in Europe, Asia, and Africa

[5] GE Website: http://www.ge.com/powergeneration/pg6.htm, 1999.

York. GE intended to "fill the gap" by transferring expertise from its aircraft engine headquarters in Cincinnati. The imperfect match resulted in a loss of organizational and institutional design knowledge, leading to a series of design difficulties and a less successful turbine model year.[6] Figure 2.5.7 demonstrates the problem. This figure is a fine example of the "swimsuit" effect: what it reveals is interesting, but what it obscures is even more interesting. The figure physically shows technical developments in successive "families" (platforms) of turbines by demonstrating how their efficiencies increase. However, the figure does not show a "G" family; it skips directly from F to H. The "G" family is subtly missing because it was an example of a product development failure. This figure shows how GE's attempt to reorganize interfered with its previous pattern of successful product development.

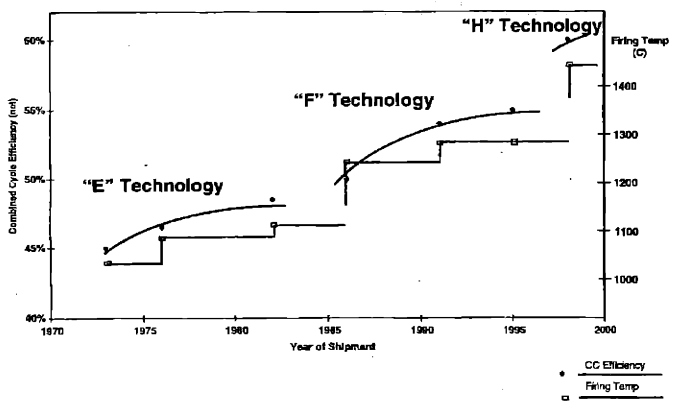


Fig. 2.5.7: GE Power Systems chart of technical product advances skips the "G" family harmed by engineering reorganization[7]

Westinghouse also tried to transfer knowledge between its aircraft engine and turbine businesses, but was less successful. Like GE, it formed power generation design groups independent of their aircraft engine designers in the 1950s. However, unlike its main rival, Westinghouse was not able to take advantage of aero-engine knowledge because it exited the jet engine business in 1960 while keeping its stationary combustion turbine division. The company has found success in organizing most of its development teams by product, and it is fostering stronger links between design and manufacturing groups.

[6] Information from an interview with Mr. L. McLurin, Combustion Turbine Development Manager , Siemens-Westinghouse Power Generation.

[7] Chart provided by Dr. J. Haynes, GE Combustion Research

ABB had a different engineering development and organization strategy. Brown Boveri closed its own combustion turbine business in the mid-1980s because of the aforementioned slump in the combustion turbine market (see Fig. 2.5.5). When it merged with ASEA in 1988 to become ABB, it gained some aircraft engine expertise, but this advantage was negligible relative to the developmental losses it incurred while withdrawn from the business for several years. As the combustion turbine slump ended and the market surged, ABB realized that it was at a competitive and technological disadvantage compared to Westinghouse and GE, so it embarked on a brand new development program. The company hired key engineering and management personnel who had previously worked for GE and integrated them into its existing development team in Zurich. It then isolated the design group and even gave the group a separate facility outside the city so that it could design a "fresh" turbine, unencumbered by past designs and traditional biases. This was a successful attempt to force a "leap" in technology by discouraging the evolutionary development that would have come with steady contact between design engineers and the manufacturing division. The result was its novel sequential combustor that was unusual and successful enough for ABB to differentiate itself and earn customers in the power turbine market.[8]

It is difficult to discern which of the above examples of managing innovation is best. All of the examples of the above companies have been successful; reasons for their organizational differences could be due to a variety of external factors, local or cultural concerns, or managerial choice. A next logical step would be to try to find larger examples of corporate engineering failure in order to see why certain (presumably other) engineering management choices were unsuccessful.

Development risks

In all cases, companies were faced with a business environment shaped by technological and market history. Given the maturity of both the gas turbine and electricity generation industry, most competition is based on technical factors: usually a combination of turbine efficiency and reliability. High efficiencies allow generation companies to reduce their

[8] Information from an interview with Dr. C. Tedmon, Former Director of ABB R&D.

operation costs by requiring less fuel per GW of electricity generated, while simultaneously offering environmental incentives by burning less fuel. High reliability allows generation companies to avoid the costly repairs and lost generation time resulting from blade and vane cracking or misalignment. Thus, companies are faced with significant technical risk in developing turbomachinery for sale to generators. Also, market risk is muted in the industry because of the common practice of offering contracts with guaranteed payments, or "liquidated damages," from manufacturers if a turbine does not reach a promised efficiency once installed. As discussed later in section 6.1, these liquidated damages translate market risk into technical risk, increasing technical risk by increasing the impacts (potential costs) of technical uncertainties.

2.5.4 Case study conclusions

This case study serves two purposes. First, it provides an example of the technology strategy chain issues presented in this chapter. Understanding the chain, including the drivers of innovation and innovation management, are critical to understanding the risks that companies face when they begin PD. The case study also provides background to the PDP case studies in Sections 6.1 and 6.6.1, which continue where this story ends.

Government and market forces were both drivers of combustion turbine innovations in both material and cooling technologies. Techniques for managing innovation, including S-curve mapping, new company organizational structures, and value capturing through standardization, all had varying degrees of success and failure, and together with policy forces shaped the development risks that companies face today. After half a century of turbine innovation, the combustion turbine market is relatively mature, leading developers to compete based on turbine price and efficiency.

2.6 From innovation to product development

The upstream innovation management decisions described and illustrated in the previous sections set the stage for product development and determine which risks will have to be managed by product development processes. Product development is where innovation management meets market reality. Successful PD that addresses development risks can

increase sales, reduce time to market, create better products and reduce expenses, but poor PD can be harmful to both a product line and the company that manufactures it. The key to this high-stakes game is successful PD process selection, yet companies face a daunting array of choices.

2.6.1 PD Literature review

PD literature may be broadly divided into several categories. Sources include general guides, discussions of architectural or modular PD, descriptions of PD tools, works on risk analysis and reduction, advocacy for specific PDPs, case studies, and limited comparisons among PDPs.

Some sources describe engineering design and PD in a broad and systematic way as a guide for improved implementation. (Cleland, 1994; Otto & Wood 2001; Pahl & Beitz, 1996; Rosenau & Moran, 1993; Ulrich & Eppinger, 2000) They give examples of many product development experiences and advocate use of a structured, multi-stage PDP to manage the competing technical or market risks.

Many authors already presented in this chapter write about technology strategy issues upstream of most PD. Others, such as Hax & Wilde (2001), Craig (2001), and Crawley (2001), combine PD with either architectural or corporate strategy. These sources explore the importance of the link between company direction, product modularity, and system architecture.

A considerable body of work has developed improved PD tools. Work stemming from the MIT Center for Innovation in Product Development extended the design structure matrix (DSM) as a useful tool for defining interactions among product parts, groups, or development organizations. (Dong, 2002; Eppinger & Whitney, 1990, Yassine & Whitney, 2000, Eppinger, 2001; Helo, 2001) The DSM tool helps identify and organize tasks and feedback loops in complex development programs.

Other applicable research explores the roles, categorizations, and management of risk. De Meyer, et. al. (2002), Hartman & Myers (2001), and Jootar (2002) organize risk by type and warn of the need to observe these risks carefully in order to improve project and development management. More general risk literature stresses the importance of maximizing expected values (Ansell & Warton, 1992), hedging and parallel development. (de Neufville, 1990)

Some authors are champions of certain PDPs. Cooper (2001) argues persuasively for the effectiveness of the stage gate PDP. Other sources, including those general sources mentioned in the beginning of this section, (Pahl & Beitz, 1996; Ulrich & Eppinger, 2000) take more tacit approaches but implicitly endorse this point of view. Boehm (1994) advocates the use of the spiral process in software development. He is joined by Hekmatpour & Ince (1988) and Gilb (1988), who similarly denounce the deficiencies of rigid waterfall processes in favor of flexible prototyping. Beck (2000) explains and makes the case for extreme programming. Smith & Reinertsen (1992) urge stepping away from phased development in favor of increased flexibility, and do not limit their recommendations to only the software industry. Many of these conflicting views are further presented and compared in Chapter 3.

Product development literature is strengthened by many studies of individual companies' PD efforts. Cusumano & Selby (1995) and MacCormack (2000A) closely examined the Microsoft Corporation's process of frequent "builds." Several authors investigated automobile manufacturers processes in both the US and Japan. (Cusumano, 1991; Ward,1995; Womack, 1991) Most made comparisons between management methods on different sides of the Pacific.

Finally, some sources begin to compare different PDPs. Krubasik (1998) argues for the need to customize PD, suggesting that "product development is not monochromatic...not all product development is alike. Each situation has a different context...[implying] different managerial actions." Other authors, such as McConnell (1996) offer brief and balanced comparisons of different PDPs, but limit the scope to theoretical examples.

MacCormack (2000B & 2000C) suggests a method of matching PDPs and context, supporting his conclusions with a comparative empirical study, but is stymied by the challenge of effectively measuring process success.

2.6.2 Common product development actions

Most product development efforts include a series of common actions, steps or stages. Enacting these tasks in an organized manner constitutes a product development process. These steps, according to prevailing literature, include:[9]

- **Product planning** – This introductory step includes market surveying, concept selection, and concept design. Rough budgeting may also begin in this step.
- **Process design and selection** – Companies must then design or choose how they will proceed with the PD process. This involves choosing from an array or menu of PD processes with different types of schedules, patterns, feedback loops, and likely results. This step frequently is the result of momentum because companies have set policies or procedures which must be followed regardless of circumstances.
- **Specification creation** – Having defined customer needs, this step involves the initial setting of product specifications. Depending on the process selection, these specifications may be either flexible or frozen.
- **System-level design** – Upper-level design of overall product and system architecture without great attention spent on individual modules or features

---

[9] The names of these steps are adapted from and consistent with several sources and are relatively common and accepted. (Ulrich & Eppinger, 2000) The exception is "process selection," which is the exclusive domain of this research and, as the thesis will demonstrate, a necessary addition to effective PD. Other sources include variants of the same major steps, such as:

- (1) Preliminary investigation (2) Detailed investigation – Build business case (3) Development (4) Testing and validation (5) Product and market launch. (Cooper, 2001)
- (1) Requirements analysis (2) Requirements specification (3) Design (4) Implementation (5) Validation (6) Verification. (Hekmatpour & Ince, 1988)

or alternatively, in the specific case of software development:

- (1) Systems analysis (2) Design (3) Plan and budget (4) Build (5) Test (6) Run (7) Maintain. (Gilb, 1988)

Many of the basic steps are similar and, for the point of discussion here, virtually identical. As future chapters will demonstrate, the key differences are not so much in the *names* of the similar steps, but rather in the order, length, repetition of, reviews, and information exchanges between the steps.

- **Detailed design** – The "guts" of any development process, this step includes the mathematical/engineering design of mechanical components or the actual coding of software.
- **Testing/prototyping** – The validation and verification of both detailed and system-level design, this step goes beyond mere simulation to confirm that behavior of the product (or its components). Depending on the PD process used, this can be a penultimate step in which success is common or a regular part of a frequently-occurring cycle, in which case the information from both successful and unsuccessful tests are used as feedback for product refinement.
- **Release** – This includes marketing, production ramp-up, and the myriad of other issues (financing, maintenance contracts or guarantees, etc.) during which customers use the product. The company has no expectation of further testing or feedback regarding the product released. Any feedback from the product goes towards either future products/versions or towards fixes or maintenance in case of trouble or recalls.

These steps are not all-inclusive. Many companies, especially in the biotechnology and pharmaceutical industries, have additional steps and requirements regarding clinical tests and pretrial approvals. Other companies must consider field service and recycling or manufacturer take-back as part of their product development. However, most companies use at least some form of the above actions.

The purpose of PDPs that include these actions is to provide a structure for managing the many uncertainties and risks that companies face. Breaking up the process into smaller actions is one way of reducing risks. Most actions in the process involve some form of risk – such as system-level design decisions that may only possibly be technically feasible, or which may only possibly meet customer demands. Literature sources recognize PDPs as risk management structures, but they do not all acknowledge a multiplicity of processes. Those that do recognize multiple processes still face difficulty in comparing PDPs because they have no common language in which to compare them. Comparisons are frequently either partisan in view or industry, limited in scope, or based

on process steps, when it is in fact relations between these common steps that set many processes apart from each other. Steps and actions can be arranged in different ways: they can be extended or shortened, checked, repeated, skipped, reordered, or reorganized depending on reviews and information feedbacks within companies. These distinctions are more difficult to categorize and make PDP design and selection difficult. Companies frequently face the difficult challenge of designing or choosing a PDP that best addresses their risks; only some of them are successful.

2.6.3 Product development problem definition

Companies have difficulty designing or selecting PDPs because process differences are poorly understood. To overcome the gap in existing literature and industrial decisionmaking, this research has three goals. First it seeks to identify distinctly different PDPs and establish that variety exists. Second, the research demonstrates how PDPs can address different recognized classes of risks. To do so, it defines parameters that allow observers to distinguish risk management differences between processes. Finally, the research uses lessons from observation and comparison to propose a method for improved PDP design and selection based on risk. The overall research goal is to help academics and business managers with the difficult task of identifying, comparing and successfully designing or choosing PDPs for risk management.

2.7 Chapter summary

This chapter provided the context for understanding product development challenges. Past literature suggests that market forces, societal change, and government action drive the need for technological innovation. Companies manage that innovation by mobilizing their organizational capacities, understanding and mapping innovations in their technological contexts, creating value and then capturing some of that value in order to profit. These upstream actions lay the groundwork for PD work, determine the risks facing companies at the beginning of their product development processes, and influence the ability of companies to build prototypes or integrated models of their products. Literature in the field agrees on many phases of PD, but fails to identify and compare PDPs in a manner that allows companies to successfully design or choose PDPs for risk

management. This research attempts to fill the academic gap and improve PDP design and selection.

3. PRODUCT DEVELOPMENT PROCESSES

"Work once, work twice."
– Benjamin Franklin, Poor Richard's Almanac

Product development literature provides many examples of how companies manage development risks. This chapter presents an array of canonical PDPs and describes some of their basic tenets, advantages, and disadvantages. The stage gate process is the traditional and dominant PDP in American industry. The spiral process, a more flexible process that incorporates cross-phase iteration, is gaining popularity in the software industry. Evolutionary prototyping and delivery processes involve customers and incorporate aspects of both the stage gate and the spiral processes. The design-to-schedule and design-to budget processes place explicit limits on themselves. Although the PDPs are not explicitly compared to each other, the advantages and disadvantages of different processes illustrate important commonalities and differences in terms of how they manage risk. The literature-based descriptions of all these model processes serve as useful bases of reference for the actual PDP case studies presented later in Chapter 6.

3.1 The waterfall/stage gate process

"Every step of progress the world has made has been from scaffold to scaffold and from stake to stake"
– Wendell Phillips in a speech for women's rights, 1851

The most widely-used type of product development process, and the standard for comparison in this research, is the traditional stage gate process shown in Figure 3.1. (Cooper, 2001; McConnell, 1996; Smith & Reinertsen, 1992; Ulrich & Eppinger 2000) This process, also called waterfall, phase-gate, or life-cycle by various authors and practitioners, has been dominant in US industry for almost 30 years.

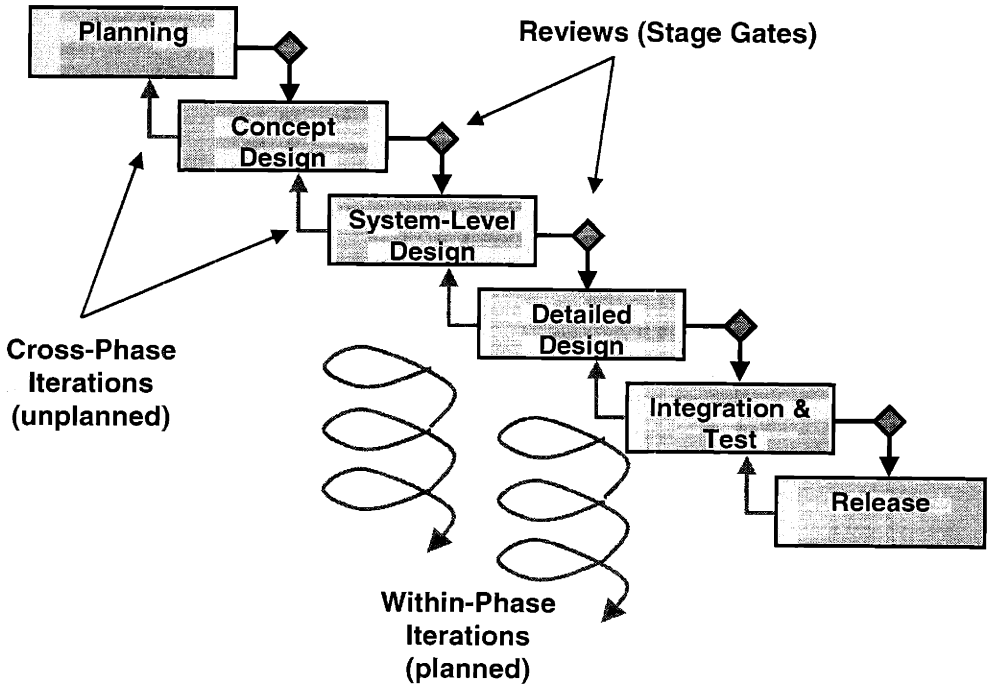


Figure 3.1: The traditional stage gate, or waterfall, product development process

This ideal waterfall process proceeds in discrete stages, or phases, from product planning to product release. The intermediate phases include concept design and specification analysis, system-level design, detailed design, and testing or prototyping. At the end of each phase is a stage gate, which consists of a phase review to evaluate whether the previous phase was successfully completed. If the project is reviewed positively, work proceeds to the next phase. If not, then work continues or iterates within that phase until it can successfully pass the hurdle.

The reverse arrows, or cross-phase iterations, in Figure 3.1 indicate that it is possible to reverse course and make changes in earlier phases, but such iterations are difficult and often costly. The purpose of the phase gates is to confirm that a phase is complete; going back to revisit a supposedly completed phase defeats that purpose, is usually not part of the original plan, and may indicate substantial rework. These major, and generally unexpected, feedback loops are accepted if necessary, but are difficult and generally confined to adjacent stages to minimize the expensive rework involved in feedback across many stages.

Stage gates and the difficulties of revisiting earlier phases lead to fixed outcomes at the end of each stage. Iterations occur within each stage, but are not planned across phases because cross-phase action would defeat the purpose of phase-gates, which exist to close one chapter of development and open the next. The resulting narrowness of iteration has both advantages and disadvantages.

One major advantage of waterfall processes is the structure that they impose on development by reaching sharp product definitions and specifications early in PD. Technical risk is reduced because narrow iterations and phase gates lead traditional waterfall processes to freeze specifications early. Rigid specifications help design teams by giving them clear goals towards which to work. The stable product definition also helps to avoid errors because midstream corrections are infrequent. Furthermore, the inherent clarity of the process allows early forecasting and minimal planning overhead.

The stage gate process performs well in cases when product cycles have stable product definitions and when the product uses well understood technologies (as in the case of upgrades or maintenance improvements to existing products.) In these cases, the waterfall process helps to find errors in the early stages of a project, when costs of changes are low. The waterfall process also works well for projects that are dominated by quality requirements rather than cost or schedule requirements. In these cases, where quality and error-avoidance are high priorities, the most attractive path is a direct one with early specifications and no subsequent changes that increase the likelihood of mistakes.

The main disadvantage of narrow iterations and stage gates is inflexibility. Because they do not cross phase boundaries, narrow iterations cannot incorporate feedback from later phases. This makes it difficult to fully specify requirements in the beginning of a project, especially in a dynamic market. Poor or misleading specifications can lead to great market risk. Failure may result if early specs and assumptions are proven wrong by subsequent market research, detailed design, or prototyping. The waterfall process does

not handle these midstream changes well and can be ill-suited for projects in which requirements are poorly understood in the beginning.

Stage gate processes are also sometimes poor matches for companies when speed and time-to-market are more important than extra functionality or total quality. Documentation of stage gates processes can be burdensome. In addition, traditional stage gate processes have difficulty incorporating cross-phase processes that do not fit neatly into individual process stages. The stage gate process also has difficulty handling parallel tasks within stages. As a result, the length of each stage may be associated with the slowest field within that stage, thus lengthening the development process. (Smith & Reinertsen, 1992)

3.1.1 Modified stage gates

If the basic stage gate process is too rigid and unchanging for some circumstances, then looser, modified stage gate processes may help PD efforts. This section examines two such modifications: stage gates with subprojects and overlapping stage gates

One of the problems with the classic waterfall is that progress can be retarded by one step of many within any given phase. For example, although there are many small steps in individual design, one component of the detailed design might take significantly longer than the others. Rather than letting this become a rate-determining step and thus delaying the entire process, the process can be improved as shown in Figure 3.3.

Figure 3.2 shows a waterfall with subprojects. If a system can be decomposed into logical and quasi-separable components, then it may make sense to do some of the work in parallel and let each subproject proceed at its own pace. This way, resources are not wasted by forcing each subproject to finish simultaneously when some may be completed earlier.

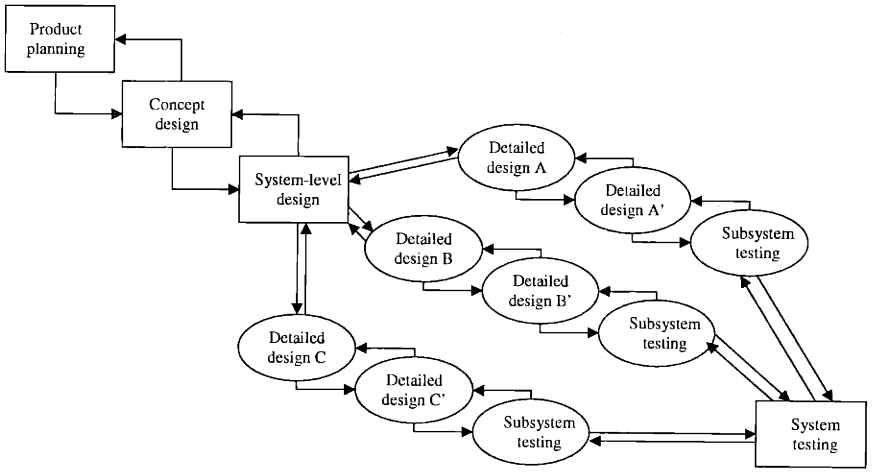


Figure 3.2: A stage gate process with subprojects

Another modification can be seen in Figure 3.3, which demonstrates the "overlap" waterfall process; in this process, phases intersect prior to the passage of stage gates. One of the problems with the classic waterfall process is the silo or "throw it over the wall" mentality associated with strict divisions between phases. The lack of continuity can lead to difficulty if different personnel are involved in each step, which remains a problem in some companies. The lack of continuity can also be problematic if an unforeseen difficulty becomes manifest one stage too late, forcing a potentially unfortunate or expensive reversal of course. (McConnell, 1996)

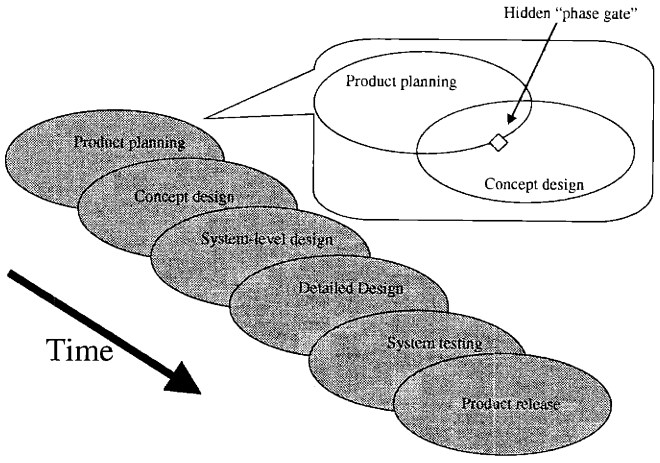


Figure 3.3: The overlapping waterfall product development process

The process in Figure 3.3 can help overcome some of these problems. By allowing stages to overlap, some knowledge – and perhaps feedback – from one stage can allow for more effective work on the next. Other benefits are improved teamwork and a more project- (rather than function-) oriented environment.

A problem common to both of the modified waterfalls is that they lead to parallel work. As noted above in Figure 3.3, parallel work may be fine if there are no interdependencies. However, if there are unforeseen interdependencies, a company with too many tasks in parallel risks technical failures (if the interdependencies are never resolved) or inefficiency (if the interdependencies lead to endless cycles of unplanned, cross-phase iterations). In addition, some milestones may be more ambiguous and difficult to track. (McConnell, 1996)

3.2 The spiral process

"The shortest distance between two points may be a 'great circle route.'"
– Repogle Globe Manual

The spiral PDP differs from the stage gate process because of its emphasis on comprehensive iteration. Unlike the stage gate process, it includes a series of planned iterations that span several phases of development. It is a relatively recent product development process that has been adopted by many in the software industry. Spiral process proponents assert that it reduces burdensome and expensive rework in software, thus lowering development time and cost. (Boehm, 1988, Gilb, 1988; McConnell, 1996)

The spiral PDP can lead to the development of a competitive product on schedule and within budget by managing risks early. Despite its circular form, it repeats regular steps, including concept development, system level design, detailed design, and integration and testing. The radial dimension in Figure 3.4 represents the remaining costs to be incurred in accomplishing the steps, while the angular dimension represents the progress made completing each cycle of the spiral. As a project spirals outwards, each loop brings it closer to completion, while each movement away from the center reflects additional cost.

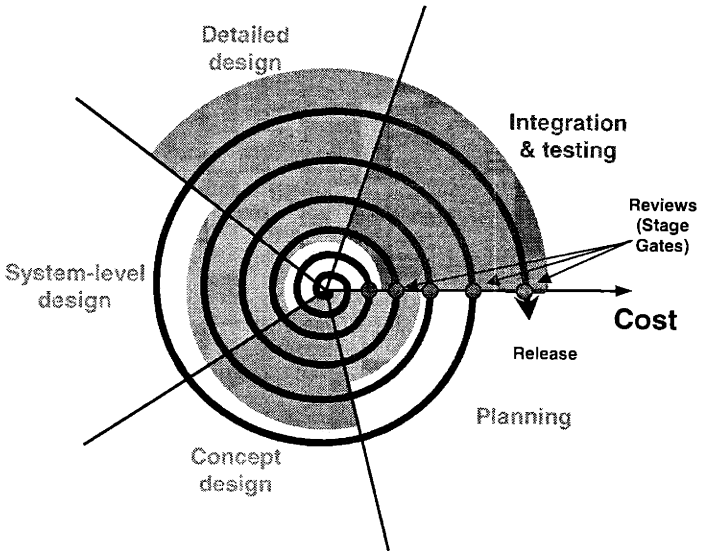


Figure 3.4: The spiral product development process

The spiral process requires managers to evaluate risk early in the project, when costs are still relatively low. "Risk" in this context entails all four major areas of risk described earlier, including poorly understood requirements and architecture, performance problems, market changes, and potential problems in developing specific technologies. These risks can all threaten a project, but the spiral process helps to screen them early, before major costs are incurred. The spiral process can be desirable in rapid product development because, as costs increase later in the project, the risks decrease. (Boehm, 1988)

A simple spiral process with minimal uncertainty and only one loop would closely resemble a stage gate process. However, most projects entail uncertainty; companies that evaluate and manage their risks with multiple cross-phase iterations choose a significantly different path. By going through many stages with the full expectation of returning to them later, the spiral process allows a brief glimpse into the future which is not allowed by the stage gate process. This glimpse yields information from later stages that can be incorporated in early concepts, requirement specifications, and architectures, thus reducing risk. The risk reduction comes at the cost of more flexible product specifications, but this flexibility can be advantageous in dynamic environments. In this

way, the spiral process overcomes difficulties presented by unclear initial product requirements, a challenge which is poorly handled by the classic waterfall process.

The spiral process has several disadvantages. First, it is more sophisticated and complex than other processes, and thus requires more management attention. Managers must define verifiable milestones to determine whether the project is ready for the next round or spiral; this shadows the phase gates that this process purports to avoid. Second, the lack of rigid specifications can potentially lead to delays in manufacturing long lead-time items. Third, the spiral process may appear to be overkill for simple projects since it could fold into a simpler waterfall process. Finally, Barry Boehm of TRW, the author and a major proponent of the spiral process, himself acknowledges difficulties in the first spiral step of determining objectives, alternatives, and constraints. Later scholarly work expands the spiral process by suggesting a split of this first step into several others. (Boehm & Bose, 1995)

A key distinguishing feature of the spiral process is the planned, large-scale nature of iterations. Risks are assessed in each iteration, allowing managers to plan an effective approach for the next iteration. Unlike the expected small iterations which occur within individual stages of stage gate processes, and unlike the large but unplanned and unwanted feedback loops which can occur in less successful stage gate processes, iterations in the spiral process are planned and span several phases of the development process. Despite this distinction, critics may consider it similar to a stage gate process if the milestones and deliverables between each spiral round act merely as simple phase gates.

3.3 Evolutionary prototyping and delivery

Learning is not attained by chance, it must be sought for with ardor and attended to with diligence. – Abigail Adams

The evolutionary prototyping PD process differs from the stage gate and spiral processes because it concentrates on learning and gaining feedback from visible prototypes of a

product. As with the other processes, iteration is common, but here the iterations focus on prototyping and refining prototypes until release. Figure 3.5 illustrates the process.

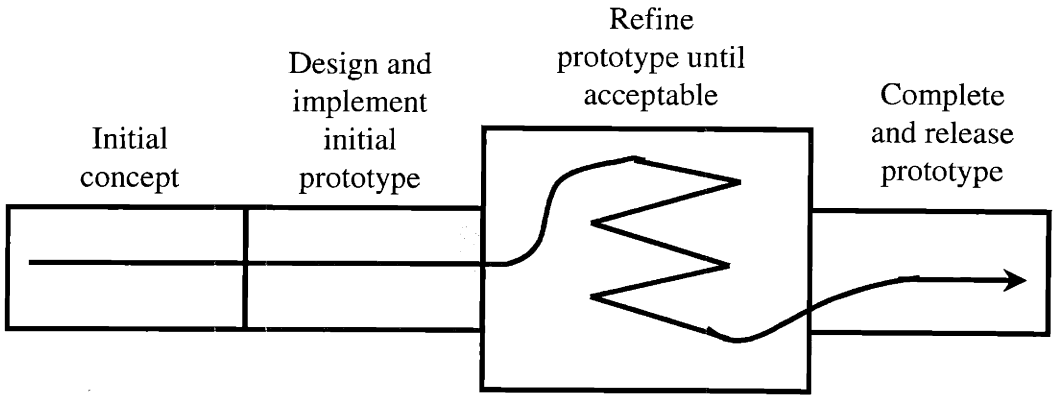


Figure 3.5: The evolutionary prototyping product development process (adapted from McConnell, 1996)

The evolutionary prototyping process allows a team to change requirements between prototype builds, and is therefore useful when the application area is poorly understood and initial specs are unclear. Unfortunately, the process does not have a clearly defined end; prototype iterations must continue until an ambiguous acceptable outcome is reached. Because it is not possible to know how long each project will take and because building an unspecified number of prototypes can be expensive, the schedule and budget risks can be high.

Evolutionary delivery, as pictured in Figure 3.6, is similar to evolutionary prototyping, but has added emphasis on the core design.

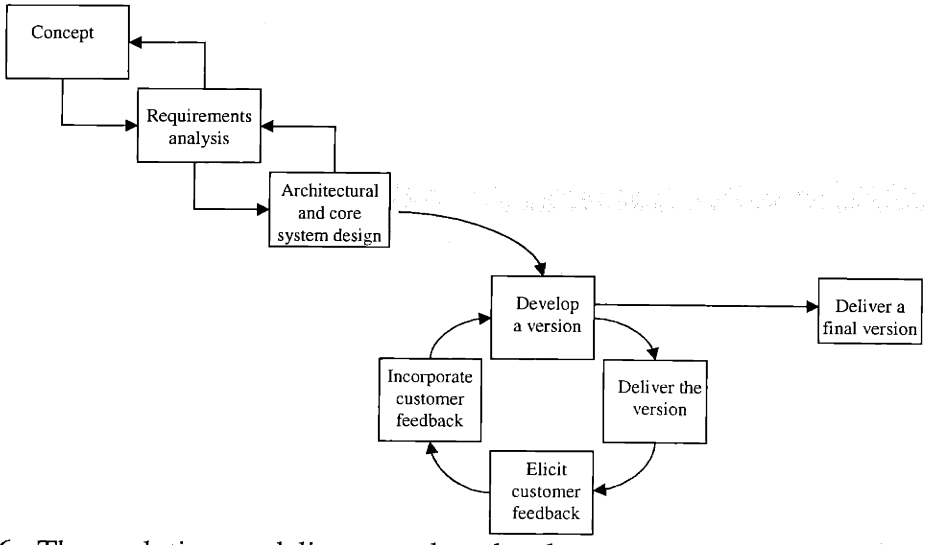


Figure 3.6: The evolutionary delivery product development process (McConnell, 1996)

This method includes a series of iterations in the latter half of the process, but does not mix design and prototyping as thoroughly as the evolutionary prototyping process. The evolutionary delivery process attempts to be flexible by including customer feedback in the iterative loop. However, if a company is prone to accommodate most customer requests or changes, it might as well use evolutionary prototyping. Evolutionary delivery merely structures the beginning of the process more formally, so that core detailed design is more insulated from prototyping and changes due to customer suggestions.

3.4 Design to schedule/budget

"You may delay, but time will not."
– Benjamin Franklin

A final PD process involves yet another type of iteration designed to limit time and budget risk. A design-to-schedule or design-to-budget process can begin as a waterfall, but then intentionally switches to cross-phase iterations during the second half of the process. This is demonstrated in Figure 3.7.

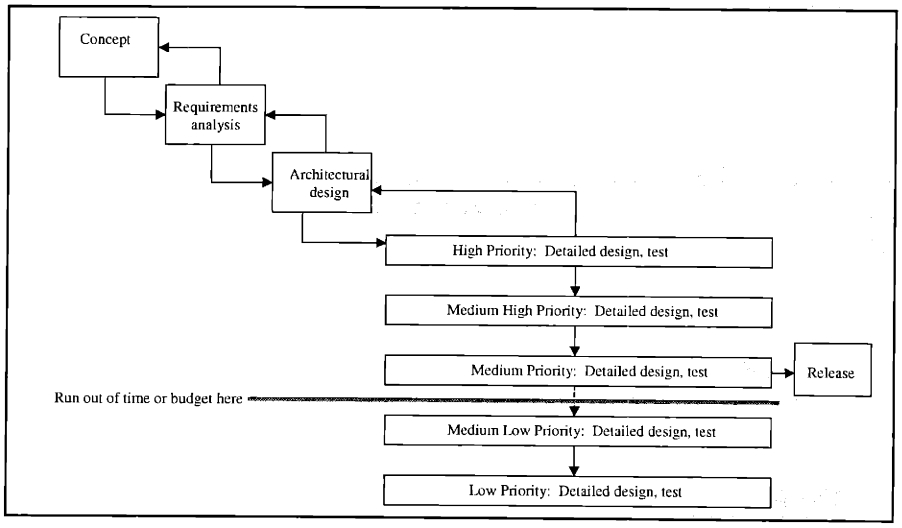


Figure 3.7: Design to schedule or budget

In this case, the later steps of detailed design and testing are merged so that developers can more easily iterate between the two. In this way, the design to budget and design to schedule methods do away with the rigidity of waterfall-style gates between phases and

add flexibility. The difficulty encountered, however, is that disposing of phase gates can lead to wider, less controlled iterations. To compensate for this, iterations are limited by quality, schedule, and budget considerations.

The iteration limitations begin when quality and functionality concerns are prioritized. Once tasks are organized by importance, product iterations – each new, would-be release is an improvement over an earlier prototype – may occur until a budget or schedule limit is reached. If regular integrations are part of the process, the product can be released whenever that limit is reached. The external limit provides the discipline foregone in the merging of process phases.

By having strict budget and schedule limits, this process minimizes related risks because their favorable outcomes are virtually assured. In this process, the risk left unchecked is technical. Functions and tasks may be misprioritized or the budget/time deadline may arrive while some high priority items still require improved design and testing. The results can be a low quality product or, in the case of a platform with plug-ins, a product without all of the expected features.

3.5 Other approaches and tools

There are several other methods that companies may employ as part of PD. These peripheral methods include broad scale approaches, specific tools, and acquisitions. Although they do not constitute defined PD plans, they are important to many companies' PD efforts.

Broad scale approaches are general strategies that companies can adopt as part of their PDPs. Examples include set-based concurrent engineering and extreme programming. Set-based concurrent engineering is a method of designing several alternative designs in parallel. This method delays major design decisions in order to gather more data. The delays and discarded efforts come at a cost, but the method can save time and prevent rework if it avoids the problem of poor product design followed by a series of changes and fixes. (Ward, 1995) Extreme programming (XP) is a flexible and fast-changing

software-specific PD method based on a collection of perceived best practices. Companies engaged in XP employ pair programming, continuous testing, and continuous feedback from small releases and short cycles. XP is geared towards small teams of programmers, although it has been adopted by some large companies as a component of their PDPs (Beck, 2000)

Companies may also pursue PD by purchasing off-the-shelf tools and sometimes even "off-the-shelf" products themselves. Although they are not the focus of this research, there are many examples of PD management software which developers can purchase in order to streamline their concept development, automate their change notices, or share design information between groups. These are used as part of companies' broader PDPs. Finally, some companies supplement their own PD efforts by buying the development efforts of others, although this is not a substitute for actual product development. If a company can afford to engage in such leapfrogging, it may either acquire rivals and the rights to their products or else license the commercial rights of another company's technologies and products.

3.6 Chapter summary and discussion

This chapter demonstrates how companies face different risks and employ a variety of PDPs to manage those risks. The PDP models differ in appearance, organization, and risk management. The unique advantages and disadvantages of each PDP suggest that no single PDP is suitable for all circumstances. Companies can choose to follow any of several process variations, but knowing which one to adopt remains a problem.

The following chapter provides an improved means for describing and comparing PDPs so that managers can evaluate them based on criteria other than mere process shapes or lists of general advantages and disadvantages. Once these metrics are defined, actual PDPs are introduced as data to support the validity and utility of the proposed parameters.

4. CHARACTERISTICS FOR PDP COMPARISION

"I do not distinguish by the eye, but by the mind, which is the proper judge." – Seneca

This chapter proposes characteristics by which different PDPs can be defined, compared, and contrasted. It begins by reiterating the academic and industry need for these characteristics. Companies try to balance structure and flexibility in their PDPs, but have difficulty measuring degrees of either structure or flexibility. Characterizing PDPs requires identifying basic traits that are shared by all processes: all PDPs employ design reviews, which uphold standards and or mark milestones; and all PDPs include iterations, which incorporate changes and feedback loops between design groups or project phases. Characterizing PDPs also requires tenets that set PDPs apart: although all PDPs use reviews and iterations, the *manner* of reviews and iterations varies dramatically. They may vary in rigidity, frequency, scope, or several other parameters that affect risk management. Thus, reviews and iterations – incorporating specifications, milestones, integrations, and tests – are advanced as useful characteristics for distinguishing PDPs.

As PDPs manage risk, they must balance structure, or predictability, and flexibility, as Figure 4.0 shows. Many managers recognize this balance as a common and inherent trade-off between order and creativity.

Predictability Flexibility

Figure 4.0: The continuum of PDP emphasis

PDPs must fall somewhere along the arrow in Figure 4.0, but the continuum is unfortunately so broad that it offers little insight as to where a PDP should be placed for maximum effectiveness. This figure of a trade-off is not helpful until it is deconstructed, as the following pages will demonstrate. The need for greater specificity requires the segmentation and quantification of important PDP characteristics so that processes can be compared. The characteristics must be traits shared by all PDPs.

All PDPs iterate and review; their ubiquity makes these two characteristics the ideal metrics for PDP comparison. The types of iteration and review and the relationships between them may be different, but all PDPs have at least some form of both.

4.1 Design iterations and integrations

"Oh, thou hast a damnable iteration, and art indeed able to corrupt a saint."
– William Shakespeare, King Henry IV

Given the uncertainties inherent in PD, iteration is inevitable and must be managed effectively. Iteration is technically defined as the repetition of an action or process. This meaning is laden with value judgments and can be perceived positively (as in renewal) or negatively (as in continual repetition.) The word stems from a more neutral word, the Latin *itarare*, meaning to repeat or rehearse. This research defines iterations broadly to include almost any kind of stepwise work that involves correction or feedback between interdependent parts, people, or processes. Integrations and tests are one form of iteration that allow feedback from early versions of products.

Effective PD efforts attempt to anticipate the future, but such prognostications can be dangerous business because, as the old adage goes, "the forecast is always wrong," especially in an era of rapidly emerging and immature technologies. (de Neufville, 1990) Intentional iterations for the purposes of feedback may allow some "future" knowledge to be learned earlier, thus reducing development risk. (Ward, 1995)

Iteration is implicit in much PD and quality management literature. For example, the PDCA (Plan, Do, Check, Act) process shown in Figure 4.2 has four key steps: (1) planning an improvement, (2) making the improvement, (3) checking the improvement results, (4) acting and replanning if necessary. PDCA is considered a cycle because successive rotations (analogous to iterations) make progress possible. Similarly, the WV framework, which is named after the zigzag pattern that models it, alternates between thought and data as it moves forward to solve a problem. (Kleim & Ludin, 1997; Shiba et. al., 1993) Both PDCA and WV are predicated on the idea of testing, assessing, or gaining knowledge and then using that information in a later repetition of the same

process. Both iterate in order to make use of feedback. Other, more sophisticated frameworks and tools also consider iterations and feedback loops to be key components of PD. (Eppinger, 1994 & 2001; Eppinger & Whitney, 1996)

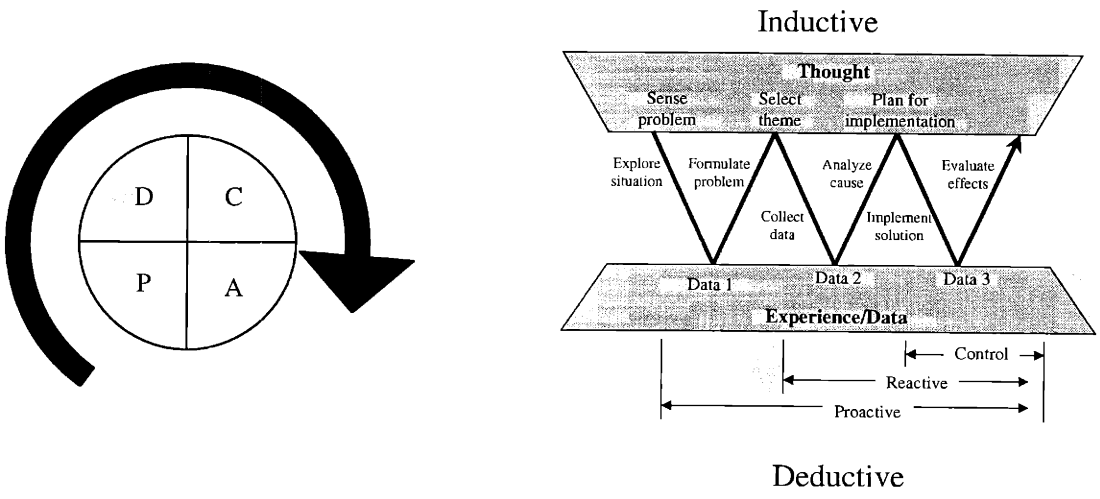


Figure 4.1: The PDCA cycle (left) and WV framework (right) are both based on iteration

Interdependent and complex tasks that require feedback introduce the potential of burdensome and expensive rework if poorly managed. Rework, a combination of feedback and corrective action, is also a type of iteration but is generally wasteful because it is a response to avoidable mistakes. Although rework can be considered a specific and unfortunate type of iteration, iteration is not synonymous with rework. Instead, well-managed design iteration can *prevent* rework and therefore reduce technical, schedule and budget risks. Other types of iteration, such as presenting a customer with a prototype to gauge consumer demands, can also alleviate market risk. Effective iteration can prevent waste and overcome the uncertainties inherent in interdependent tasks.

How can iteration prevent rework when it involves doing something over again? The answer lies in the type of work done in each iteration. Iteration is more than merely trial, error, and rework of previous wasted effort. Effective iteration provides feedback with each round, thus increasing the likelihood of success in the next round. An analogy can

be seen in the simple "higher/lower" child's game that involves guessing a number from 1 to 100. The guesser states a number and then learns if the correct answer is higher or lower. The guesser proceeds to iterate logically, narrowing down the choices and margin of error until finally the correct answer is reached. The first few iterations narrow down the most, while later iterations pinpoint the final solution. Although early guesses are frequently wrong, they do not waste effort if they are chosen strategically.

The iterations in this simple example of a game are analogous to well-managed iterations in product development. There are many types of iterations in product development. Iterations can vary in three main ways. First, they can vary in breadth or scope of iteration. Second, they can vary in the number of inter-phase loops they entail. Finally, iterations can vary in degree of planning. Each of these three parameters can be seen in Figure 4.2.

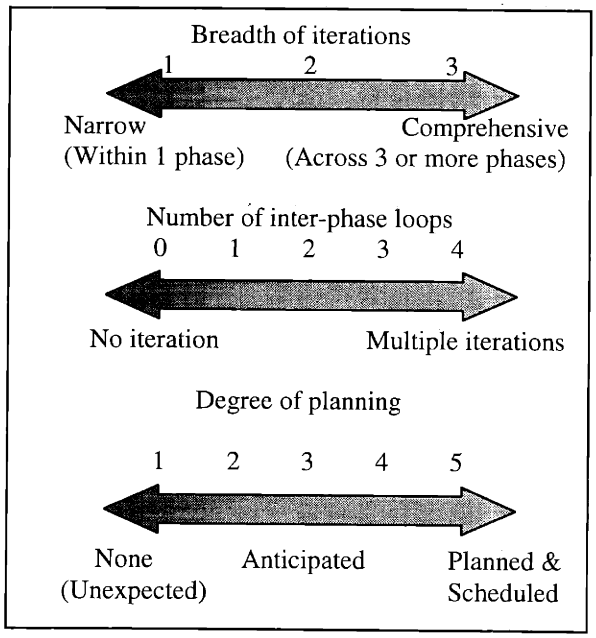


Figure 4.2: Parameters for measuring PDP iterations

The first parameter, the breadth or scope of iteration, is a critical descriptor of a company's PD process. Breadth can range from narrow to comprehensive. Narrow iteration is intra-phase, exemplified by several rounds of interdependent detailed design tasks. Comprehensive iteration is cross-phase, exemplified by processes that cycle not

just around a specific stage, but rather over a range of process stages from concept to prototyping. Both narrow and comprehensive iterations are shown in Figure 4.3. There is a continuum between these two types of iteration, and processes vary in their iteration breadth.

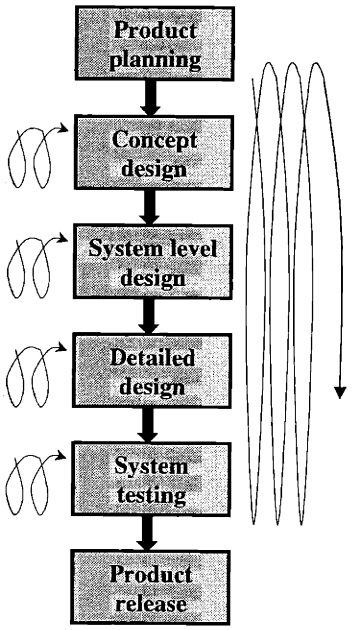


Figure 4.3: Narrow iterations (left) and comprehensive iterations (right) in a PDP

The number of iterations can also greatly affect the nature of a PDP and its success in managing risks. Whether a design is considered several times or just once is a major distinguishing feature between processes. Only the inter-phase loops are of importance to this part of the study because intra-phase loops are so common (and often automated in CAD programs) that they can barely be distinguished from one another.

Finally, the degree to which iterations are planned also varies. Companies may have *unplanned*, *anticipated*, or *scheduled* iterations. *Unplanned* iterations occur when mistakes or feedback loops unexpectedly require a step backward, often in the form of regrettable rework. Anticipated iterations are iterations that are planned or expected, but that do not have specific schedules and which may not happen at all. For example, a manager who expects several rounds of detailed design on a specific component may be familiar with the design process and expect to succeed on the third try. A fourth try is not

out of the question, and a lucky estimation might allow for success on the first try. Here, the iteration is *anticipated* – it is tacitly expected and the routine is known – but the number and time of iterations is not planned. Finally, *scheduled* iterations are both anticipated and planned. The number of cycles may be planned, may be subject to time and budget constraints, or may be dependent on customer satisfaction and quality assurance. In cases of product or process failure, the number of iterations may expand unpredictably.

4.2 Design reviews

"Be sure you are right, then go ahead."
–Frontiersman and future congressman David (Davy) Crockett during the war of 1812

Design reviews are critical to product development. Like iterations, they are present but different in all PDPs. Design reviews are sometimes called gates, checkpoints, or milestones, but always involve a decision or assessment of progress. As demonstrated in Figure 4.4, reviews examine the deliverable of previous action and decide whether to continue on to the next step, stage, or series of stages.

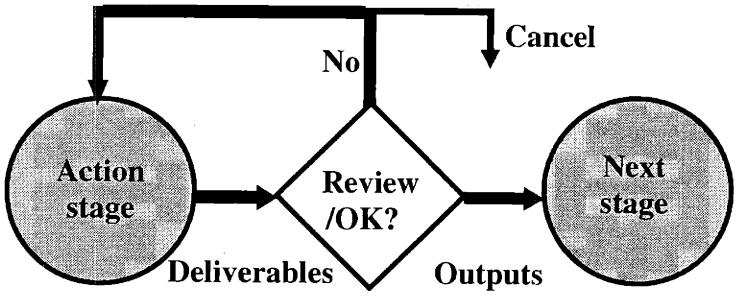


Figure 4.4: The format of a design review

Companies developing products handle reviews in different ways. The goal of some reviews is to assess completion, while the role of others is to ensure that there are no technical design problems. Sometimes the reviews are internal and performed by the design groups themselves, while other times reviews are performed by upper management or by a disinterested group of peers from another project. The level of formality of the reviews also varies dramatically. Some companies treat reviews

casually, while others have strict qualifications and training requirements for anyone who is to be a review leader.

Figure 4.5 shows how two main factors, rigidity and frequency, serve as metrics to characterize reviews. Rigidity of review is defined by the degree to which deliverables are held to previously-established criteria. In a rigid scenario, a project is probed for problems and not allowed to continue until a deliverable exactly matches or exceeds the criteria. Failure to do so impedes the entire development process. In more flexible situations, projects or designs may conditionally pass reviews, subject to assurances of future change. In the most flexible cases, reviews can be mere checks, design assessments or mere status reports of developments or problems.

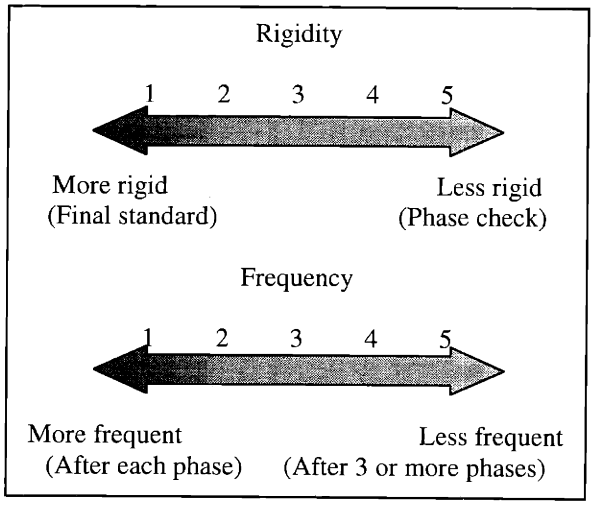


Figure 4.5: Parameters for measuring PDP reviews

Frequency also affects the character and impact of reviews on PD. Some companies have reviews at regular time intervals, thus forcing the completion of activities or integrations on a regular schedule. However, most companies schedule design reviews at the planned completion of a deliverable. Deliverable-based reviews have the advantage of always having deliverables in existence to judge, but may occur at irregular intervals. Irregular timing can be due to schedule delays, to variation in the amount of time it takes to complete different phases, or to variation in whether the deliverables are the result of

either one or several phases. For example, in stage gate PDP, reviews occur after each stage. In spiral PDPs, reviews may occur after each spiral, or series of stages.

4.3 Risk management through iteration and review combinations

PDPs include many possible types and combinations of iterations and reviews. Each combination can act as a fingerprint that identifies or defines a PDP. Each iteration/review combination also manages risk differently.

4.3.1 Fingerprinting and identifying PDPs

Iteration and review combinations can act as fingerprints by uniquely identifying or defining PDPs. For example, stage gate processes entail narrow iterations and rigid reviews after each stage. Conversely, spiral processes employ more comprehensive iterations and flexible reviews after several stages. To illustrate how an iteration/review combination can define a PDP, the spiral PDP of Figure 3.4 is redrawn in Figure 4.6. The figure shows the same process with the same number of iterations and reviews, except Figure 4.5 more clearly displays the breadth and number of cross phase iterations. This sample process has two cross-phase iterations that span five stages of development, one iteration that spans four stages of development, and several smaller iterations that span only two phases. Since this process was presented earlier as a model spiral process, most of these iterations are planned. This information, when combined with knowledge about the type of design reviews the process employs, allows an observer to set it apart from others. Other PDPs, such as the design-to-budget process or evolutionary prototyping process, can also be defined and identified by their distinct iteration and review characteristics.

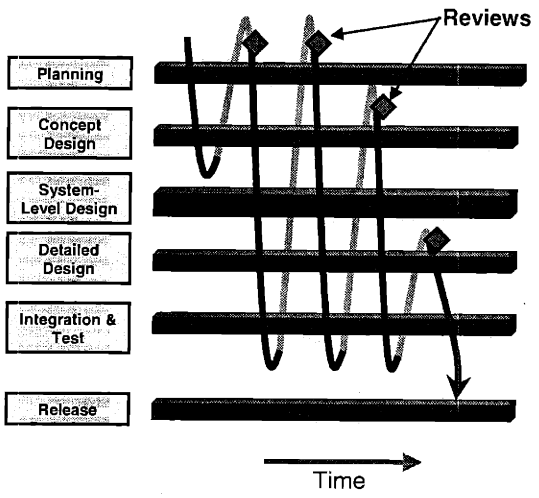


Figure 4.6: The spiral process defined with specific iterations and review characteristics

Measures of iteration and review allow PDPs to be compared more precisely than was possible with subjective descriptions of advantages and disadvantages or informal assessments of process structure or flexibility. Earlier investigations of PDPs either identified only one main process or identified a few and distinguished them only with descriptions of their diagrammed shapes or broad generalizations of their perceived strengths and weaknesses. Now, the characterizations of iterations and reviews become a basis, or language of sorts, by which all PDPs can be distinguished and compared regardless of shape or industry.

4.3.2 Iteration/review combinations manage risk differently

Specific iteration/review combinations manage risk differently. Chapter 3 observes how companies face different uncertainties and risks. That chapter also implies that no single PDP is suitable for all risk circumstances. The characteristics of review and iteration introduced in this chapter help to understand why some PDPs manage certain risks better than others.

A product with many interfaces and interdependencies between hardware and software may face a high degree of technical uncertainty. That technical uncertainty might be best addressed with predictable, early iterations that test the technological feasibility of the concept design and early specifications. It might also attempt to gain feedback from an

early integration as part of one of those iterations. This need to emphasize predictability can also lead a company to use strict, frequent reviews to avoid the need for later changes.

In contrast, a product in an immature industry may face entirely different risks if specifications are defined and frozen early. A company in this situation may opt to employ early market tests to make sure that the specs accurately reflect fast-changing customer needs. The early market tests lend themselves to an emphasis on flexibility and the use of several cross-phase iterations so that information learned from prototyping can be easily fed back to earlier design stages.

Thus, companies should differ in their iterations because of the different risks that they face. Sometimes, the iterations are valued, as in cases when a prototyping iteration identifies a problem that can be corrected easily before a final product release. This is a favorable cost-for-information trade, where a company sacrifices both time and manufacturing cost to build a test model in exchange for knowledge that will allow the company to make corrections, thus reducing technical and market risk. In a particularly fortuitous situation, a helpful iteration may even reduce overall cycle time by identifying "showstopper" problems earlier, thus preventing the PD process from being delayed later. (Boehm, 1988)

Sometimes, however, iterations are frowned upon because they increase schedule risk. Some companies fear a culture of endless iterations and instead emphasize "speeding up the iterative loop" and "doing it right the first time." (Ward, 1995) Others argue that new product contributions in a serial fashion are both "inefficient and ineffective." (Heany, 1989)

Iterating over different sets of PDP stages can have a wide range of effects. Not all iterations address each type of risk perfectly. For example, building a prototype near the end of a development project may mitigate technical risk by determining if the product performs to the level of quality promised by design. It may also address market risk by

providing information on whether the product will satisfy customer needs. However, the iteration may contribute less to mitigating schedule risk because if the prototype is built only late in the process, then acing on the information feedback would force a delay in release schedule. In contrast, an early cross-phase iteration to determine if a potential architecture is reasonable may help managers estimate schedules accurately but will not necessarily mitigate specific market risk.

4.3.3 Parameterization of PDPs

As this chapter proposes, and as will be shown by data and discussions in Chapters 6 and 7, PDPs can be described by their different review and iteration characteristics. Those same variations make certain PDPs better at addressing some development risks than others. The benefits of these categorizations can be seen in Figure 4.7, which summarizes major characteristics of some of the common PDPs described earlier. The variations in the chart's column values demonstrate how the different PDPs are distinct. The differences in risk management suggest a correspondence between the types of iterations and reviews and the kinds of risks addressed.

Parameter		*Waterfall/ Stage-gate*	*Design to Sched/Budget*	*Evolutionary prototyping* Start	End	*Spiral*
Iterations	**Breadth**	1	2	1	3	3
	# of inter-phase loops	0	Unspecified	0	Unspec-ified	Unspecified, but many
	Planning	2	3	2	3	5
Review gates	**Rigidity**	1	2	3	3	5
	Frequency	1	2	2	1	Unspecified
Risk	**Profile of key risks**	Manages tech risk well	Manages sched./budget risk well	Manages market risk well		Manages market risk well

Figure 4.7: Parameterized common PDPs

Figure 4.7 uses only a few simple parameters to characterize different processes. These overarching merics of iteration and review are not the only characteristics relevant to PDPs, but they are the only ones universal across all PDPs. If decomposed, some tenets of iteration would appear in only some industries. We could potentially enrich our PDP

descriptions by considering other factors such as degree of integration within iterations, lead times, or degrees of modularity, but the added depth would come at the cost of reducing the scope of comparison to individual industries.

Further decomposition of iteration characteristics highlights how some categorizations can be local to certain industries. Cross-phase iterations often include some aspect of integration when different detailed design components are assembled as part of a prototype (or "build" or "digital buck," depending on company jargon). The degree of integration could potentially serve as another distinguishing characteristic between PDPs. However, the fidelity of those prototypes and assemblies varies dramatically across industries. It is relatively easy to gain feedback from software or digital integrations because no physical components have to be built or assembled. Manufactured products, on the other hand, may gain the ability to perform frequent or rapid integrations with digital modeling, but the models are only models; they do not provide the same fidelity of feedback as software integrations. Among manufactured products that must be physically integrated and tested, the ease of prototyping and integration may rest on the types of materials used. For example, metal parts that can be cast quickly are more likely to lead to fast lead times than components that must be more painstakingly forged and machined. As a result, analysis that dealt with more specific characteristics such as degree of integration within iterations would find itself analyzing mainly software companies while most manufacturing companies' PDPs would be grouped together with little to distinguish them.

In addition to being decomposed, the iteration and review characteristics can be aggregated and averaged to provide a general measure of PDP flexibility. In general, higher values in any of the categories defined above imply a more flexible process. Thus, a process with comprehensive iterations (maximum score=3), four reviews (score=4), high degree of planning (score=5), less rigid reviews (score=5), and infrequent reviews (score=5) would be the most flexible process. Normalized to a common scale, a weighted average of these measures can provide a rough measure of process flexibility.

4.4 Chapter discussion and summary

This section explains how PDPs can be distinguished by the types of iterations and reviews they employ. These proposed characteristics not only create a fingerprint by which PDPs can be identified, but provide a means of PDP comparison based on risk management ability. The segmented characteristics can help determine where PDPs lie on the scale of structure and flexibility. Further segmentation or decomposition of the characteristics is possible and can be descriptive, but is only helpful in describing certain industries because the decomposed characteristics are not universal across all PD.

The proposed descriptive characteristics are an improvement over the descriptions in Chapter 3 and allow for a common framework in which all PDPs can be compared. The next three chapters introduce and discuss actual data and case studies to support the utility of these characteristics.

5. METHODOLOGY

"It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."
– Arthur Conan Doyle's Sherlock Holmes in A Scandal in Bohemia

5.1 Case studies

This chapter explains the methodology of the ten company case studies that form the basis of the research findings. Case study methodology suits the goals of this research for three reasons. First, it provides empirical data to help build theory about the subjective relationship between PDPs and risk. Second, it demonstrates the utility of using quantitative iteration and review metrics to characterize PDPs and distinguish them from each other. Finally, the resulting understanding of some companies' PDPs provides counterexamples to conventional wisdom regarding the applicability of certain processes.

Case study research is a well-established social science research method. It is an effective practice for understanding phenomena in depth, especially while building theories in an immature or subjective field of study. (Judd, 1991) Case studies can also help build grounded theory in areas where initial ideas or hypotheses may change over the course of the study. (Cressy, 1953; Dougherty, 2002) These traits make case studies appropriate to this research, which investigates the relationship between different PDPs and development risks.

Case studies also support the second goal of this research, which is to propose characteristics that describe and distinguish different PDPs more effectively than before. There are no currently established quantitative measures for PDP comparison. In proposing such metrics, they must be compared to qualitative process descriptions; case study methodology is particularly useful for gathering this qualitative information. (Judd, 1991)

Finally, case analyses can effectively provide counterexamples to existing theory and conventional wisdom. If conventional wisdom or existing literature suggests that certain PDPs only apply in certain cases, a single example of an exception may suffice to

successfully challenge the idea. Although discovering such exceptions was not the main goal of this research, some findings did in fact counter the presumptions of prior literature in the field.

The limitations of case study research were of limited consequence to this research. Case study research has inherent limitations in proving causality because cases demonstrate only their own existence. (Ward, et. al., 1995) However, this research does not attempt to prove causality between development risks and PDP design. Rather, its main goal is to build grounded theory explaining the important but non-causal relationship so that it can be considered in future decisions.

Other limitations of case study methodology include its labor intensity, which is generally excessive and cumbersome if quantitative data can instead be gathered quickly from a larger population. However, although this research lays the groundwork for quantitative comparison of PDPs in the future, the current infancy of the field forces it to compare any quantitative findings to the qualitative data from which they derive. This qualitative data is effectively gathered through interviews, discussion, immersion, and other case study techniques that increase depth of understanding. Limited quantitative data – in this case the iteration and review metrics proposed in the previous chapter – are determined in two ways. Some measures, such as the number of iterations, can be directly ascertained from both interview and questionnaire responses and from observation of the number of cycles in a development process. Other measures, such as the rigidity of review, are gauged based on an pre-established rubric that allows for an objective comparison of different interviews with subjective responses of how "tough" reviews are. These standardizations of subjective responses can be reinforced by documentation, such as company procedures that define design review scorecards or passing criteria.

Most of the case studies in this research were specifically selected to represent different data points in disparate industries and circumstances. For example, companies can develop either manufactured goods or software, or could face primarily technical or

market risk. Thus, IDe was selected because it develops software, while SWPG and UTC were selected because they develop turbomachinery. Xerox was included in part because its products included both large software and hardware (manufactured) components. ITT Industries was included because it was anticipated that its role as a defense contractor would lead it to have a uniquely different risk profile from most other companies in this study.

Several case studies were selected either because of opportunism or because they were based on companies with public PDP information. Opportunistic case studies occurred when some companies were included because of professional contacts or CIPD sponsorship. Care was taken to reduce any possible bias in these cases, and the validity of this practice is supported by Buchanan et. al. (1988) who advocate balancing what is theoretically desirable with what is practically possible. Case studies of companies using public data provide other researchers or reviewers with the means of independently examining source data. It also allows readers who are familiar with some companies' PDPs – such as those of Microsoft and Ford, which have been extensively investigated by many researchers – to compare these research findings to their own knowledge and interpretations.

5.2 Company approaches and data collection

The goal of each case study was to gain a rich understanding of the company's risks and PDP. The challenges were to identify what type of subjective risks were greatest and to learn of any differences between official company PDPs and the processes that were actually implemented. Meeting those challenges required conducting interviews, administering questionnaires, reviewing public company literature, and studying private company PDP documentation.

In most cases, one company manager served as a lead contact and provided process documents and lists of employees working on specific product development teams. In some cases, the lead contact would also recommend studying certain product lines in response to the request to examine both "new" and "variant" products. When available,

official process documents were always read first. Later, project team members were interviewed or given questionnaires about their PDPs. A sample interview guide and questionnaire are included in Appendix A.

Interviews followed the procedures for semi-structured "interview-conversations" described by Blum (1952), Burgess (1984), and Buchanan (1988). Some common PDP questions were asked consistently in all interviews, but in most interviews the latter half was conversational and varied according the person interviewed. Areas of questioning included both the PDP and development context. PDP questions dealt with review and iteration characteristics, implementation of the official PDP, and perceived problems and advantages of the PDP. Contextual questions probed the types and timing of prototypes, tests and validations, program schedules, budgets, and major risks.

Most interviews were one-on-one discussions with employee expectations of anonymity. Anonymity remains important because of the sensitivity of some questions about PDP implementation. In some cases, official PDPs were not followed faithfully or were criticized by interviewees, who were more at ease making admissions or accusations because they were assured that they would not be personally identified. Some interviews were recorded on cassette tape when allowed, but only for purposes of later transcription. In addition to private interviews, case studies at two companies (ITT and Printco) also included public group discussions of the companies' PDPs, prompting open and lively debate on the implementation, merits and disadvantages of their development processes.

Some companies required non-disclosure agreements prior to the interviews. In these cases, descriptive summaries and interview quotes were submitted to the companies before the public distribution of the research. Companies were invited to make limited editorial changes to "sanitize" the results and protect confidential information. In all cases, these editorial changes were cosmetic and had no discernable effect on the overall results. (Burgess, 1984)

As discussed earlier, some companies were investigated with the help of public data, either instead of or in addition to interviews. The purpose of including such examples is to allow other researchers to either repeat the exercise or compare results to their own knowledge of the same public information. In these cases, such as Microsoft and Ford, existing literature was considered first, followed by any data from interviews.

6. CASE STUDIES AND RESULTS

"...We can pursue all these studies until we see their common ground and relationship, and can work out how they are akin" – Plato's Republic

This chapter presents case studies of ten companies and their PDPs. Each case study begins with a qualitative description of a company's development context, products, and major risks. Next, each study describes how the company employs reviews and iteration, including integration and testing, in its PDP. The final section of this chapter presents comparative findings from all ten case studies and shows how their processes differ according to quantified metrics.

The cases include five primary case studies and five secondary studies. The primary studies examine Siemens Westinghouse Power Generation (SWPG), Integrated Development Enterprise (IDe), ITT Industries, Xerox, and Printco. In these cases, observations about PDP implementation are supported by representative and illustrative quotes from interviews or questionnaires. These quotes are usually attributed to people by title. Interviewee names are included in Appendix B. The secondary studies include Aviation Technology Systems (ATS), Ford Motor Company, United Technologies Corporation (UTC), DeskArtes, and Microsoft. These case studies are shorter and are used to support and extend theories derived from the core case studies. The secondary case studies use information that was collected primarily from public sources instead of through company interviews or questionnaires, although several employees were interviewed when public data was scant. These employees are also listed in the appendix.

These case study companies represent several different industries and operating environments. Four of the case study companies produce mostly software. Six of the case study companies produce mostly manufacturing goods, although several of them have important software components in their products. Most cases study subjects are large corporations, although three of them are smaller or startup companies that number (or numbered) their employees in the hundreds rather than in the tens of thousands.

Some of the companies almost served as two case studies in one. ITT and UTC, for example, are both large conglomerates whose different divisions or units sometimes follow different processes. In each of those cases, two different products were investigated. Printco also provided multiple product and process examples.

Because the case studies attempt to paint a realistic, "as-is" portrait of the PDPs, they do not simply repeat official company process documentation. What companies say they do is not always what they actually do. The case studies in this chapter reach beyond formal company descriptions to include individual engineers' and managers' assessments of how the PDPs are actually implemented. This research focuses on the elusive PDPs *as implemented*, because otherwise its findings on iterations and reviews would be based on fictional processes.

Each case study description begins with a summary outline box explaining key points and traits about the company and its PDP. The metrics are based on the parameters defined in Chapter 4. After the summary box, each section briefly describes the company and the product whose development was investigated. The studies then describe the official company PDPs before discussing how the PDPs actually function.

6.1 Siemens Westinghouse Power Generation

Case study synopsis

Company: SWPG
Type: Large manufacturing company
Products: Turbomachinery for power generation
Cycle time: Slow (years)
PDP: A rigidly documented and implemented stage gate process
Major risks: Technical risk is based on QA requirements and need to reach high thermal efficiency. Market risk is mitigated by early contracts and system of guaranteed liquidated damages.
Notes: "Phantom" gates or reviews sometimes cancel projects, but otherwise the strict stage gate is strictly implemented.
PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
1	0	1	2	1

Company and product description

Siemens Westinghouse Power Generation (SWPG) designs and manufactures equipment that generates electricity in power plants. The company has nearly 26,000 employees and net sales of over $8 billion per year. This case study focuses on the development of gas, or combustion, turbines that are used in single or combined cycle power plants. The market for these products consists mainly of utility companies and independent power producers who choose between Siemens-Westinghouse, GE, and only a few other companies in an oligopolistic market.

Siemens and Westinghouse are former rivals, and the long-term innovation and technology strategies of the entire turbine industry are described earlier in section 2.4. Fifty years of competition and innovation resulted in a mature market and an industry-wide development emphasis on raising efficiencies. The two companies merged when Siemens acquired Westinghouse Power Generation from its parent company (CBS) in 1998.

Combustion turbines are large machines that must be designed to precise measurements as small as thousandths of an inch because of the narrow clearance of their many turning blades and vanes. Gas turbines are supplied as parts of various packages, ranging from "econopacs" to "power islands" to "turnkey plants." Econopacs include only the turbine-generator units with their fuel and control systems. The intermediate power islands include heat recovery steam generators and additional power generation equipment to capture waste heat from the gas turbine. Finally, turnkey plants include full balance-of-plant piping, pumps, water towers, transformers, and switchgears.

In contrast to turbine packages, individual turbines are generally not modular despite some attempts by the company to modularize some components for common use between different turbine families. Most turbines are sold as packages and most turbine components are unique to a single turbine product.

One example of an individual combustion turbine, the 501G, can be seen in Figure 6.1.1. The 501G is the largest turbine in the 501 family of turbines and can generate 253 MW at a maximum efficiency of 39% as a single cycle turbine. When combined with a downstream steam turbine as part of a combined-cycle power plant, overall thermal efficiencies can reach almost 60%.

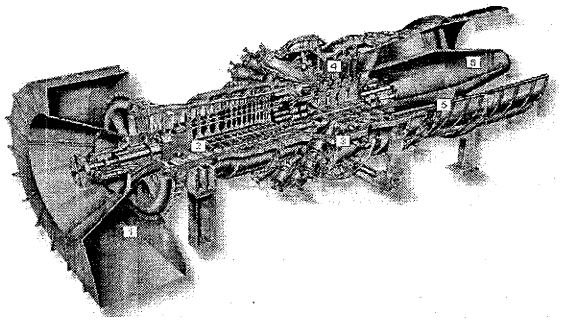


Figure 6.1.1 The Siemens-Westinghouse 501G combustion turbine

The most prominent risk in turbine development is the technical risk, in part because market risk is reduced by early contracts and a system of liquidated damages. As described in section 2.4, the market for turbomachinery is relatively mature and well-

understood. Competition between manufacturers is tremendous (since a ring of collusion was broken by government regulators in the 1970s) but occurs primarily on the basis of efficiency (turbine heat rate) and price. As a result, marketing and sales divisions contract to sell, design and install power plants and guarantee the heat rate of the product in the form of contractual liquidated damages.

A guarantee of liquidated damages is different from a standard warrantee. Warrantees are guarantees of repair or replacement of damaged or inadequate parts for a specified period of time. However, entire turbines are too large, too rare, and too expensive to be replaced entirely in case of systemic underperformance such as a "missed" heat rate. Guarantees of liquidated damages are actual cash payments to compensate a customer for any loss suffered by low efficiencies.

The result of this contract system is an early freezing of customer specifications. Sales and marketing departments promise low heat rates (high efficiencies) to customers and these aggressive targets are assigned to engineering early in the development process. The "frozen" requirement specifications lead to limited market risk but high technical risk. Budget risk is relatively low because there is little expectation of profit in the initial turbine sale. Most company profits are made in service fees and replacement parts. Technical risk is the preeminent development risk, which explains why SWPG uses a waterfall PDP.

PD process description

The current Siemens-Westinghouse PDP was installed shortly after Westinghouse was acquired by Siemens. Until then, the company used a less formal stage gate process and frequently engaged in one-time products and customer-order engineering. The merger led to the six-month launch of a new, more formal PDP and an increased emphasis on product families and platforms.

The SWPG process is a strict stage gate process, as demonstrated in Figure 6.1.2. The PDP has a series of gates and reviews that are absolutely sequential and rigid. The five gates (G) and eleven major reviews (R) include:

G1 Program initiation	R0 Review of Product Strategy
	R1 Review of Product Requirements Specification
G2 Design	R2 Review of Product Design Specification
	R3 Design Review
G3 Product release	R4 Commercialization Plan Review
	R5 Final Design Review
	R6 Procurement Review
	R7 Product Review
G4 Series release	R8 Commissioning Review
	R9 Product Monitoring Review
G5 Program closure	R10 Performance & Reliability Review

Gates are controlled exclusively by managers and are decision points where managers determine if an entire program will continue or be terminated. Reviews include larger meetings, engineers' presentations, and technical questions and evaluations.

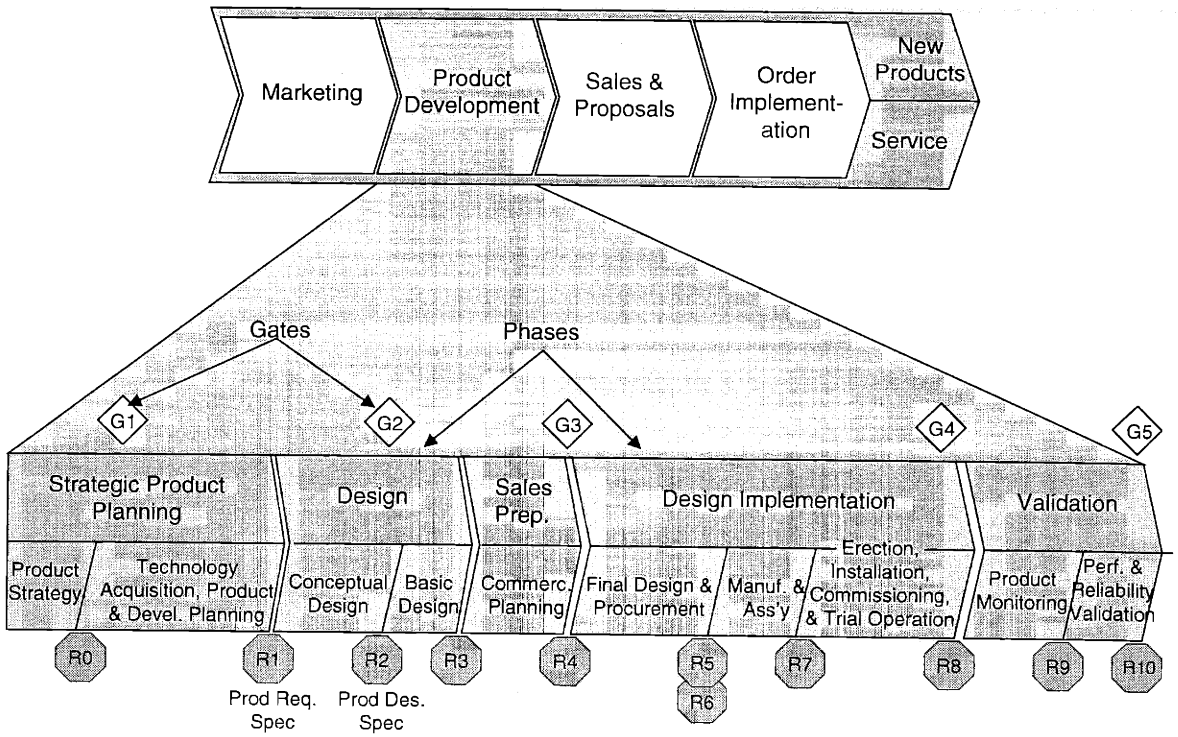


Figure 6.1.2: The SWPG product development process

Of particular interest is the great rigidity of the reviews. Each review committee includes peers and managers assigned by position and training. The PDP even assigns qualifications to design review leaders, who must have experienced a minimum number of reviews previously and who must have completed internal company training on leading design reviews. The review criteria are equally rigid. Criteria, in the form of a checklist, are assigned before the review takes place. With rare exception, each criterion must be fulfilled for work to continue. On some occasions designers may receive a "conditional release" at a review, meaning that work is allowed to continue while a design is changed or corrected. In these cases the PDP specifically defines who may waive the normal rule and also assigns a maximum length of time allowed for the corrections to occur.

Cross phase iteration is almost nonexistent. Only intraphase design iteration is common. These narrow iterations sometimes limit the ability of designers to respond to market needs that change during product development. As a result, the company attempts to add some flexibility by dividing specifications into product requirement specifications and product design specifications. Both types of specifications are frozen relatively early in the process, but the time lag between them allows for some specifications to be frozen later than others.

In practice, there are some exceptions to the official PDP. These unofficial events, reviews or changes are very much part of the overall company PDP, although their existence would be considered failures according to the written process. One instance is the existence of unofficial reviews. In the words of one design engineer,

> Every once in a while a program just stops. Somebody from above kills it, but not at a review and not at a gate. Those sorts of decisions are only supposed to happen at set times, but sometimes we're just told not to work on a project anymore...it's like a phantom review.

Such phantom reviews do occur and reflect the ability of upper management to overrule the company PDP in extreme cases. Sudden cancellation of a development program (in favor of others) usually occurs if a market change creates the potential for a particularly dire financial situation if work on a certain program continues. Since such cancellations

are not addressed – or technically allowed – by the official PDP, there are no set financial criteria for these decisions. Rather, such cancellations are left to the discretion of senior management.

Another break from formality in the PDP implementation is the inequality of complexity between reviews. Although the PDP does not state this explicitly, some reviews are more important than others. This does not diminish the rigidity of the less important reviews, but rather emphasizes the difficulty of passing some of the major reviews that involve large numbers of groups or components. In the words of another engineer,

> It all revolves around R5 [the final design review, number 6 of 11 reviews in the process] because that's where it all comes to a head. They're big because there are so many people and components involved.

Such reviews stand in sharp contrast to other reviews, such at the R6 procurement review, for which only one group (supply management) is responsible.

Despite the "phantom" reviews and different emphases placed on different gates, the SWPG PDP is a strict stage gate process with an emphasis on frequent, rigid reviews and narrow iterations.

Case study conclusions

SWPG employs a strict stage gate process to manage almost exclusively technical risk in gas turbine development. The technical risk is high because of the quality requirements of small-clearance parts and the manufacturing requirements of large-scale, complex product platforms. The SWPG PDP allows for few mid-project design changes because specifications are frozen early and cross-phase iterations and design work are rarely allowed. Exceptions to the ordered process occur exclusively in the negative, such as when a project is delayed or killed by senior management.

Other divisions of the company, such as the fuel cell division, must deal with more market uncertainty because immature markets make it difficult to predict which future products might be in demand. In contrast, the maturity of the gas turbine market leads to an emphasis on design for thermal efficiency. The system of liquidated damages

contracts ensures that market variability is translated into a specific cost that can be avoided exclusively through technical success.

6.2 Integrated Development Enterprise (IDe) case study

Case study synopsis

Company: IDe
Type: Small software company
Products: Product development software
Cycle time: Very fast (months)
PDP: Evolutionary delivery
Major risks: Market uncertainty is high. The company risks losing customers if the product is not occasionally customized to meet specific customer needs..
Notes: Originally claimed to be a "spiral-plus" process, the company's custom modifications and tenuous position in a new market make the customer king and lead to multiple deliveries of products. PDP would be spiral if product generations were grouped together as a single evolving product.

PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
2	3	4	4	5

Company and product description

Integrated Development Enterprise (IDe) is a privately-held company that develops and markets internet-based development chain management software. The start-up company, located in Concord, Massachusetts, was incorporated in 1998 with 12 employees. The company has since grown to over 100 employees and now markets four software product lines:

1) *IDweb*™ – development chain management software that also serves as a database to integrate, manage, automate, and reconcile development chain information. This product can include several integrated modules (*IDpipeline*™, *IDresources*™, etc.)
2) *IDpartner*™– software used to integrate and coordinate partners as members in a development chain
3) *IDfinancials*™ – PD financial management software

4) *IDreportview*™ – software used to create customized reports and charts from *IDweb*

The company's products all consist of software that helps other companies manage their product development. As the company itself advertises, the products provide "integrated solutions for development chain management." Its main products are *IDweb* and *IDpartner*, which are both platform software packages. A sample view from the *IDpartner* application is shown in Figure 6.2.1

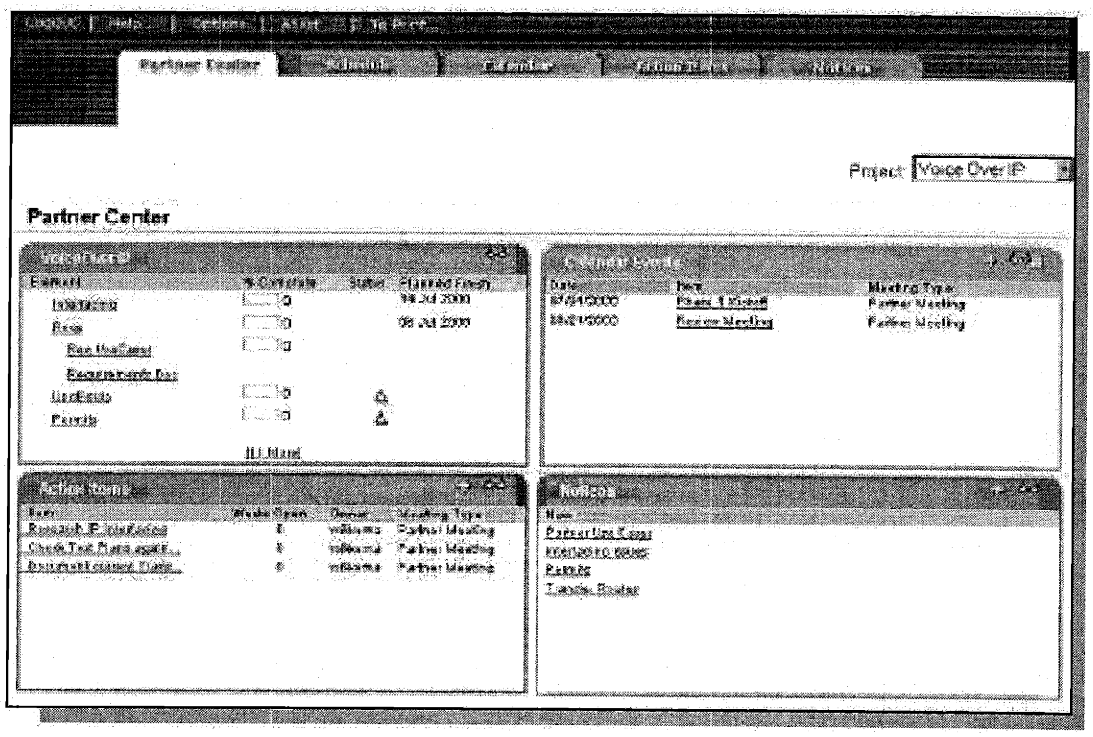


Fig. 6.2.1. The Partner Center screen view of *IDpartner* application. The Partner Center allows the lead developer to consolidate all the shared project information onto one page.

IDe serves a broad market, and its customers range from toy manufacturers to telecommunications suppliers. Although its customers span several industries, all share a need to engage in large-scale product development. IDe software is sometimes customized to users and often requires training and support. The field is still relatively new and customers have unique needs and processes, so the operating environment is fiercely competitive and based on the features and availability (release date) of the software products.

The primary risk that IDe faces is market risk. Market risk is preeminent in part because of the fast-paced nature of the software industry, which demands frequent changes, updates, user comfort, and customization. IDe also faces market risk because it is a new and small company trying to establish itself and build a successful base of major customers.

PD process description

IDe currently follows an evolutionary delivery process, but has changed processes several times during its first few years of existence. When the company first started, IDe began with an admittedly "ad hoc, loosely defined" PD process. An ad-hoc process may have been adequate for a small team working on the initial components of its first product, but by early 2000, IDe had increased in size to 61 employees who were working on 4 or 5 projects concurrently. The company began using a more "reactive planning" development process by using its own *IDweb* software (de facto using and testing its own product) to pinpoint and correct bottlenecks. Changing processes was a balancing act; the company first had a "very detailed" process that one manager asserted was "too rigid" and then overcompensated by introducing a process that was "too generic."

Less than a year later, once the company had grown to over 100 employees working on up to 15 projects, IDe moved from "reactive" to "balanced" planning with improvements in its own planning software and more careful resource distribution among its multiple projects.[10]

IDe's current PD process has four phases which are so broad in scope that, although they appear linear, many nonlinearities and loops occur within them. The four main phases, concept, planning, implementation and launch are demonstrated in Figure 6.2.2

[10] From interview with and information from Ms. Eileen Blanchette, IDe Program Manager.

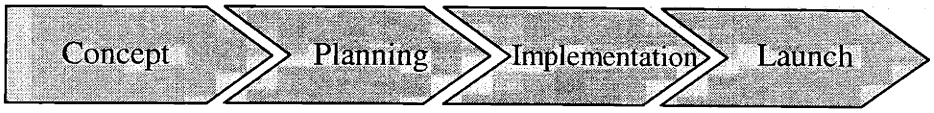


Figure 6.2.2: The IDe PDP Phases

The first three phases each mark their completion with phase reviews, usually by the Product Approval Committee (PAC), a committee of four senior managers who give "go," "no-go," or "redirect" verdicts at each review. The concept phase (Phase 0) includes the definition of the business opportunity and scope, risk assessment, and preliminary resource loading. The planning stage (Phase 1) calls for understanding project completion requirements, identifying project milestones, and forming and engaging the core development team. The core team is a cross-disciplinary team of members from different functions of the organization, (sales, services, development, finance, management, sales) each of whom contributes about 20% of their time to that development program. Implementation (Phase 2) begins with coding and ends with QA/integration testing. Launch includes marketing and training roll-out, but involves few activities for the product developers.

Project timing varies, but an ideal product release takes a relatively short 115 days. Phase 0 takes 7 days, followed by a 21-day planning process, 70 days of implementation, and a 7-day launch period. In calendar time, the company usually takes about 5 months to complete a variant version of a platform product.

Two subprocesses are nestled within the four main phases. The first is the broad *product release process,* which is performed by the core team and includes all steps from concept generation to training, marketing, and sales. The second is the narrower *feature development process*, which was the focus of this case study. Both subprocesses include distinct steps, milestones, and deliverables, but the feature team development process focuses only on phases 1 and 2. It is more engineering-intensive and, because the products are so feature driven (in the view of several, products are really "collections of features,") it is the process that actually develops the software product.

The feature team development process combines serial and parallel activities. Figure 6.2.3 demonstrates how this occurs at IDe. The planning phase includes two steps that occur in series. First, a feature team details *feature requirements*, then it details the *functional specifications*. In contrast, once the team moves into the implementation phase, series often occur in parallel. For example, the team will begin by laying out *technical specifications,* but will begin *coding* before all the technical specifications are complete. Meanwhile, *QA test planning* occurs almost completely in parallel with coding, beginning just slightly after coding begins. Once the coding and QA test planning are complete, *shakedown* testing begins prior to the launch gate.

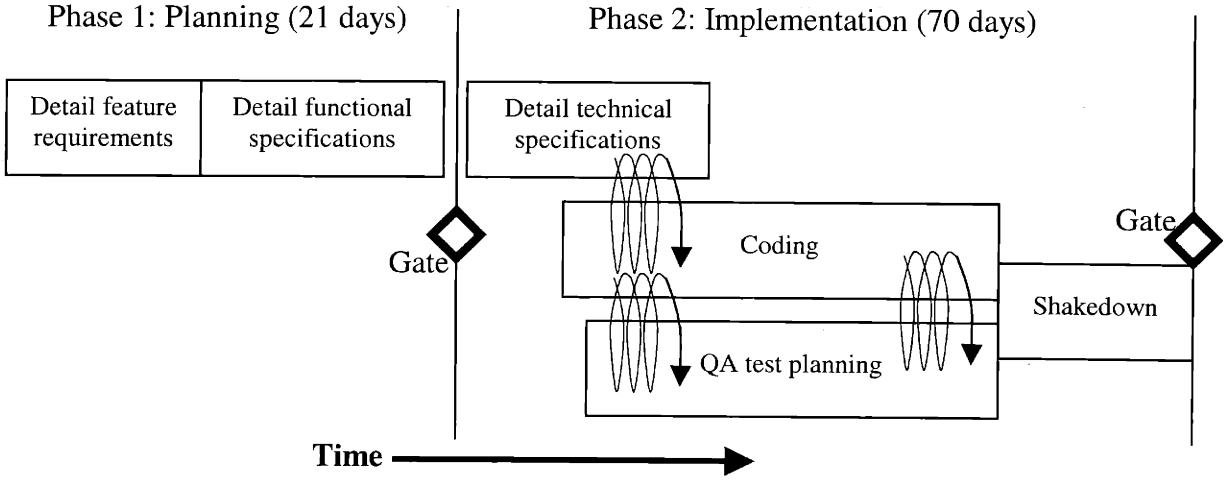


Figure 6.2.3: The IDe feature team process steps (not to scale)

As shown in Figure 6.2.3, IDe divides its requirements and specifications into three different parts: feature requirements, functional specifications, and technical specifications. This three-part compartmentalization allows for high-level requirements and specifications to be frozen early and before coding begins. Technical specifications, however, are considered more flexible, and changes may occur with iterations and feedback loops between coding, testing, and the technical specifications.

PDP implementation

IDe claims to have a special spiral process because of the great flexibility that it expects in its own product development efforts. Several engineers and managers suggested that flexible development was part of the IDe PDP:

> [Our] process really spiral-plus. We use a modified spiral...we always have a release date.
>
> If you're not ready to change the roadmap, it's not a spiral [and] we can change to suit customers.
>
> I plan the number of spirals, but I don't know what [features] I put into each release.

However, the actual IDe PDP reveals that the process iterations are not implemented as broadly as claimed. Three aspects of the IDe PDP demonstrate that the first part of the process is serial, while the latter half the process spirals among the design and testing stages.

The first instance of serial behavior is inclusion of parallel testing and quality assurance only after coding begins. In the words of one software manager, "QA was not involved early on...not until the coding process." As in several other software products, quality need not be of the highest priority because IDe products are rarely mission-critical to their purchasers. Further, QA need not intensively span system level design because most of the bugs are buried in "plug and play" features and are too local to affect system architecture.

The second aspect of mixed serial and spiral behavior is the feedback among successive releases in product families. IDe makes many changes to its products based on customer testing, but often these feedback loops are used in the next generation of product instead of in the immediate product release. As a result, the spiral aspect of the PDP appears only when examining several generations of a product rather than any single product release. Within one product release, the iterations in the process are more indicative of evolutionary delivery because prototypes and customer testing feed back to feature teams that may change product feature offerings, but not system design, based on feedback. Thus, the process' cross-phase iterations only span testing and detailed design.

A third and final demonstration of the quasi-serial nature of IDe development is the freezing of specifications. The freezing of specifications, or at least those that are established during Phase 1 (feature requirements and functional specifications) indicate that although cross-phase iteration occurs in the IDe PDP, iterations and feedback loops are not allowed to reach far back. One manager confirmed the limits of specification change at IDe while giving credit for this control to strong interfaces between feature teams:

> Halfway through the cycle and the specs don't change anymore....cross-functional teams allow you to get to this [point].

Cross-functional core teams are important to IDe because its products are highly feature-driven and feature decisions require inputs from several areas. Feature prioritization and selection process is a critical, though unwritten, part of the company's PDP because the product market is new and because the company lacks market power and is extremely sensitive to customer demands. As a small start-up company, IDe will change its products dramatically in order to keep potential customers, so product customization is a fact of life in the company. One manager gave an example of having to choose 20 of 200 features for the next product release. To rank the features and decide which ones to include, the company divides features into three categories:

A – High priority features that must be included in a product version
B – Medium priority features that should be included but may not
C – Low priority features that are desirable but that may have to wait for the next release

The core team leader ranks the features with the input of the core team and with approval from upper management. Feature prioritization is heavily swayed by customer involvement and beta testing.

Case study conclusions

IDe is a software company that uses an evolutionary delivery PDP to provide fast releases of its product families. The company has evolved to its current PDP after a history of growth but has difficulty characterizing its own process. The PDP includes several

planned cross-phase iterations to incorporate market feedback on product features, but only in the latter part of each development effort.

The cycle times in IDe's market are so fast that the company misleadingly considers itself to be using a spiral process. This is true only if the observer considers an entire product family, with all of its revisions, to be one product. In this sense, the process is spiral because it goes through all the PD phases during each new release. However, according to the more common definition of a product used in this research, the PDP uses evolutionary delivery for each release.

As a company, IDe devotes considerable attention to its PDP, which is especially desirable given that its products are designed to help other companies implement their own PDPs, which are usually stage gate. Many of IDe's unique characteristics, including its size, its rapid product cycle times, its quality requirements and its need to cater to customers, factor into its PDP decisions and have allowed the company to grow significantly since its founding. The company relies heavily on market feedback and iteration, although the decision between whether that feedback and iteration occurs within a product or within a product family is sometimes clouded by the importance of a customer order.

6.3 ITT Industries

Case study synopsis

Company: ITT
Type: Large manufacturing and software company – defense contractor
Products: Military electronics
Cycle time: Slow (years)
PDP: Stage gate with "progressive freezes"
Major risks: Mainly technical or schedule. Although products designed for the military usually have little market uncertainty, market risks are still present here because the military is itself unsure of what capabilities it wants in its radios
Notes: Progressive freeze process increases development flexibility by setting specifications piecemeal and allowing some work and iterations to begin before all specifications are frozen. The process is applied to several programs despite their different risk profiles.

PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
1-2	1-2	3-4	3-4	1-2

Company and product description

ITT Industries is a large engineering, manufacturing, and military contracting company based in White Plains, New York. It employs approximately 38,000 people worldwide and generated sales of $4.7 billion in 2001. One of ITT's four main segments, the Defense Electronics and Services segment, is the focus of this case study. The Defense Electronics and Services segment generates slightly more than a quarter of the company's sales by designing and producing air traffic control systems, jamming devices that protect military aircraft from radar guided weapons, night vision devices, satellite instruments, and tactical radios and communication equipment.

Two ITT products were examined as part of the case study. The first product, a radio and communications system called SUO, is a new military product whose development poses high technical risk. It is being developed by engineers organized on one program team rather than individually by function. The second product, software and equipment for an

updated transmitter on global positioning satellites, is a modification to an existing military product and has high schedule risk. It is being developed by engineers organized in functional teams rather than in one product group. The differences in product type, risk and organization suggest that ITT faces diverse development challenges.

The SUO radio system is being developed as part of the Defense Advanced Research Planning Agency (DARPA) program on military situational awareness. The situational awareness system is exclusively for military use and thus the only market for this product is the US Department of Defense (DOD). The product is still an experimental system with significant technical risk. Major development challenges on this project include technical design work on hardware and network scalablilty.

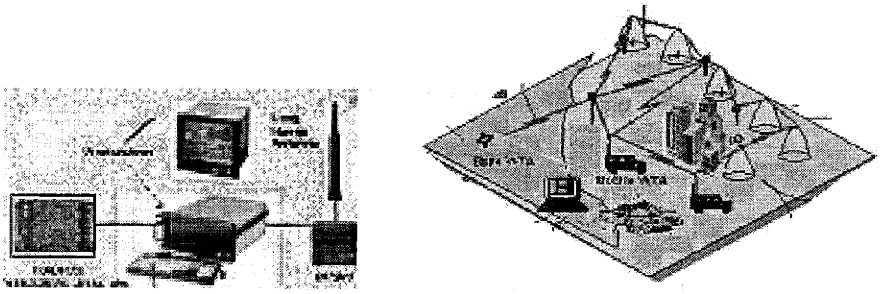


Fig. 6.3.1: SUO sensor communication module and schematic for battlefield transmission

The second ITT product examined was an update to the Global Positioning Satellite (GPS) system. The product will modernize transmitter bands, and add new signals on satellites awaiting launch. Although the GPS system has both military and civilian uses, the contract is exclusively with the DOD. There are no plans for large-scale production of this update because it is likely to be used on only about 12 satellites. The major challenge on this project is the schedule, which is inflexible because of the satellite launch timing.

PD process description

The ITT integrated product development (IPD) process employs stage gates with "progressive freezes." Figure 6.4.2 demonstrates the overall process, which is marked by clear phases and reviews after each phase. Progressive freezes mean that specifications

can be set in a piecewise fashion without delaying the entire program. Subsequent work can start on those requirements or design aspects that are known to be solidly defined and unlikely to change. Progressive freezes resemble the kinds of reviews seen in the overlapping waterfall process shown earlier in Figure 3.4.

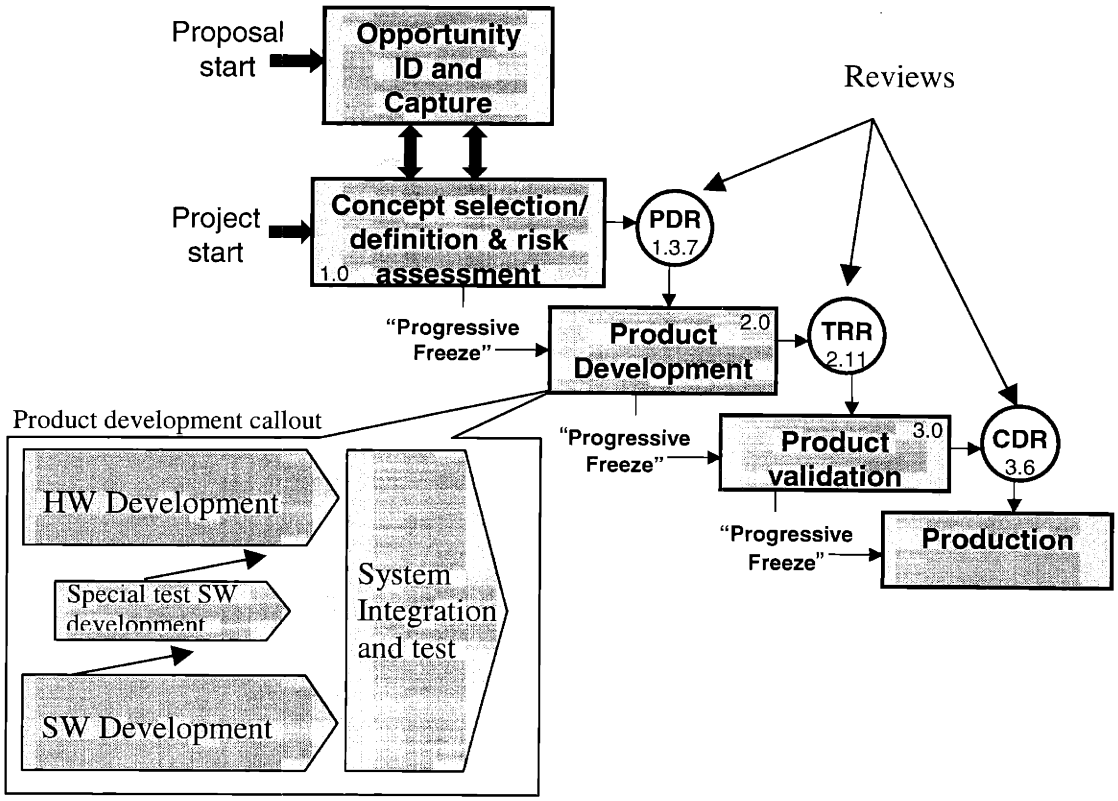


Figure 6.3.2: The ITT IPD process

The ITT PDP is rigidly defined and highly segmented. Each one of the stages in Figure 6.3.2 is itself subdivided into many sub-stages with their own smaller reviews. Iterations are common within each phase or subphase, but cross-phase iterations that span reviews are rare and unplanned. The inclusion of progressive freeze into the PDP is a direct result of the desire to avoid some iterations, especially those that span earlier product specifications, while still allowing some parallel work.

There are several substages within the product development stage, including parallel development of software and hardware for the same product. A separate track is

established for special software used for hardware testing; this separation protects the core software development from being distracted or derailed by the need to write code for testing other components. Integration and testing occurs near the end of the PD stage and culminates with a review and progression to product validation.

The segmented step structure is matched by the rigidity of reviews. Reviews are rigid and have predefined checklists and personnel assignments. For example, preliminary design reviews (PDRs) are divided into separate software PDRs and hardware PDRs. Each review has a list of required attendees, including both internal and independent reviewers. Specific project metrics, such as whether a design may be reused, are required at this review and there is a detailed scoring system for exit criteria. Answers to specific review questions are tallied as

- Yes – all cases (3 points)
- Yes – partial (2 points)
- Yes – minimal (1 point)
- No (explanation required)

It is acknowledged that material presented at reviews may come from documents required earlier, and by previous intra-phase peer reviews.

Risk assessment is a required part the first major stage (box 1 in Figure 6.3.2) and occurs alongside concept selection and definition stage. Each development project must have a "risk register," which is a chart that identifies project risks and categorizes each as technical, cost, or schedule risk. In a reminder of the frequently military nature of ITT's work, market risk is notably not included. For each risk, the register lists:

- The probability of occurrence
- The cost if the risk occurs
- The impact of risk occurrence – both qualitative and quantitative
- The risk mitigation plan
- The risk plan leader

All ITT development programs are defined by individual Integrated Management Plans (IMPs) which serve as PDP roadmaps that a program follows until product launch,

abandonment, or completion. Any variation from the overall company PDP must be defined in the IMP.

PDP implementation in different programs

Despite the appearance of rigidity, the ITT PDP allows for flexible implementation. Most IMPs follow the task flows and sequences suggested by the PDP. However, the PDP itself says process may be tailored to the particular needs of each project* so long as these changes are documented and approved by management in the IMP. As a result, there are some deviations from the official process, such as when a small project involving a minor enhancement skips some tasks and design reviews. Process flexibility is necessary at ITT because the company's numerous product development programs face different challenges.

Small Unit Operations (SUO)

The SUO program is unique in ways that affect PDP application. The SUO program is part of an ITT organizational experiment to move from functional teams to project-oriented teams. Traditionally, ITT employees have been divided into functional groups, which has led to occasional difficulty when one group had little incentive to support another. However, on the SUO development program, the project manager does not need to request time or manpower from a software development manager; rather, he has his own software engineers exclusively assigned to the SUO program and under his direction. As two managers said,

> We want more collaboration between different functions: hardware, software, and systems. We don't want to build walls between them...When the work wasn't 'projectized' we had trouble getting support. It's a lot more civil now. We would have worried more in the old, organizational system.

This organizational difference impacts the PDP because of greater fluidity between disciplines. As a result, the SUO program can more easily iterate between groups of engineers who otherwise would not work together.

---

* Within defined limits. For example, tasks may only be deleted if they are not germane to the project and are not required by contract, or if they were already accomplished on another task or project and the results are pertinent. Only sequential reviews can be combined. Peer reviews may not be eliminated.

Another unique aspect of the SUO program is that the product is itself experimental. Programs supported by DARPA are often in the infant stages of development because of its emphasis on advanced – and thus early – research. As a military contractor, ITT considers the program to be development rather than simply research, because the company has already beat several competitors out of the market during earlier bidding and prototype cycles with DARPA. Nevertheless, the experimental nature of the development work makes a marked difference in tailoring the PDP for the SUO. As the SUO project lead explained,

> This is a learning technology, not a product. Because of that, we could have a lot more flexibility in the hardware.

Thus, physical builds are less important. This is unusual in military contracting, where the customer usually has very specific and rigid requirements. The advanced nature and novelty of this product also leads to greater challenges in setting specifications because, as a PD manager said,

> The biggest challenge is that people expect to be told what to do. They want a high degree of solidity. But here we've got to take functional needs – without requirements – and translate them into engineering terms.

This gives ITT an unusually high (at least in the defense industry) level of discretion in determining customer needs setting its functional and technical specifications. The result is a push to adapt the PDP to be more flexible to account for this high level of uncertainty. Although the market uncertainty is high, the market risk is only average because the market is monopolistic, monopsonistic, and thus relatively uncompetitive and safe. As a result, the major risk on the project comes from the many electronic challenges that ITT faces in developing the ambitious SUO radio.

GPS Satellite

The GPS program contrasts with the SUO effort in three major ways that affect PDP fit. First, like most of the company, the GPS teams are organized by function rather than by project. This restricts early integration – and thus iteration – between different product components.

The second way that GPS program differs from the SUO effort is in the level of risk and product definition. The GPS specifications are based on both a previously existing product and well-established Air Force needs. As a modification of an earlier product, the product specifications were clear and frozen early. Also, the specifications were made easier by the fact that there is no large product line or series of product lines for GPS components. Rather, there will only be 12 units made in the foreseeable future.

Finally, the preeminent GPS risk was schedule risk. Although both the GPS and SUO programs had reduced market risk because of the monopsonistic military customer and post-bidding monopolistic environment, GPS had a more difficult schedule challenge. The GPS space vehicles would only be available for modification until the life of earlier satellites, already in orbit, expired. Any schedule delay that extended the (quite literal) launch date would jeopardize constellation sustainment, so the PDP would have to emphasize schedule when applied to the GPS program.

Common development problems

Despite the differences between SUO and GPS, both programs followed the same PDP with only minor variations. Also, both programs suffered some rework during development.

Both programs applied the PDP through their own IMPs. Program and project managers did not follow every aspect of the PDP ("We don't even read that book" joked one project engineer*) in ways that were both helpful and harmful. Helpful examples of personal discretion occurred when unnecessary documentation was circumvented. Potentially harmful changes from the PDP included the lack of categorization of risks in the IMP risk registers. Also, different sites held to PDP documentation with varying degrees of fidelity, leading to some geographic tension.

* The joke is not as ominous as would appear because the PDP is entrenched whether employees read the "IPD book" or not. Put another way, the documentation and work commonly required of employees constitute many of the criteria expected of them in PDP reviews anyway.

Both programs also had to deal with rework stemming from difficulty in planning iterations, including integrations and prototypes. Sometimes – and ideally – the ITT process generates a prototype near the end of development that requires only a few detailed changes and no system-level changes. However, ITT still faces several examples of iteration and integration problems. The first type is when a prototype does not provide the information necessary to either check functionality or feed back into the process. One GPS manager bemoaned this problem:

> We wasted time on iterations that didn't help. There are big holes in integration. We called [had to cancel] a test of prototype that didn't test what we should be doing.

A second problem is when prototypes are not built because of time or expense. An SUO manager suggested that this occurred on some hardware components:

> We [prototype] for software but not hardware. For hardware it takes a long time to get a breadboard. Building up a board that has enough of the final components...takes a long time. Software you can build up on PCs and other evaluation tools and simulations are available. You can build a skeleton and add things. For hardware, form is a factor. You're putting stuff in and could do it faster, but that would a throw-away effort.

The "throw-away" aspect of physical prototyping apparently sometimes acts as a deterrent to some tests and integrations. This also suggests a schism between hardware and software development.

Finally, at least one experiment in more frequent integrations and the spiral process failed. One manager recalled from another product development attempt,

> We're not extremely successful at using the spiral process. We tried to do things that way and we got an end-to-end flow, but we took so many shortcuts that we had to throw it away...it was a waste of time. To look like we accomplished things, we did simple stuff and didn't retire all of the risks early, but that may have been a matter of experience.

The same manager suggested that this failure may have been a matter of experience rather than process fundamentals; but regardless of cause, ITT does not follow this process and instead pursues less frequent prototypes.

Process experimentation is accepted practice at ITT. The PDP includes a section describing process change, maturation, and evaluation that suggests the use of small pilot projects to test proposed process improvements.

Case study conclusions

ITT uses its "progressive freeze" stage gate process across a wide array of products. The process uses rigid reviews, many narrow iterations, and only a few cross-phase iterations in which lessons from integration and prototyping are applied to early development phases. The progressive freeze allows the process to be slightly more flexible than other stage gate processes.

The ITT PDP exhibits itself differently when used on different projects. On the exploratory SUO program, the process is used to build a technically challenging product for which the customer has only loosely defined and exploratory needs. On the GPS program, the process is used to develop technical modifications on a product with a critically limited timeline. Both projects are military products, so traditional market risks are skewed by the unique aspects of the military contracting.

6.4 Xerox Corporation case study

Case study synopsis

Company: Xerox
Type: Large manufacturing and software company
Products: Copiers and document centers
Cycle time: Moderate
PDP: Hybrid process of spiral and stage gate
Major risks: High levels of market risk are controlled by an emphasis of on-time delivery. Technological risks are managed to a lesser degree by a hybrid process.
Notes: Traditional "Time to Market" (TTM) process was not adequate for software, so a spiral subprocess was implemented for software development. Of note is the inadequacy of a stage gate process for a product with many components being developed at different speeds. Several other PDP experiments ongoing, including the limited introduction of extreme programming.

PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
1-3	0-3	3-4	1-4	1-3

Company and product description

Company and market overview

The Xerox Corporation is a leading hardware, software, and service company that produces document systems, including copiers, printers, scanners and fax machines. The company is based in Stamford, CT, has major campuses in Webster, NY and Palo Alto, CA, and employs approximately 80,000 people worldwide.[11] Xerox has been struggling financially in recent years. Annual revenues – and costs – have hovered around $18-19 billion over the last three years, although only $10 billion of the annual revenues are from actual sales, as opposed to services. Stock prices dropped while the company made an uncertain shift towards developing digital products. During this industry transition, Xerox started changing from "light lens" copying technology to digital copying and

[11] Xerox employed 92,500 people at the end of 2000, but has cut almost 12,000 employees in the past year and a half. (Xerox 2001 Annual Report and New York Times, 1/24/02) Actual Xerox earnings are in question pending an SEC lawsuit against the company over accounting practices.

printing as it tried to weather a potentially disruptive wave of innovation. More recently, the company has created two business models. One focuses on "providing replicable end-to-end document solutions for global customers by industry," while the second "focused on broadening...networks of indirect channels to reach more customers with more plug-and-play products." (Xerox, 2000)

Xerox estimates that the global document market that it serves is nearly $200 billion. This market, in which Xerox holds a dominant share, is segmented into several parts, including a general office market of almost $70 billion and a SOHO (Small office/home office) market of over $40 billion. Xerox operates in a self-described "environment of significant competition, driven by rapid technological advances and the demands of customers to become more efficient." (Xerox 10-K reports, 2000 & 2001). In addition, the company must develop its products in response to government regulations regarding environment, safety, and health, ranging from the Americans with Disability Act to the EPA Energy Star program. (Xerox Newsroom 2001, Xerox CEHSS 2001)

The document market is changing. Customers have been demanding more color and more digital functions. Today, black and white light-lens copiers represent only 30% of company revenues. Xerox is responding to the market changes by developing new products both internally and externally. Internally, Xerox has been developing new series of multipurpose office machines based on its own R&D. Xerox spends almost $1 billion annually on R&D, or about 5.5% of total revenue (or of expenses, which have been nearly equal to revenue).[12] Externally, Xerox has acquired several companies to obtain both products and know-how. An example of this is Xerox' $925 million acquisition of the Color Printing and Imaging Division (CPID) of Tektronix and its solid ink and laser color printers.

[12] The given annual levels of R&D expenses do not include the additional annual investment of nearly $700 million by Fuji Xerox.

Product description

Product development is crucial to Xerox. Its products include several systems and hundreds of parts, including both software and mechanical components. (Otto & Wood, 2001) The development process examined in this case study is that of a Xerox Document Center with a quasi-embedded *Endeavor* software package. One version of a Document Center, the 460-ST, is shown below in Figure 6.4.1. This digital combination black-and-white printer, copier, scanner, and fax machine retails for $32,420 in 2002 and is one of several "ST" models ranging from the 220-ST to the 480-ST.

Key technical specifications for such Xerox document centers include performance (including print speed, copy speed, and warm up time), document handling (including standard input sheet capacity, maximum monthly volume, and inclusion of features such as collators or staplers), and other capabilities (including type of print technology and subsequent image resolution, reduction/enlargement features, and energy use).

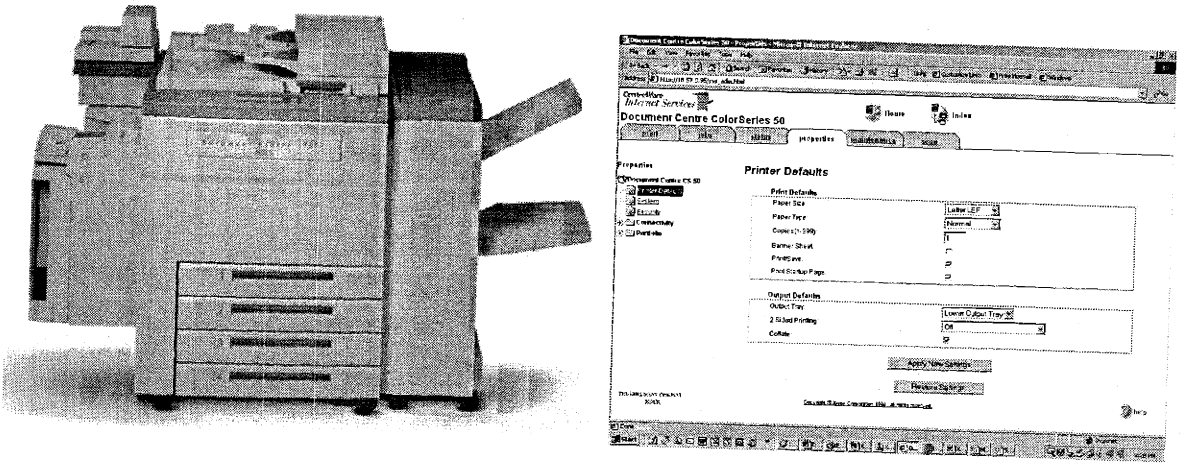


Fig. 6.4.1: A Xerox Document Center 460-ST and a sample user interface screen. This product serves as a copier, printer, fax and scanner. It includes both fully embedded and partially embedded *Endeavor* software for user control.

The mechanical part of a document center includes its body, engine, document handler, and other physical components. The software part is equally complex and is divided into two sections: control software for the mechanical components and the network software for user control. Of these two, the network software is more critical because it allows the document center to perform multiple office functions. This software includes an interface at the machine itself (control panel), as well as features for communicating with

computers and faxes on both internal and external networks. The network software package that handles connections between users and the product is called *Endeavor*. *Endeavor* is platform software that transforms a stand-alone copier into a connected multifunction product, or Document Center. Although *Endeavor* is partially-embedded software in a final product, it can be considered in many ways to be an independent product within the company. It is not an independent commercial product only because, for strategic purposes, Xerox has not tried to market it this way.

Product development process description

Process overview

Earlier literature and discussions suggested that Xerox subscribed to a relatively linear waterfall process. Indeed, Xerox was chosen as a case study in part because it was assumed that the company would be a good representative of a "pure" waterfall process in action. Reality proved otherwise. Although the Xerox PDP is generally characterized as an effective, stage gate process, it is not a pure waterfall process. The case study supported conventional wisdom on the effectiveness of stage gate processes, but also added depth by uncovering several subprocess layers, finding some hidden surprises and even exploring internal process experiments.

The official Xerox product development process is called the company Time-To-Market (TTM) process, part of which is illustrated by Figure 6.4.2.

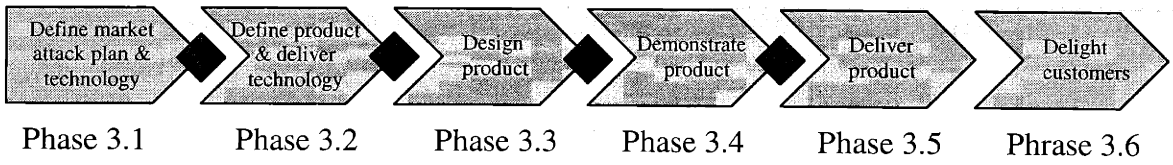


Fig. 6.4.2: Portion of the Xerox TTM process showing phases and gates

TTM is a strikingly linear stage gate process; each phase includes specifically-prescribed inputs, steps, milestones, outputs and mandatory reviews. However, the TTM process is broad and encompasses everything from basic research to post-delivery customer service. TTM's wide span exceeds the scope of this product development research, which focuses

on the center of this process (mainly Phases 3.3-3.5). Nevertheless, a contextual analysis requires a brief explanation of both ends of this process before delving into the center.

Xerox differs on the two extreme ends of its own TTM process. On the far upstream side, Xerox has displayed a string of notorious failures in identifying, funneling, and commercializing promising technologies. Indeed Xerox PARC (Palo Alto Research Center) was an incubator of several ingenious ideas that went unnoticed and were left unpursued by faraway East Coast managers, much to the detriment of the corporation. The most famous of these upstream failures was LISA, the unguarded, Xerox-developed graphical user interface that was functionally appropriated by the Apple Macintosh in the 1980s. On the other hand, the far downstream side, Xerox has made a concerted and successful effort to satisfy its customers and has a broad array of warranties, service programs, and customer support mechanisms.

The focus of this research begins after research concepts have been narrowed or selected for development and ends with product delivery. This core process in Figure 6.4.2 is handled in a rigid stage gate fashion. Each phase in the figure includes specific guiding principles, clear delineations of responsibility, phase transfer criteria or tests, expected input/output lists, and documentation requirements. Phase 3.1 may occur in parallel with other market strategy activities, and Phase 3.6 is considered a "continuous" process, but phases 3.2-3.5 are defined by Xerox as asynchronous processes. For example, Phase 3.3, "Design Product," may not begin until technological readiness is demonstrated and product specifications are defined in phase 3.2. Phase 3.3 includes design and review activities, as well as building, testing, and integration. To pass to the next phase, the project must demonstrate design stability and readiness for production start-up. Once a phase is completed, it is expected that there will be no return. The expectation is that any future changes or features will be introduced into the next version or product. Therefore, design iterations are strictly intra-phase.

Responsibilities within phases are divided between several teams (design, subsystem design, marketing, launch, etc.) and their respective managers. The TTM "decision

team" is a segregated group of people who serve as external reviewers and make go/no-go decisions at phase review meetings. TTM decision teams generally consist of peers who are working on other projects, meet only as necessary, and are kept insulated from other responsibilities regarding the specific product being developed.

Risk management is specifically addressed in Xerox processes and is defined as "a set of continuous activities that identify, track, and control the risks" in a process. Xerox PD process texts usually emphasize technology risks and devote a great deal of attention to technical reviews to ensure the deployability of technologies and the early assessment of risks by design groups. More specific process steps include the categorization of risks into four levels:

1)Very high *2)High* *3)Medium* *4)Low*	based on three elements	*1)Project size* *2)Project volatility* *3)Development environment volatility*

Other types of risks, such as schedule slip, market fit, and even exposure to financial risk from changes in foreign currency exchange rates, are discussed more extensively in other Xerox documents.

The software subprocess

TTM contains several subproccesses, including one for software. The software subprocess is notable for two reasons. First, it marks a recognized difference between hardware and software development. Second, it makes a slight break from the standard waterfall process prescribed by TTM. The software subprocess (first developed in 1998 with the help of Barry Boehm, and still bearing marks of his influence) appears at first to follow a stage gate pattern:

1) Inception **2) Elaboration** **3) Construction** **4) Transition** **5) Delivery & Maintenance**

The software subprocess is also marked with "anchor points," or milestones, so that the software can remain matched to other parts of the product being developed via the regular TTM process.

Despite the initial stage-like appearance of the software subprocess, a diagram of the steps indicates that major cross-functional iterations occur, in contrast to the standard TTM process. An example of this can be seen in Figure 6.4.3, which reflects the spiral process.

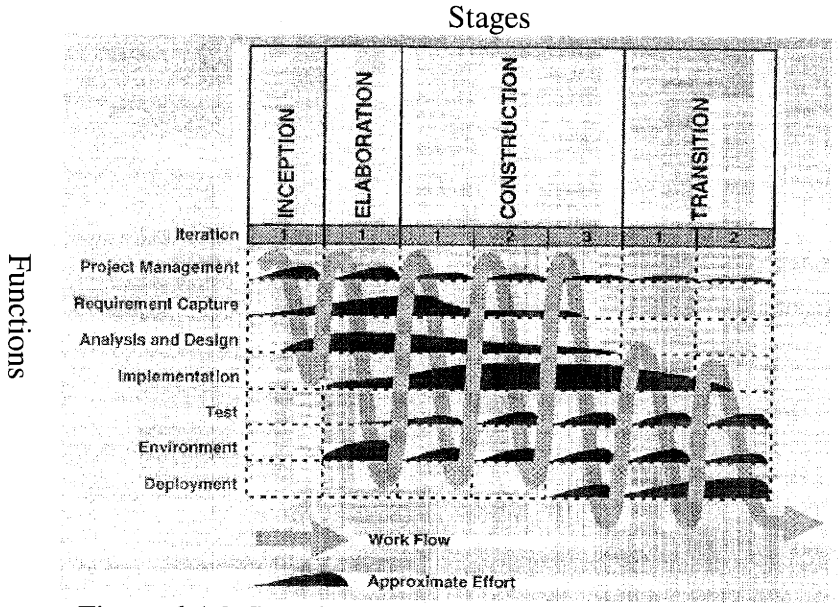


Figure 6.4.3: Iterations in the Xerox software subprocess

As prescribed the spiral process, this software subprocess includes planned, cross-phase iterations – if the functions along the vertical axis are considered phases instead of functions. These frequent iterations help to avert "silo" or "throw it over the wall" organizational difficulties, and also prevent the early freezing of specifications. However, Figure 6.4.3 does not represent a pure spiral process, as defined in Chapter 4. The software subprocess is not as flexible as a complete spiral process because it must still conform to the more regimented TTM process. Every stage has a series of prescribed transfer criteria even though rigid transfers preclude truly simultaneous work.

PDP implementation

The TTM process and software subprocess described above reflect official Xerox policy, but do not comprehensively describe what actually happens in Xerox PD. Although the processes are usually followed diligently, there are some exceptions and surprises in the actual working environment.

The most notable difference between prescribed and practiced policy is that Xerox allows for process experimentation. For example, one group of software engineers received a waiver and is being allowed to develop their code by using extreme programming (XP), an entirely different process described earlier in Chapter 3. It is evident that Xerox is having difficulty forcing these experimental processes to conform to TTM. As one of the XP practitioners states:

> "There's idealistic XP and there's what we do here. Generally XP is done with small groups of people, maybe 10 people. If you did true XP fashion, you [couldn't] fit XP under the [TTM] hood. The problem with XP in this environment is traceablity. TMM compliance requires a lot of traceability...traditionally XP doesn't have that. So we're working on making XP traceable."

This is one of several examples of internal process tensions. Another case in point gets to the root of the tension:

> "Management tends to like to have all their planning up front, all...which is very anti-XP. XP says just figure out enough and get going...and we'll figure out the rest later. Management freaks out about this. They want everything up front. They want to know when the schedule is going to be ready. So traditionally management gets this but they tend to get...all the schedules wrong, they tend to not be correct and tend to have a lot more features and functionality in there."

Mismatches of this kind are not indicative of process failure; it is too early to tell whether the XP experiment will work. Nor are process mismatches unique to experimental processes.

Another prevalent mismatch is that of different "natural" cycle times between software and hardware components of the same final product. Some of these alignment issues are not specifically addressed in TTM, but make their presence felt in almost every product. In the words of one senior hardware manager,

> "[The software schedule] contrasts with the engine. We may not be able to do it...the *cadence* is different."

This issue of cadence was noted with concern among software managers as well. The difference in timing may also be due to asymmetrical quality standards for hardware and software. As one manager stated:

> "We have different criteria for hardware and software problems in our quality assurance cycle. Hardware problems, when they exist, usually exhibit themselves in a more critical way to enabling a product launch. The criteria would be its usability...having [the hardware] jamming all the

time...would be a bigger problem than having a software-related [glitch, or a local user interface] that just says the wrong thing. "

Another software manager expressed her frustration with the hardware-oriented, TTM quality requirements being foisted upon the non-critical software:

"They try to make [our software] perfect and that's why it takes them so long to get things out the door...they over-engineer, over-think, and pay too much attention to quality."

Another important discrepancy between official process and actual occurrence is the prevalence of testing. Although TTM distinguishes between "designing" and "demonstrating" a product, testing occurs much earlier in software design than would be expected from the prescribed process alone. Unit and software testers frequently become involved as early as stage 3.2. As one testing engineer pointed out, the early involvement exists,

"not to test, but to develop a good test...[and plan] what a good testing strategy would be."

Later, the more common beta-testing is nearly universal. Receiving feedback is such an important component of marketing that nearly everything is tested for 6-8 weeks. Usually companies receive a preliminary version of a product for free during that time in exchange for feedback regarding the product's use and performance.

Another hidden tenet of Xerox product development – one that would not be discernable from the company TTM alone – is a strong emphasis on schedule. There is a corporate culture of on-schedule delivery, at the expense of almost anything else if necessary. This stems directly from a perception that market risk must be the most controlled of all risks. The unity of this sentiment is remarkable, as can be seen from the unusually strong chorus of reinforcing statements from a software engineering manager, a marketing manager, and a hardware engineering manager:

"You won't be able to talk me into adding [any features if it will push out the schedule.] Even if my boss asks for a change, if we say no because we can't move the schedule, that's it. Nobody argues. That's the extent of the culture because it's clearly schedule-driven."

"It's important at Xerox if you control the schedule. Being in this market, it's a commodity and you've got to be on time or you lose the market."

"Most of our incentives are to launch on time, so the whole culture is that way."

> "[Changing] a schedule...ugh...there really has to be a shift in the marketplace to do that...like someone has leapfrogged you and you would be at such a disadvantage that it wouldn't be viable from a business standpoint to launch a product. It would have to be pretty severe to [delay a schedule just to] get a feature."
>
> "Schedule is king."

Indeed, in almost any contest between extra features (technical) or on-time delivery (schedule/market), the schedule will almost invariably win at Xerox. Although not explicitly emphasized in the TTM process, managing schedule risk becomes functionally unnecessary because schedule risk is not accepted.

A final important aspect of Xerox PD is the difference between "variant" and "clean sheet" products. Most Xerox products are contain less than 75% newly-designed parts and are therefore classified as variants. The company only develops clean sheet products with the expectation that they will become platforms for future variants.

The TTM process does allow some flexibility in developing variant products because the core components of variant products are relatively well-known. Technically, Xerox goes through the same process for both variant and clean sheet projects, but variant projects have considerably shorter cycle times than clean sheet projects (one year instead of up to 4-7 years) because some TTM steps can be skipped for variant projects. Variants, especially those with simple "plug and play" feature additions or enhancements, have other advantages as well. They tend to require only half the number of prototypes as clean sheets. Additionally, the prototypes that are built tend to cost just 25% to 33% of the cost of clean sheet prototypes. The incentives to develop a variant product remain strong. As one marketer put it:

> "In past years we'd try to clean sheet everything...now we're cleansheeting one component at a time. So when you ask about variants, I could make the case that almost everything we do from now on is a variant."

Case study conclusions

Xerox follows a rigid, stage gate PD process and defines specifications early. However, the company makes several exceptions to its stage gate Time to Market (TTM) process. First, product testing sometimes overlaps with other functions earlier than officially prescribed. Second, software development is permitted a more cross-functional and

iterative subprocess. Finally, process experimentation occurs, as in the case of control software programmers who received a waiver from the ordinary PD methods in order to prototype Extreme Programming (XP) within the company. Xerox has some difficulty integrating these three iterative and experimental subprocesses into TTM. The company recognizes that some external software development processes may be effective, but Xerox is no Microsoft and adopting more flexible development processes into its hardware-oriented TTM remains a challenge.

The company's imperfect inclusion of flexible processes means that Xerox still consciously emphasizes predictability over flexibility. The mere hint of variable specifications leads to instant skepticism and managerial fear of "doom loops," a cleverly derogatory reference to iterations between design and specifications that threaten to never end.

The stage gate TTM process appears to be effective for Xerox, so why does the company struggle so painfully to integrate less rigid subprocesses? After all, the Xerox corporate emphasis on-schedule delivery mitigates the threat of slow development, an otherwise common problem in stage gate processes. Also, Xerox has adopted a product line of several modular platforms and many "featured" variants, a system which favors stage gate processes. Finally, the company's on-time delivery and intensive customer feedback ensures that any necessary changes can be included in the next version of a product.

Xerox tries to infuse more iterative and flexible subprocesses into TTM because it still faces challenges despite the advantages and seemingly good "fit" of the stage gate process. In the digital age, Xerox recognized that its new software components sometimes had faster development cadences and lower quality requirements than its hardware components. The TTM process had – and still has – difficulty managing these differences. Also, Xerox' immunity to late delivery is not absolute. Even with cultural forces outside the PD process urging timeliness, market uncertainty creates overwhelming pressure for more rapid development. Finally, even Xerox sometimes requires development flexibility in a *product* rather than in a *product line*. What the

company sometimes calls "iteration" is in fact a series of alternations or additions that occur so slowly that changes are allowed to wait until the next version of a product.

Xerox's unique risk profile and mix of processes provide some interesting insights into the application and selection of PD processes. In this case, industry context led to an array of development risks that the company attempts to manage with different PDPs. Xerox manages schedule risk well and keeps development programs on time, but is still struggling with its hybrid process.

6.5 Printco

Case study synopsis

Company: Printco
Type: Medium size manufacturing company
Products: Case coders and ink jet markers
Cycle time: Moderate
PDP: Aspiring stage gate
Major risks: Technical and market risk in new marker program; market risk in variant coder program
Notes: Difficulty in incorporating early feedback from integrations and marketing
PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
1-2	1-2	1	4	1-2

Company and product description

Printco Inc, a medium-sized company of 1500 employees, develops and produces coding, labels, and markers for shipping and inventory. Printco is a pseudonym for a US-based company that provided data and interviews for this study, but which chooses to remain anonymous out of concern for trade secrets. The name of the company and the trade name of its products have been changed in this report.

Most Printco products are used in the food packaging industry. Although its products are capable of marking and labeling many varieties of plastic and paper substrates, the most common are bags and boxes used to ship food. In some cases, products are sold directly to end user food companies such as Frito Lay. In other cases, products are sold to integrator companies that build packaging equipment for others. The company is expanding to other, non-food industries.

This case study examined the development processes of two Printco products that have recently been introduced to market: the 26K Series Ink Jet Marker and the Model P220/440 Case Coder. The first product, the 26K Series Marker, is a digital marker

designed to mark printed circuit boards (PCBs) with white ink. This was a new product for Printco; there was no previous incarnation of the product. The major risks in product development were technical because of the challenge of ensuring precise line definition and permanency. The 26K Series includes high-resolution Drop on Demand (DOD) jetting assemblies, robust printhead and controls architecture, specially formulated inks, and designer software.

The second product, the P220/440 Case Coder, prints scannable bar codes, detailed logos, or product information on various cases, as shown in Figure 6.5.1. It can print on most substrates commonly used in case and carton packaging, including coated and uncoated cardboard, rigid plastics, and shrink wrap. The P220 has two print heads, while the P440 includes four print heads for printing on both sides of a box. Both are variants based on Printco's older Model 200 product family, which served the same market but printed at slower speeds.

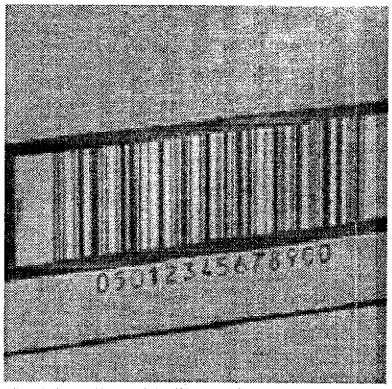


Figure 6.5.1: A universal product code printed on a carton with a P220 Case Coder

PD process description

Process overview

Printco is currently revamping its PDP, continuing its evolution. As recently as the early 1990s, Printco's process was relatively unstructured and typical of the company's smaller, entrepreneurial status. During the past decade, the company nearly doubled in size and launched a formal PDP to ensure ISO certification. It also introduced standard

financial measures and additional reliability testing to avoid relying on end users (and warranties) to validate product quality.

Printco's personnel organization is key to understanding its product development. Most development programs include about 10 engineers from three technical teams: chemical, mechanical, and electronic/software. The chemical team is primarily responsible for designing the inks to be used in the printers, and must address such issues as drop size, resolution, adherence to substrates, and drying speed. The mechanical team is responsible for designing the printheads and mechanical body of the machines. The electronic/software team designs the software necessary for customers to control printing. Teams must work together in overlapping areas, such as avoiding potential trouble when ink jams a printhead. The joint efforts of chemical and mechanical engineers are needed to correct such problems. One project manager is responsible for the technical development of the product, including coordinating the technical teams. A separate program manager has overall responsibility for the product, including control over the relationship between the project engineering and marketing.

Printco's official process employs a series of phases, stages, and reviews, as shown in Figure 6.5.2. The first phase, sanctioning, includes concept design and endorsement of the development effort by senior management. The second phase, design, incorporates occasional reviews, although the design reviews are described loosely and with no set schedules. Individual program and product managers may also freely interpret Printco's flexible provisions for controlling design changes and determining which proposed changes are accepted into a final product. Integration and prototyping occurs near the end of the design phase, and products are tested both internally and at customer sites. The final phase, production, occurs after most design and development work, and includes manufacturing and installation of final products.

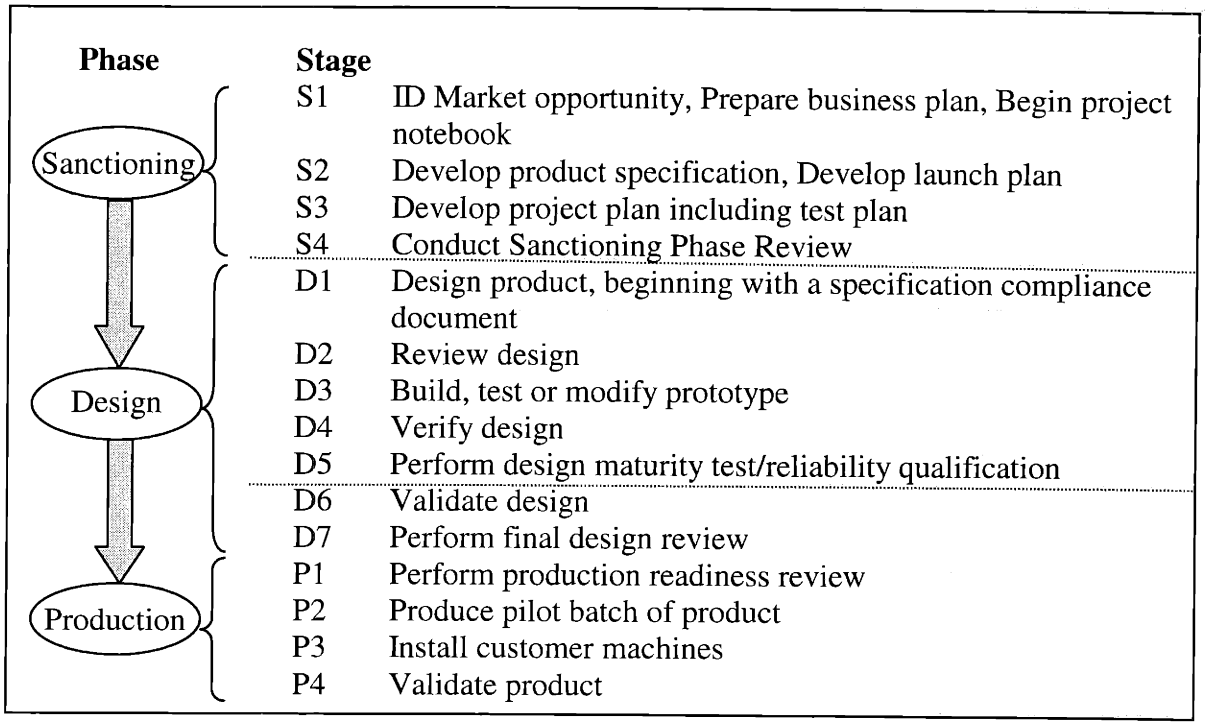


Figure 6.5.2: A diagram of the Printco product development process

The company's recent products have been deemed successful by both Printco and its customers, yet their development schedules and budgets have regularly exceeded projections. The company is working to improve both process and implementation.

PDP implementation

The actual Printco PDP is not as well-defined or implemented as Figure 6.5.3 might suggest. The process may have been rigidly followed at first, but implementation has drifted and evolved informally over the years. Sometimes straying from the process was advantageous, as when people used an informal process to circumvent a flaw in the official PDP. In these cases, the written processes merely were not updated to reflect the effective practices already undertaken by the company. Other times, straying from the process was harmful, as in the case of one recent product that took nearly double the predicted development time (at an extra expense of almost $1 million) because of design changes throughout development.

Interviews with a series of Printco engineers and managers (see interview guide in Appendix A) revealed two interesting points about how the company actually implemented its own PDPs. First, the case study demonstrated that the process did not work equally well for different product portfolios, at least in part because those products faced different risks. Second, neither iterations nor reviews were well controlled, leading to difficulty in meeting development goals.

The two products examined, the Model P220/440 Case Coder and 26K Series Marker, were specifically selected in order to compare disparate development efforts. The Case Coder was a derivative product for which market risk dominated over technical risk. The Case Coder market was thought to be well understood, but the process did not incorporate market feedback early into the process and market risks dominated because of this process failure. In the typical sentiments of one engineer,

> We should have had more steps earlier on...we needed to get the information earlier. Marketing first said we don't need [a feature] anymore, so we didn't design them in. One year later, and two months before release, it changed to 'Yes, we want [the feature]!'

In this specific case, some market feedback was incorporated late in development because the PDP did not mandate engineering inclusion of marketing input during early stages. The company was lulled into thinking that it understood customer needs, which resulted in abrupt scope expansion when added features became necessary late in the development effort, increasing expenses and delaying overall development. This was not the gradual scope creep found in other case studies, but rather a more sudden realization of market-driven additions necessary in the latter stages of development.

This example also challenged conventional wisdom because the primary risk was market risk and not technical risk. Risk is usually low in derivative development projects, and whatever risk exists is usually dominated by technical risk because of the incremental nature of development. Past experience in a known market usually prevents market risk from dominating, but in this case past experience led to false assurance, which combined with process failure to result in a poor assessment of market needs.

In contrast, the 26K Series Marker was a riskier program with both more market and technical risk. The 26K Series Marker faced great technical risk because it was an entirely new technology (the printhead, ink, and intended substrates were all new). It also faced significant market risk because the printed circuit board market was new to Printco, which was trying to expand to areas besides food and case printing. Despite these differences in risk profile, the same PDP was used, although it was implemented with more structure than in the Case Coder.

The result of the 26K Series Marker development effort was a technical success, but the product is still so new that it has yet to prove itself in the market. Development went over time and over budget, but for different reasons than the Case Coder. As implemented, the PDP led to several late market adjustments for the Marker, but none of the market-induced changes were as surprising as those that were initially neglected on the Case Coder. Iterations between customer needs and engineering designs were not abrasive, although it is unclear whether this is due to PDP guidelines or due to good personal interaction among project and program managers. Most delay was due to technical, rather than market-driven, difficulties in designing an ink-printhead combination that could achieve the necessary quality specifications.

The Printco PDP did not lend company-wide uniformity to these two separate PD efforts, which faced significantly different risk profiles. One of the reasons for this discrepancy is that the PDP does not adequately assign or control reviews or iterations, including integrations and prototypes.

Design reviews are only loosely defined in the Printco PDP, so they vary dramatically between projects. Although reviews were casually described as "not rigid at all" and "ad hoc," reviews occur frequently. The most important gates are the sanctioning review and final design review, but even these formal gates are implemented loosely. As one manager stated,

> You don't stop the work waiting for the sanctioning. That's not the hard gate one might expect. Not all things are reviewed in same way. It's not random, but the process doesn't clearly define what review should be.

The standards are more stringent in final design reviews, but the tension is absent because of several informal design reviews that have already occurred. This reduces some formal design reviews to mere formalities. As one engineer claimed, "by the time you have a final design review, the designs are clean." In such a case, a review is reduced to a formality, while informal reviews gain in importance. Engineers themselves determine when informal reviews occur and who is on their own review committee. Conditional reviews, where some parts of a design are provisionally vetted despite the need to make additional changes, occur frequently.

Iterations, design changes, and prototypes are also considered but not controlled by the Printco PDP. This leads to inconsistent standards for determining when design changes or specification changes should be approved. The following three examples demonstrate how changes may be either accepted or rejected: a specification can change, a design can change, or a change may be rejected. In the first example (specification change), one aspect of a printhead design did not meet a temperature specification. An internal review determined redesign was unnecessary because the specification had been too stringent. The specification was changed to correspond to the design with no quality penalty. In the second example (design change), a specification changed and engineers had to scramble – unhappily – to redesign because of this "scope creep":

> As we investigated this brand new marketplace we discovered a new thing or feature to be added. Each would cost a week or calendar month. Initially we planned to include one of those features only in a later release, but we had to do it immediately. That was a two week hit.

> Keeping designs open tends to frustrate everyone. The definition of the product has a tendency to stay liquid too long. The design isn't fixed and changes happen that cause us to go longer than the planned schedule. We need more voice of the customer earlier and we need the description to freeze design. Everyone wants the spec harder and earlier."

In a third example (rejected change), a market-driven change proposal threatened, but did not change, the software system architecture. As two employees described,

> When [a manager] proposed a change [that would affect system architecture], we demanded a majorly good reason to do so. We pounded on the table and used harsh language. I told them that

> you'll have to kill me and get a new team leader to make this change...changing a spec is something you don't want to do very late unless you have a compelling reason to do so.

The proposed change was dropped to avoid the major rework that would be required otherwise. These preceding three examples show how different situations can lead to different decisions on change control, but all three of these decisions were made without specific guidance from the PDP. Decisions were made on a case-by-case basis by the individuals working on each project.

This section and series of observations demonstrated that the actual Printco PDP is a conglomeration of both official process and unofficial implementation. These inconsistencies between official and actual PDPs are not necessarily helpful or harmful. Sometimes the unofficial process is an improvement over the written PDP, while other times it creates delays and rework.

Printco PDP challenges

In revamping its PDP, Printco faces a major challenge in controlling its development iterations, including the information flows, integration of different teams' work, and prototyping. As in other case studies, hardware and software groups sometimes develop their product components at different cadences. Mechanical engineers frequently require bits of code simply to be able to test mechanical components. Software engineers then design test code that is later discarded

> Sometimes a design without code can't be tested, so the mechanical folks need some quick control software. So [the software engineers] do some garbage code so you can try out the design...we're always waiting for the software guys to finish the design. (Mechanical engineer)

> We would write some junk code or stubby tool to check out mechanics components...especially fluidics. Some stubby thing would fire off the solenoids and record results or troubleshoot timing, but the impact of this development jitter on mechanical design was minimal. (Software engineer)

There is a natural tension between teams regarding whether the software lags are in fact minimal or whether the development "jitter" leads to significant delay, but the example demonstrates the existence of interactions between groups that require management throughout development. Not all interaction and information flows are "jitters." Some

cross-phase iterations involve late feedback from prototypes and integration in the latter part of the development process.

One example of large-scale iterations are design changes that result from prototype testing. Printco develops its prototypes relatively late in the development process. As one manager described the history of prototyping in the company:

> In the long term less prototyping won. Our prototypes are closer to what the final product will be. It used to be easy to make a prototype, but with electronics it's more costly, so we try to do more first.

This late prototyping sometimes leads to unintended iterations in the form of rework and schedule delays. In one example, a manager lamented the late discovery of a technical problem that only arose during the integration of parts and testing of a prototype:

> We found [the problem] late. The signs of [the problem] – a major 'gotcha' – were there, but they were isolated and random. We needed a bigger population to find it [prior to the prototype revelation.] We scrambled to redesign because of that.

The current Printco PDP does not advance the cause of early prototyping because of the perceived complexity of design. Future versions of the PDP may either freeze specifications earlier, and thus disallow late changes, or may allow more flexible specifications to allow for improved feedback from later stages of development.

Printco PDP conclusions

The Printco PDP is a combination of official process and broad individual discretion by individual managers and engineers. This combination of official and informal PDPs resembles a waterfall process with frequent and flexible reviews, moderately-controlled intra-phase iterations and poorly controlled cross-phase iterations that result in occasional rework and delay. The company has thus far maintained a strong product line, but seeks to continue improving its PDP.

Printco's PDP developed from less official processes to incorporate common milestones and improved quality control, but struggles to control products facing different levels and types of risk. The PDP does not work equally well for different projects, partially because of the different risks faced by those products and partially because of the team

dynamics among those working on each product. A common frustration throughout the company is the tendency for redesign when early specifications are considered frozen, but then change with some late-entering market data. A lesser frustration is redesign stemming from technical problems that are revealed only in late tests and prototypes.

Lessons from Printco include the need for a PDP to incorporate market and technical feedback earlier. Uncontrolled changes – from the "jitter" of software and mechanical cadences to the "last-minute" inclusion of market data and major design revisions – evidently leads to rework, added costs and extended schedules. Specifically, the Printco case study illustrates that a process of frequent but loose reviews is not conducive to design change control unless it is supported by a system of controlling cross-phase iterations that allows designers to incorporate information gained from the traditionally late stages of testing, integration and prototyping. The case study also demonstrates that, as in other companies, variant projects may be less risky and less expensive than new platform projects, but that even "derivative" markets that are thought to be well-understood can be risky.

6.6 Secondary case studies

6.6.1 United Technologies Corporation (Pratt & Whitney and Sikorsky)

Case study synopsis

Company: UTC
Type: Large manufacturing company
Products: Jet engines (P&W) and helicopters (Sikorsky)
Cycle time: Slow
PDP: P&W: Stage gate with parallel development
Sikorsky: Stage gate with unused loops
Major risks: P&W: Technical risks dominate as thrust, quality control, and efficiency are the key market criteria
Sikorsky: Most sales are military and military specifications are common, so market uncertainty is extremely low. Technical uncertainty dominates.
Notes: P&W: Products technically similar to SWPG and many of the engineering design skills are nearly identical.
Siksorsky: Unused inter-phase loops stem from feedbacks that supposedly span FAA approval, but do not in fact iterate across that milestone.

PDP Metrics:

	Iteration			Review	
	Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
Sikorsky	1	1	2	1	1
P&W	2	4	2	1	1

Company and product description

United Technologies Corporation (UTC) is a large and diverse manufacturing company. Its businesses include Pratt & Whitney (P&W), a turbine manufacturer, Sikorsky helicopters, Otis elevators and escalators, Carrier heating and ventilation, and several other units. The Connecticut-based company employs over 30,000 people worldwide and generates revenues of almost $28 billion per year. This investigation uses Pratt & Whitney and Sikorsky as PDP examples.

Pratt & Whitney (P&W) designs and manufactures gas turbines, mainly for commercial and military aviation, but also for power generation. P&W generated sales of almost $8

billion in 2001, slightly over one quarter of UTC's overall sales. The company's engines are similar to the turbomachinery designed and manufactured by SWPG described in Chapters 2 and 6.1. Figure 6.6.1 shows a sample P&W engine, a 112-inch-fan PW4000. This engine is an ultra-high-thrust model covering the 74,000 to 98,000-pound-thrust class to meet the current requirements for the Boeing 777 twinjet. Like SWPG turbines, the machines are large but include thousands of components with precise and small clearances.

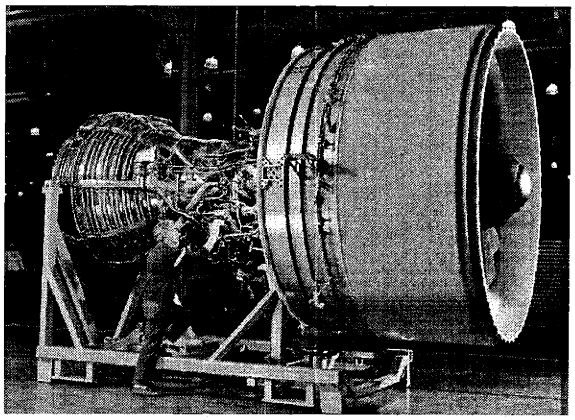


Figure 6.6.1: (left) The PW4000 engine
Figure 6.6.2: (right) Sikorsky S-76 commercial helicopter

Sikorsky Aircraft Corporation is a smaller subsidiary of UTC, generating $1.8 billion in revenue in 2001. The company designs and manufactures military and civilian helicopters. One example can be seen in Figure 6.6.2, which shows a $7 million S-76 commercial helicopter that can transport 12 people.

PD process summaries

P&W and Sikorsky each have their own PDPs. Each is a different version of the stage-gate process. The Sikorsky PDP is the more traditional stage gate process, as can be seen in Figure 6.6.3. Development progresses through a regular series of concept, system design, detailed design, and system testing stages. Many specifications and reviews are required by either military or FAA standards. These standards lead to a series of rigid reviews that occur at the ends of stages rather than at regular intervals. Stages are

sequential, although there are some minor iterations that occur between detailed design and testing/refinement.

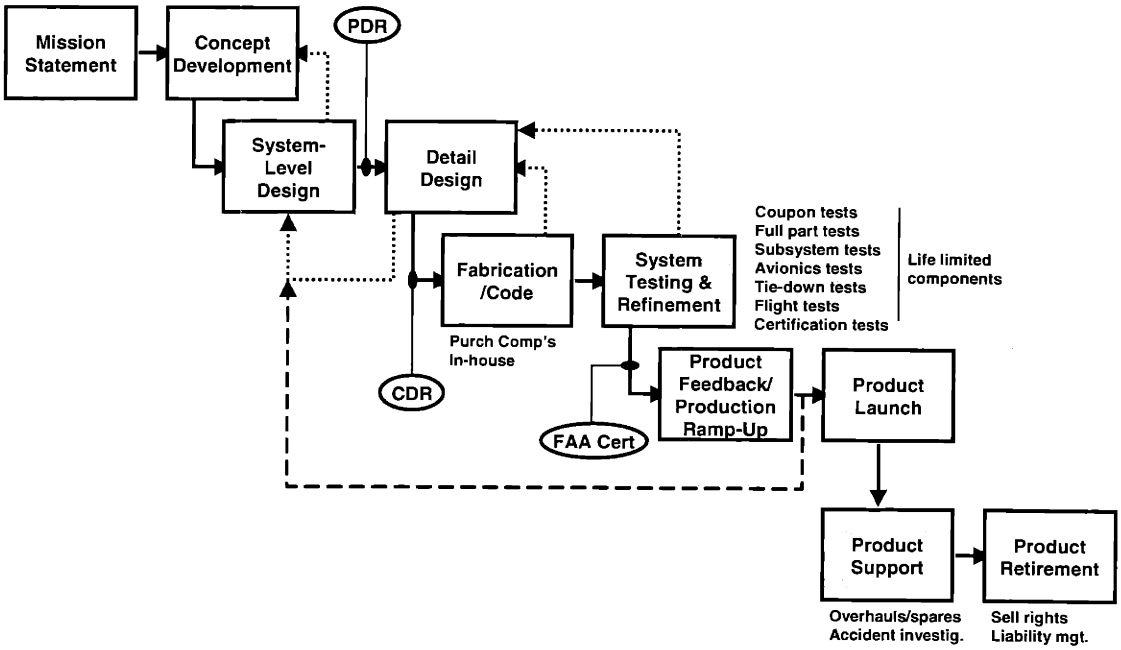


Figure 6.6.3: The Sikorsky PDP includes unused loops

The official Sikorsky process cites iterations that are rarely enacted, thus remaining a strict stage gate process in practice. Figure 6.6.3 reflects the official PDP, which misleads by showing so many feedbacks. In practice, the process is more sequential. For example, the feedback loop that occurs after the FAA certification is rarely used. The feedback and knowledge from the total product most goes towards improving future products, but rarely the product being developed at the time. Feedback to the same product would force the need to repeat certification procedures.

In contrast, the P&W process includes several uncited iterations. The P&W process is also a stage gate process, although not as rigid as that of either Sikorsky or SWPG. In most ways, the P&W process is sequential because the product must also meet rigid FAA criteria. However, CAD prototyping has reached a level of fidelity that allows the company to reach forward and make several early digital prototypes that provide feedback to the system-level design phase. These prototyping iterations, although not

officially included in high-level P&W process descriptions, make the P&W stage gate more flexible.

Before explaining the iterations of the P&W process, which are not core to the process but remain important exceptions, it is helpful to explain how FAA requirements reinforce both Sikorsky and P&W's use of stage gate processes. FAA criteria are often not merely standards applied to the end result or final product. Rather, some requirements-based criteria are actually entire production life-cycle frameworks, requiring companies to undergo a series of stages and reviews. For example, one FAA certification framework (DO178B for certifying safety-critical and embedded engine controller systems used on P&W 6000 engines) describes a software life cycle that includes system definition, software planning, software requirements, software design, code, integration, verification, configuration management, and quality control. This rigidity can be even more intense on physical, rather than software, designs. Such criteria limit the flexibility of PDPs in order to ensure predictable and reliable safety systems.

P&W employs some cross-phase iteration across its stages. Figure 6.6.4 shows the P&W integrated product development (IPD) phases as officially defined by the official process, followed by stages as actually used and defined by Hague (2000). Hague describes how the company cycles through several simulation and modeling iterations as it designs aircraft engines, even though these iterations are not recognized by most of the PD organization. Using a parameter-based DSM, he shows how requirements are revisited and simulations are included during each of four PDP stages: concept design, preliminary design, detailed design, and validation. The figure shows these iterations as a broad spiral with one cycle in each phase.

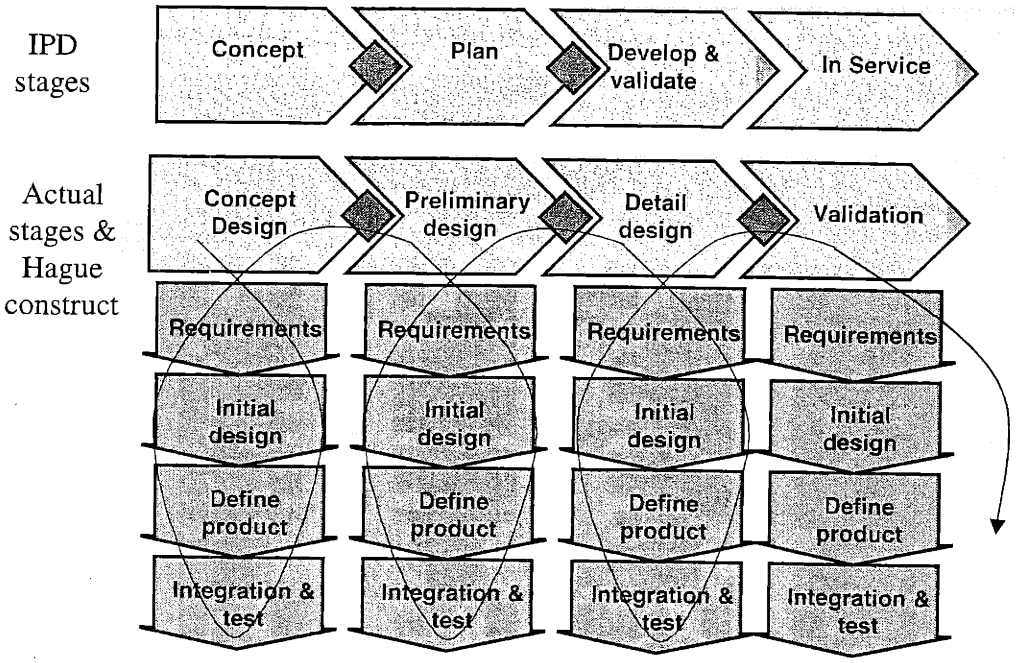


Figure 6.6.4: P&W Integrated Product Development (IPD) process and an interpretation of how it is implemented

The iterations shown in Figure 6.6.4 are idealistic and not as comprehensive as they appear. Some requirement "cycles," for example, are merely based on the assumptions of the previous stage and do not constitute actual feedbacks. Further, most specifications are frozen during the later phases, so some revisitations of requirements are merely cursory checks. Finally, the Hague interpretation of the spiral process is broad; iterations are allowed to entire products, with subsequent loops being the next generation in a product family. Thus, although the DSM data does support the finding of some cross-phase iteration in the form of simulations and modeling cycles, they do not overwhelm the overall stage-gate nature of the P&W process.

Both P&W and Sikorsky design detailed components in parallel, thus forming multiple branches within each of their stage gate processes. Although true integration and testing must often wait until near the end of processes, improved computer simulations are allowing for some improved early testing of both components and integrated product modules.

UTC Case study conclusions

Both of the UTC divisions that were studied, Sikorsky and P&W, employ stage gate processes and face high technical and quality control risks. Both develop products where many requirements are frozen early, in part because of FAA regulations and military specifications, and in which most integration and testing is limited to computer simulations. The Sikorsky process is a strict stage gate. Although its plans include some inter-phase iterations, such iterations are rarely used. The P&W process does not officially acknowledge significant inter-phase iterations, although earlier DSM analysis shows some feedback – and thus iterations – between parts, requirements, and groups from different development stages. Although the process remains a stage-gate, the introduction of interphase iterations makes the P&W process more flexible.

6.6.2 Ford Motor Company

Case study synopsis

Company: Ford
Type: Large manufacturing company
Products: Automobiles
Cycle time: Slow
PDP: Stage gate with feedback loops
Major risks: Market uncertainty dominates the auto industry as competitors try to fill specific market niches with each product.
Notes: The Ford product development system (FPDS) is a well-documented process that is applied with varying levels of success to all products, regardless of their level of independence from previous generations.
PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
2	3	3	1	1

Company and product description

The Ford Motor Company is the largest of the case study companies. It employs almost 165,000 people in the US alone, generates annual sales of approximately $135 billion, and spends about $7 billion per year on R&D. The primary products are automobiles and

trucks, which are possibly the most complex products that most people individually own. Ford automobiles have hundreds of subsystems and over 20,000 parts, many of which are made by a complex network of suppliers.

Ford vehicles are complex and design must be partitioned many times. The vehicle is constituted of five main systems: body, electrical, powertrain, chassis, and climate control. Each of these systems is divided into subsystems, which are themselves divided into second level subsystems and end items or components. There are frequent tradeoffs between the many systems, components, and product targets. For example, a body designed for increased safety may increase vehicle weight and reduce fuel economy. Alternatively, reducing weight may increase the use of material that is lighter, but not as recyclable as steel. Managing these tradeoffs involves balancing market, technical, and budget risks. Market risk is often the most pressing of Ford's development risks because of the fickle nature of consumers. Although the company faces technical design risk as well as pressing schedule risk due to the need to release new products annually, market risk is the only risk to which the company devotes an entire section in its annual report. The company makes enormous efforts to gage customer needs, and products are offered with varieties of options and option packages that customers can select themselves.

PD process summary

The Ford Product Development System (FPDS) is a well-documented process that includes major actions linked by reviews and iterations. The overall FPDS process is diagrammed in Figure 6.6.5, which shows how the architecture (system level design) is broken down (or cascades) to the system, subsystem and component level detailed design before later integration. The process takes almost four years for a new design, although the process is scalable to different projects. Vehicles are divided into six levels of novelty. The most complex level includes both an entirely new platform (S6) and new powertrain or engine (P6) and can take 41 months to develop. The simplest level is mainly a carryover vehicle (S1) with only peripheral new parts and designs, such as door trim or seats, and no engine or powertrain change (P1). These types of products take 18 months to develop.

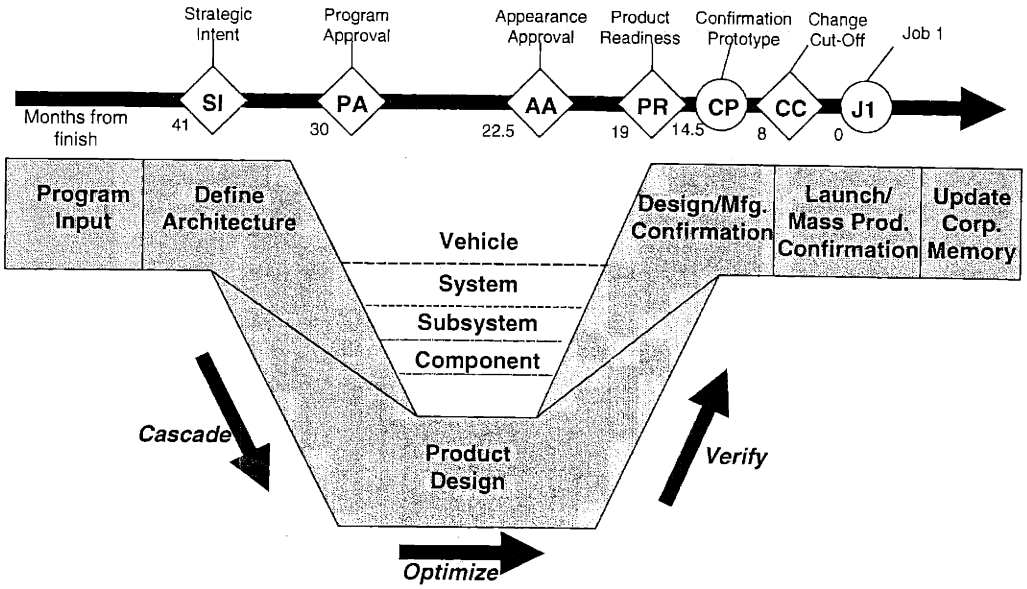


Figure 6.6.5: The Ford Product Development System

The process begins by defining requirements for a new product, incorporating customer needs, and engineering specifications. Specifications are defined and frozen in order, with vehicle design specifications defined first and component design specifications defined later. Milestones, shown at the top of Figure 6.6.5, mark progress. Each milestone includes a review and has defined deliverables. For example, to pass the product readiness milestone at month 19, a program must have a full vehicle analytical sign-off, a confirmed and issued launch plan, CAD files reflecting verification changes, and several other key deliverables. Reviews alternate between "hard" reviews that serve as engineering tests and softer ones that serve as assessments of progress.

Process iterations are usually one of two varieties. The first common iterations are intra-phase iterations between engineers working on detailed design functions. The other kinds of iterations are medium-breadth cross-phase iterations that include prototypes. The prototypes are of various quality at different points of the process, and vary depending on what aspect of design needs to be tested. Ford does produce some solid prototypes but the cost and time of these are usually prohibitive for complex integrations. Instead, the stated FPDS goal of creating "first time right" designs leads to a series of digital models of vehicles or components. The verification stage of FPDS focuses on analytical engineering and the results of computer aided engineering or prototyping.

Together, the review and iteration combination at Ford indicates a process with rigid reviews and mild cross-phase iteration to better address market risk. Ford tries to manage market risk by emphasizing instead better initial understanding of customer needs in the beginning of a program, rather than tracking changing customer needs and applying those changes to an ongoing project.

6.6.3 Arabia/DeskArtes

Case study synopsis

Company: Arabia/DeskArtes
Type: Small software and aesthetic design/manufacturing companies
Products: Ray tracing CAD software and/for ceramic tableware
Cycle time: Medium
PDP: Evolutionary prototyping
Major risks: Market risk inherent in product visual aesthetics
Notes: Arabia specifically plans and schedules cross-phase iterations, but those iterations are of relatively narrow breadth because they are encompass only the final development and prototyping phases

PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
1-2	3-4	4	3	4

Company and product description

DeskArtes is a small software company that was formed during a 1985-1989 joint research project between Arabia, a Finnish manufacturer of ceramic tableware, and the Helsinki University of Technology. DeskArtes now develops industrial design software emphasizing aesthetic features. One of its products, the Render Expert software tool, uses ray-tracing methods to generate photo realistic images from 3D geometry for presentation and marketing. Features include specialized shadows, highlights, and the ability to show various textures. Arabia, a Helsinki-based company of 300 people, uses this software to prototype its designs for ceramic tableware. These ceramic products are

the focus of the case study. Most Arabia products are made of vitro porcelain and must be at least partly hand-crafted. The 3D simulation in Figure 6.6.6 shows how the company can use the software tool to avoid some of the expensive manual prototyping work.

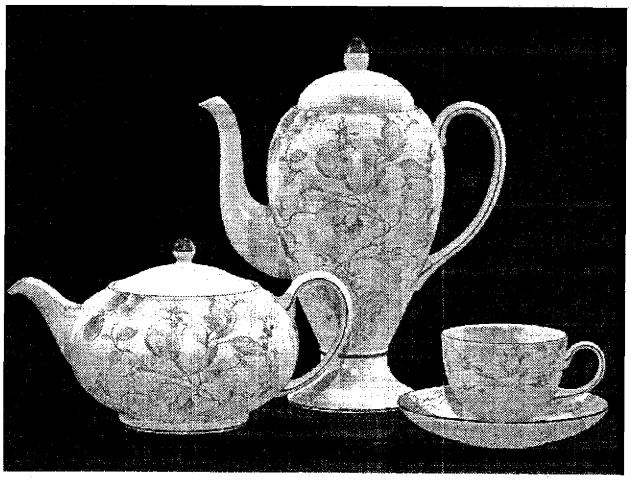


Figure 6.6.6: A computer image showing texturing on ceramic tableware

PD process summary

Product development in the aesthetics-sensitive ceramic tableware manufacturing industry has traditionally been slow because of time necessary to make ceramic prototypes. First, an artisan or designer must sculpt or form the clay. Several models may have to fired, depending on the level of deformation that occurs while the model is heated in a kiln. As Figure 6.6.7 shows, making even a single physical prototype can take weeks. Designing, prototyping, refining, and manufacturing entire multi-piece settings can take years.

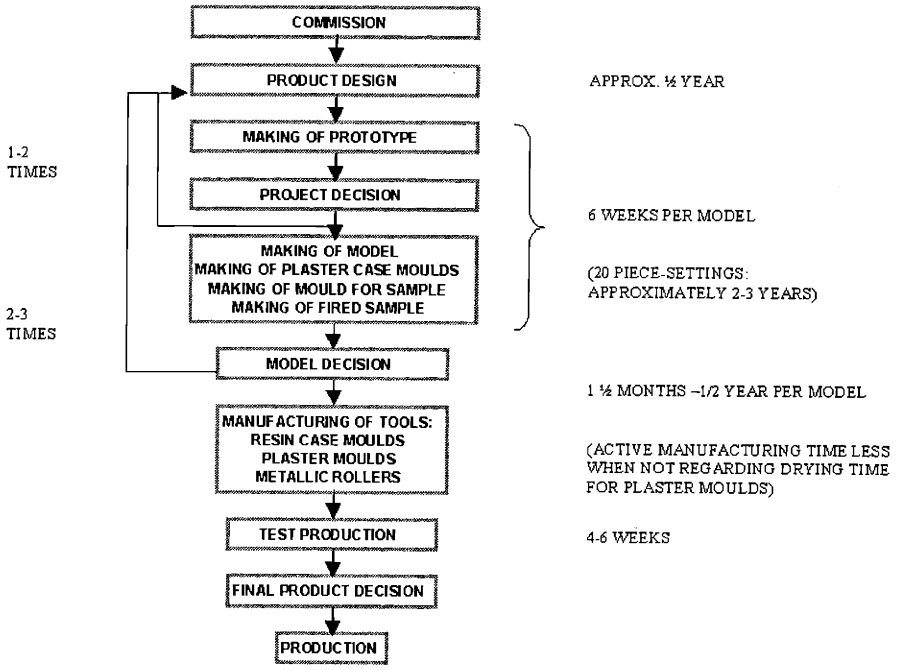


Figure 6.6.7: The traditional Arabia PDP plans iterations starting at the prototype stage (From Arabia & DeskArtes, Woodward, 2002)

The introduction of DeskArtes software allowed Arabia to save several weeks and usually even months in production time by enabling designers to visualize ideas more quickly and by allowing customers to provide immediate market feedback on aesthetic features. CAD models do not entirely eliminate the need for physical models, but can usually replace several formerly physical iterations with digital iterations.

The most telling part of Figure 6.6.7 is the degree of planning of each iteration. Arabia uses several cross-phase iterations in its PDP, and specifically plans the number of cycles (in this case, prototypes) that it will include before final production. The planned, cross phase iterations that are usually evident in software companies are used here by a manufacturing company instead. The process typifies evolutionary prototyping because of its iterations between individual design and prototyping.

6.6.4 Aviation Technology Systems

Case study synopsis

Company: ATS
Type: Small software company
Products: Air traffic control software
Cycle time: Moderate
PDP: Spiral
Major risks: Quality control
Notes: Challenges theory that mission-critical software cannot be developed using the spiral process.

PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
3	3-4	4	4	3

Aviation Technology Systems (ATS) was a small, 100-130 person company in Arlington, Virginia. It produced air traffic control software as an independent company until it was acquired by a division of Hughes Electronics Corporation that was subsequently acquired by Boeing.

ATS developed software and telecommunications processors for air traffic control, as illustrated in Figure 6.6.8. The work was performed mainly under contract for government agencies such as the Federal Aviation Administration (FAA) and Department of Defense. The system in which ATS played a part was the Future Air Navigation System (FANS-1), which aids in oceanic data links and traffic control between airplanes. The system enables oceanic, pilot-controlled free flight with frequent relaying of aircraft positions.

Figure 6.6.8: The FANS-1 system is used for oceanic data linking and traffic control

ATS faced many risks in its communication system development. As a developer of software, it faced great market risk because the overall FANS-1 system was itself immature. Market needs were uncertain because the government effort that initiated it was itself experimental. However, this market uncertainty was mitigated by the fact that, as a government contractor, ATS could be assured of receiving clear requirement specifications from its customer, even if those specifications dealt with the broader FANS-1 system instead of just the ATS components. Further, the FAA was a very involved customer. It kept close tabs on the company and frequently involved at least one of its own people in ATS' product development efforts.

Technical risk was extremely high because of the necessary product quality. Air traffic control software is a mission-critical product with low tolerance for error. This set ATS apart from many other software companies that can frequently release software with non-critical bugs.

Precise levels of budget and schedule risk for ATS remain unclear because of the lack of existing data, but managing budgets was evidently a problem for the company. The company did receive some slight bidding advantages for government contracts as a minority-owned business, but this did not compensate for the company's tendency to overspend its projected development costs.

The ATS PDP most closely resembled a spiral software development process. It cycled through a regular series of phases ranging from concept design to prototyping. In the words of one former software development manager,

> [The ATS PDP was] pure spiral model. We would go around the cycle three or four times depending on the effort, and testing was part of each round.

ATS had highly variable product development times, ranging from 9-36 months, so although the cycles would occur several times, the duration of each cycle could vary widely.

The most unusual aspect of the ATS PDP was the emphasis on quality assurance (QA). The company maintained a QA-to-developer ratio of 1/3, an unusually high ratio for a software company. In the words of one former worker, "QA was involved the *whole* time" (emphasis original). This was understandable given the technical quality concerns of the FANS-1 system, but nevertheless marks an unusual departure from conventional wisdom on the use of spiral processes.

PD literature that mentions the spiral process (McConnell, 1996; Boehm, 1988 & 1994) suggests that its weakness may be quality control. Its emphasis on flexibility allows it to adapt to changing markets, but the frequent design changes and lack of design continuity may lead to bugs. The ATS case study challenges the assertion that the spiral process only works in cases where quality control is relatively unimportant. The ATS product was mission-critical and had high quality requirements, yet used a spiral process with cross-phase iterations that led to major redesign so that information from prototypes could be included in later cycles. In this case, a spiral process was conducive to high-quality product.

6.6.5 Microsoft

Case study synopsis

Company: Microsoft
Type: Large software company
Products: PC operating and application software
Cycle time: Moderate
PDP: Spiral with elements of design-to schedule
Major risks: Technical risk manifests itself as integration risk mitigated in frequent builds. Market risk is muted because of dominant position.
Notes: "Synch and stabilize" system includes frequent cross-phase iterations
PDP Metrics:

Iteration			Review	
Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
2-3	5	4	3	3

Company description

Microsoft Corporation is the world's dominant software company. The Redmond, WA company employs almost 50,000 people and has annual revenue approaching $30 billion. Microsoft products include several large product families, including Windows, Office, games, network and server software.

Microsoft's dominance places it in a unique market and development situation. Microsoft Windows has become the standard PC operating software, and Microsoft applications, including Word, Excel, Powerpoint, and Explorer, have become standard on most PCs. Producing standard operating packages helped Microsoft edge out competitors because only it could make series of products compatible with both the standard operating system and each other. Customers were quick to discover the benefit of these network effects in the information industry, so Microsoft quickly benefited and gained market share, even if its products were not technically better than competitors' offerings. The company's policy of making software products that were merely "good enough" – instead of perfect or top-of-the-line – was more than good enough to generate profit margins of almost 80% and 90% market share.

Microsoft is more famous for its stunning success and its corporate strategy than for its product development process, but PD plays an enormous role in the company. Almost 42% of its entire work force is devoted to R&D. Many previous studies have examined Microsoft's success. (Cringely, 1992; Cusumano and Selby, 1995; McConnell, 1996; MacCormack, 2000) This section describing Microsoft is based primarily on the works of others, although one interview was independently conducted specifically for this study. The goal of this case study is to demonstrate the ability to analyze and describe a company's PDP using primarily public information.

Product development process description

The Microsoft "synch and stabilize" PDP of frequent "builds" is well documented in several sources. (Cusumano and Selby, 1995; MacCormack, 2000) The process consists of three main stages: planning, development, and stabilization. The planning phase includes writing a functional specification document, defining feature functionality, assigning priorities to features, and setting a product development schedule. The development stage includes several iterations of

- evolving specifications
- designing and coding
- continuous "local" stabilizations, testing and debugging.

Each of these iterations concludes with a milestone marking the completion of a group of features. The stabilization stage begins after the last iteration, when the features are finished and a "code complete" version of the product is released. Stabilization includes comprehensive internal and external testing of the entire software product. The process ends with product release to manufacturing.

Cusumano suggests six important tenets of the Microsoft process:

- Development and testing occur in parallel.
- Specifications are allowed to evolve.
- Not all features are developed in parallel. Iterations divide development efforts into clusters of features with the most important features in the first iteration.

- Integration does not occur in one fell swoop near the end of the process. Instead, frequent and highly-structured "builds" stabilize the product by continuously (often daily) integrating components. The company's process focuses on managing this integration risk.
- The company does not try to be perfect. A non-critical fix may have to wait until a later release before being corrected.
- The company incorporates customer feedback continuously through process

Some of these observations are also visible in the graphical adaptation of Figure 6.6.9, which uses the same terminology and divisions defined by Cusumano. This figure illustrates a spiral process as company iterates through its three PDP phases. Planning and development appear to occur serially, except that part of what is traditionally called planning, including defining functional specifications, is revisited during the development stage. The frequent builds and parallel testing are visible in the three loops spanning development, component integration and testing/debugging. Stage and milestone reviews are at the end of each stage or loop, although "code complete" is difficult for even Microsoft employees to define clearly. Three planned, cross-phase iterations are bounded by early planning and final stabilization.

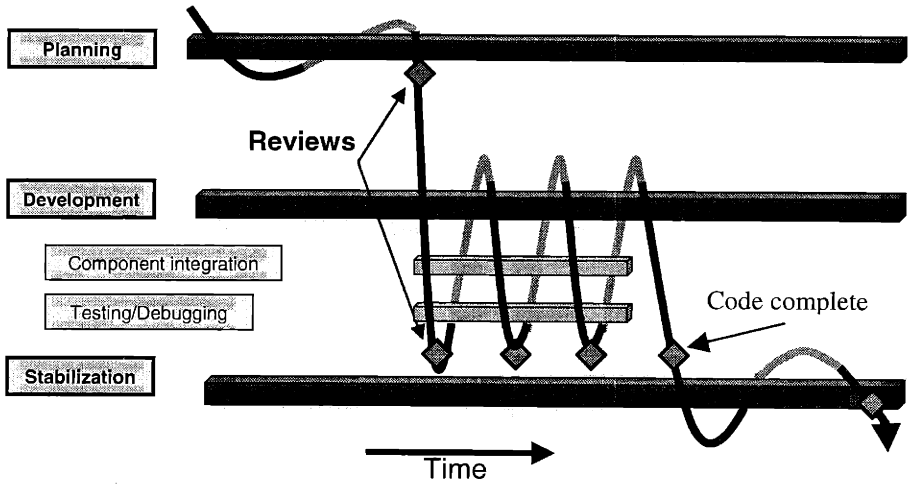


Figure 6.6.9: The Microsoft PDP (adapted from Cusumano & Shelby, 1995)

These findings were supported by other sources, which also observed several other key aspects of Microsoft PD. MacCormack (2000) demonstrate how the Microsoft process

places an emphasis on schedule by assigning 20-30 exit criteria at major milestones. He also pointed to some of the multiple ways in which developers receive feedback after making design changes. Design feedback may return from a build, from testers testing a private release, or from a formal code review by senior development team members. McConnell (1996) also confirms the use of broad iterations in software design.

An interview with a former software testing engineer supported the Cusumano finding of rapid code redesign, or "churn," by demonstrating how frequent changes prevented the design and testing cycles from being perfectly synchronized. Microsoft has a database in which software testing engineers post bugs to be corrected (or "killed," because the database is amusingly named "Raid" after the pesticide). However, the Raid database is not always updated with the latest daily build, in part because sometimes design outpaced testing. As the engineer pointed out,

> In some cases, the code changed so frequently that chances are that any bug [a software tester] found would be from 3 versions back. Bugs were often not fixed because the code that the bug was found in was already gone and replaced.

Together, the various sources help paint a detailed picture of the Microsoft process with its advantages and disadvantages. Microsoft attempts to balance structure and flexibility by using a spiral "synch and stabilize" process. Broad iterations address market risk when features are added or changed based on customer input. Iterations also address technical risks when frequent integrations and tests identify bugs that are subsequently corrected.

The process used to be even more flexible, but has been reined in slightly by the perceived need for additional PDP structure. The system had problems, as witnessed by late schedules and persistent quality issues; so Microsoft has moved to increase structure in its process by disallowing specification changes after reaching a schedule threshold. As Cusumano and Selby note, these limits to changing specifications are easier to accomplish on products that have fast cycle times, because then any changes pushed off until the next product release do not have to wait long.

The primary advantage of this PDP is that it helps Microsoft manage its technical integration risk. Without frequent builds and tests, integration near the end of a complex program would become overwhelming, and the inevitably necessary corrections would severely delay schedules and release times. The same iterations allow customer involvement feedback to affect feature development. Disadvantages of the system include schedule difficulties and quality problems in the form of bugs.

Microsoft case study conclusions

Microsoft practices a large company's version of the spiral process. Its PDP includes planned, cross-phase iterations and frequent but only moderately rigid reviews. The scope of some of the Microsoft's iterations narrowed slightly as the company incorporated more structure into its process, but the PDP still includes nearly continuous feedback for most of the development time.

The frequent builds and "synch and stabilize" processes are methods for rapid-fire integration and testing of product components and features. They serve as a mechanism for code control, debugging, and most importantly in terms of PDP analysis, feedback. The entire product still requires a major stabilization stage following code completion. It is therefore not a pure spiral process, although Microsoft incorporates most aspects of spiral development such as comprehensive iteration that entails integration and testing of subprojects.

Microsoft's PDP helps it address some of its technical and market risk. The correspondence between PDP and risk is difficult to ascertain with confidence because the company's main risks are not as clear as the risks of other companies. For example, quality problems continue, but are allowed to remain because of Microsoft's dominant design and because corporate strategy of standard-setting outweighs quality problems and renders product quality relatively unimportant. Time to market is reduced (or at least lateness is reduced) by increased rigidity and a large investment of human resources that most other companies would find difficult to emulate.

Even though many companies with fewer resources might not be able to sustain a PDP like Microsoft's, the PDP can still be described and understood effectively in terms of the parameters used to describe other PDPs. Iterations and reviews play strongly in Microsoft's process. The reviews and iterations used correspond to the types seen in other companies facing short product life cycles and rapid market and technological change.

6.7 Comparative case study findings

This section makes qualitative comparisons between individual case studies and quantifies some of these comparisons. Qualitatively, the case studies show that companies face different risks and employ different PDPs. Correlations between company risks and PDPs are reserved for the discussion of results in Chapter 7. The case studies also reveal discrepancies between companies' written policies and how they implement them. The cases further demonstrate that companies have no clear method for designing PDPs and have inconsistent reasons for making PDP decisions. Quantification of PDP characteristics allows further findings, including reinforced displays of PDP variance and visible application of the PDP metrics proposed in Chapter 4.

Different risks, different processes

Each case study company faces different risks. Technical risks are overriding concerns for some companies. The preeminent risk for SWPG is technical risk in areas of thermal stress and efficiency, at least in part because the industry market conditions translate many other risks into technical risk. ATS also faced technical risks, although its technical risk stemmed from high software quality requirements. ITT's overriding risk in the SUO program is technical by default because market risk is limited by the company's status as a monopolistic defense contractor. Other companies, such as IDe, Ford, and Printco, are more deeply concerned by market risks. IDe customizes its development to suit customers. Ford products must be offered in many modular packages in order to meet customer expectations. Printco is introducing a PCB marker into an untested market with unfamiliar customers. A third set of PD programs, such as the ITT GPS and Xerox Document Center efforts, are most severely limited by time constraints and schedule risks.

These findings that certain risks are preeminent do not preclude the existence of other risks. As discussed in Chapter 3, most companies face several interdependent categories of development risks. These case study findings demonstrate such overlapping risks. For example, although Printco faces market risk in developing its new PCB marker for unfamiliar customers, its chemical and mechanical engineers also face technical risk in

developing an ink that prints with adequate quality without clogging printheads. This technical risk sometimes leads to extra work (impacting budget), delays (impacting schedule) and later release dates, which impacts market acceptance. Similarly, SWPG's technical risks overshadow but do not obscure other company risks, including schedule concerns or budget risk stemming from the expensive machining of some turbine components. Despite these overlaps, qualitative evaluations and interview results with each company often identify either a preeminent risk or a category of outstanding risks. These are the most pressing challenges that, if unmet, would likely doom a PD program.

Just as risks differ between companies, each case study company also employs a different PDP. Most of the companies, including SWPG, ITT, Ford, Sikorsky, and Printco, use stage gate processes, although the forms of these stage gate processes differ. For example, SWPG employs the most rigid stage gate process, while ITT uses progressive freezes to lend flexibility to its process. Several other companies instead use spiral processes as either their primary PDP or as one of their development subprocesses. For example, Microsoft and ATS employ "cycling" PDPs, and Xerox uses the spiral process for some of its software development. Meanwhile, IDe and DeskArtes customers use evolutionary delivery processes that emphasize prototype development and customer testing. Finally, several companies combine processes. For example, ITT uses a more flexible process in its SUO development than in its GPS development. Similarly, UTC employs different PDPs for its Sikorsky and Pratt & Whitney companies. Also, both Microsoft and IDe combine their processes (spiral and evolutionary delivery, respectively) with the design-to-schedule process whenever they are forced to prioritize and cut product features. This qualitative survey of PDPs shows that they vary tremendously between companies.

Thus, both development risks and PDPs differ among companies, although these qualitative observations alone do not yet confirm a correlation between risks and PDPs. Additional observations and quantifications – both included in subsequent sections – are still necessary to be able to better compare and define the risk management roles of PDPs.

Designing and implementing PDPs

The case studies also reveal management difficulty in designing and implementing PDPs. First, the cases demonstrate various reasons and inconsistent methods for choosing PDPs. Second, the cases display frequent discrepancies between companies' written and implemented processes

There is no consistent method by which companies design or select their PDPs. Although the case studies did not examine the underlying philosophy of management decisions that led to PDP definitions, several disparate paths were evident. First, some companies changed their PDPs due to organizational shifts. For example, SWPG formalized and added rigidity to its process after Siemens purchased Westinghouse and chose to impose Siemens' more ordered processes on its new acquisition. Second, some companies redesigned their processes when leading individuals perceived and wanted to address specific problems. For example, IDe progressed slowly from a loosely-designed and flexible process to a more rigid evolutionary development decision as its four lead managers determined that the company's rapidly-growing workforce required more order. Similarly, the Xerox process was reformed with the lead of the company's chief engineer in part to overcome persistent PD lateness. Finally, some companies had their own idiosyncratic reasons for PDP designs. Several of these companies hired consultants to help them design or redesign their PD efforts; one of them specifically adopted a process as "a management fad" that was advertised by a consultant as having worked elsewhere. In another example, some of Printco's PDP changes were driven by a manager who had read a good book.[13] On the other hand, Microsoft modeled its PDP after the culture of its developers by retaining "hacker" traits of frequent changes in development code. Meanwhile, another company showed inertial resistance to modifying its PDPs, so it used the same PDP for all products even when the company entered new markets with new types of products. In summary, some companies carefully consider the PDPs they would implement, but others employ PDPs with little regard for the "fit," or

[13] The company president was personally convinced of the need to change his PDP after reading *Winning at New Products*, by Robert Cooper (2nd, edition, 1993) This research uses the 3rd edition (2001) of the same book as one of its sources.

suitability, of those processes to company-specific risks or challenges. Companies based their PDP decisions on many different factors, but none had an analytical process to follow.

Companies also have difficulty implementing the official processes they design. The case studies investigate and probe actual, implemented PDPs because they frequently differ from companies' written processes. One of the few commonalities among all case studies is that every one reveals discrepancies between written and implemented processes. Sometimes, those differences are minor, as in the case of SWPG's "phantom" reviews that are never supposed to occur according to the written PDP. Other times, the differences are due to informal improvements to the written PDP, such as when ITT informally allows program managers to omit sections of integrated management plans that they deem extraneous. Finally, some differences between written and actual PDPs are harmful and the result of poor implementation of good ideas. An example of this occurs at Printco, which does not implement the rigid review after its sanctioning phase, as specifically prescribed by the company PDP. Thus, discrepancies between written and actual processes are frequent, although those differences can be helpful, harmful, or innocuous depending on the circumstances. These discrepancies must be taken into account in order to gain accurate understanding of companies' PDPs.

Quantified findings

Some case study findings are quantifiable according to the metrics proposed in Chapter 4. Design iterations and reviews for each company are rated on their own multiple scales according to the information gathered from interviews, questionnaires, company documentation, or literature. These data suggest three immediate findings.

- Each company does indeed iterate and review, although the characteristics of those iterations and reviews are different.
- The differences reaffirm the earlier qualitative finding that different companies employ genuinely different PDPs.
- The data show a trend towards more flexible PDPs among software developers.

Chapter 4 reasoned that all companies use iterations and reviews, and these findings confirm that this true for each of the case study companies. Although there is significant variation in the breadth or number of iterations, some form of iteration is common to all companies, as can be seen in the charts included in each case study synopsis. Some companies, such as SWPG, include mainly intra-phase iterations, while others, such as Ford and DeskArtes, include more cross-phase iterations with prototyping. Similarly, all companies employ design reviews, although the characteristics of reviews vary. For example, Xerox design reviews are rigid while ITT design reviews are sometimes less rigid and based on peer reviews and evaluations.

The differences in the iteration and review characteristics reaffirm the earlier qualitative finding that PDPs differ across companies. The quantitative differences can be seen more clearly in Figure 6.7.1, which displays the iteration and review characteristics for each of the studied processes. As described in Chapter 4, higher values imply a more flexible process and lower values tend toward processes that emphasize predictability. Just as Figure 4.7 displayed theoretical processes in terms iteration and review, this figure compares actual processes.

Company			SWPG	IDe	ITT	Xerox	ATS	Markem	UTC	Ford	DeskArtes/ Arabia	Microsoft
PDP desicription			Strict stage gate	Evol. delivery	Progressive freeze	Hybrid	Spiral	Aspiring stage gate	Stage gate	Stage gate with quasi prototyes	Evol. delivery	Spiral
Parameters	Iteration	Breadth	1	2	1-2	1-3	3	1-2	2	2	1-2	2-3
		# of inter-phase loops	0	3	1-2	0-3	3-4	1-2	2	3	3-4	5
		Planning	1	4	3-4	3-4	4	1	3	3	4	4
	Review	Rigidity	2	4	3-4	1-4	4	4	1	1	3	3
		Frequency	1	5	1-2	1-3	3	1-2	1	1	4	3
	Risk	Profile	Tech risks dominate – (heat rate /efficiency) Mkt risk muted by contract structure	Major risk is market - new company is highly customer sensitive	Major risks are tech & schedule, depending on product Market risk limited by military contracting	Market risk translates to sched concerns	Major risks are tech, primarily QA	Mkt on one project, tech on another from late integration	Major risks are tech – FAA regs and quality require ments	Mkt risk greatest, but complex tech & budget risk also high	Market risks dominate - customer aesthetics	All risks muted by dominance: features driven by mkt risk, tech risk dominates

Figure 6.7.1: Comparative chart of PDPs and major parameters

Key:
Breadth of iterations: Narrow 1 ◄——► 3 Comprehensive
Number of interphase loops: None 0 ◄——► 4 Multiple
Degree of iteration planning: Unexpected 1 ◄——► 5 Planned & scheduled
Review rigidity: More rigid 1 ◄——► 5 Less rigid
Review frequency: Frequency 1 ◄——► 5 Less frequent

No two columns of defining characteristics are identical in Figure 6.7.1, suggesting that the corresponding PDPs are also different. Each column represents a PDP's "fingerprint." Although not as uniquely different as actual human fingerprints, each column's values can identify a different PDP.

Finally, the data suggest a trend towards more flexible PDPs among software developers, as suggested by McConnell [1996] and Gilb [1988]. As described in section 4.3, the individual metrics can be normalized and aggregated to create composite iteration and review values. Together, these values can provide an overall image of process flexibility, as show in Figure 6.7.2

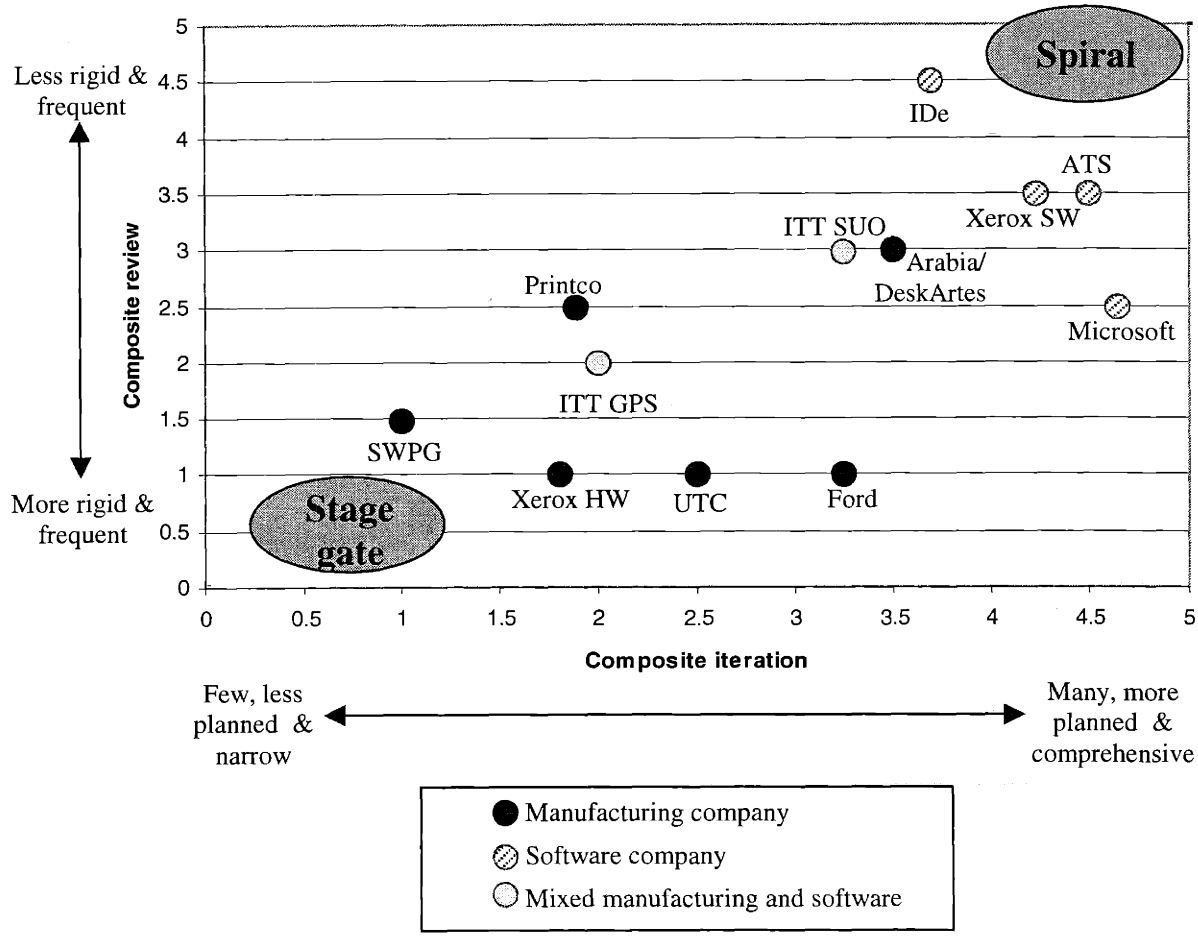


Figure 6.7.2: Overall PDP flexibility by iteration and review

Figure 6.7.2 shows the composite metrics with a plot of PDPs in terms of aggregate iterations and reviews. High iteration and review values indicate a process favoring flexibility while low iteration and review values indicate a process favoring predictability. Companies are plotted individually, except for Xerox and ITT, which are included both in total and by their two divisions. In the figure, manufacturing companies tend toward the lower left of the chart because they employ more rigid reviews and have fewer cross-phase iterations. Arabia and ITT SUO are notable exceptions, and demonstrate the use of flexible processes in the development of manufactured products. Software companies tend toward the upper right of the chart, while companies with mixed manufacturing and software components are shown in the middle. This trend suggests that software developers are more likely to favor flexibility in their PDPs. The reasons for this apparent trend – the different risks the companies must address through their iterations and reviews – are not shown on the chart but are discussed in Chapter 7. Hardware/manufacturing discrepancies alone do not predict PDP flexibility. Finally, the chart shows most PDPs underneath the diagonal, suggesting that the case study companies are more flexible with their iterations than they are with their reviews.

In conclusion, the case studies provide several findings, both individually and collectively. Individually, each case study provides a rich story of companies' risks, PDP design, and PDP implementation. Collectively, qualitative findings include the presence of different risks, the use of several PDP variations, and frequent difficulty in designing and implementing PDPs. The quantification of key iteration and review characteristics reinforce the finding of differences between companies' PDPs, demonstrates the applicability of metrics to PDP analysis, and reveals that software companies tend to use more flexible processes.

7. DISCUSSION OF RESULTS

There's a world of difference between truth and facts.
Facts can obscure the truth.
– Maya Angelou

This chapter discusses PDP results and draws lessons from the case studies. The most important research result is the establishment of a relationship of PDPs, risk and integrations. Risk and integration are instrumental in determining the applicability of PDPs. This relation is supported by case study findings on PDPs, risk variety, and the role of iterations, reviews and integrations in risk management. Secondary results include the establishment of metrics for identifying and comparing PDPs, the discovery that the spiral process may be more broadly applicable than expected, and the recognition that companies' PDP design and implementation can be improved.

7.1 Linking PDPs and risk

The clearest research finding is that companies use substantively different PDPs. These differences are evident both qualitatively and quantitatively. Qualitatively, each case study describes a different process with unique PDP diagrams, individual prototype schedules, varied levels of customer involvement, and distinctive employee process descriptions. Although there are many similarities among processes, including frequent use of the major PD actions described in Chapter 2 (planning, concept design, detailed design, integration and test, and release) the processes themselves vary in the order of, repetition of, and thresholds between these actions. Quantitatively, these differences become clearer. The differences in iterations (including integrations) and reviews that are shown in Figure 6.7.1 reflect actual variations between processes. These findings suggest that PDP differences are substantive. Most PD literature assumes a basic process – suggesting that all PD efforts should or do follow one major process – and focuses on how to implement that PDP effectively by describing the individual actions in great detail. The results of these case studies indicate that such an assumption of a single or basic PDP is unwarranted because of the variety of different PDPs. The collection of

case studies exhibits several PDPs and suggest many more permutations, which is enough to indicate variety among PDPs.

A second research finding is the variety of risks faced by different companies. This result was entirely expected because development risks have been well-documented in previous literature and remain a pressing challenge for most businesses. However, it is necessary for the case studies to clearly identify these different risks because one of the goals of this research was to link PDPs and risk management. The case study companies face a representative variety of PD risks, ranging from the design of challenging technical components to market uncertainty about whether a product feature is actually demanded by customers. It is helpful, although not necessary, to categorize these risks as technology risk, market risk, schedule risk, and budget risk. These categorizations are not always possible because some risks may be both causes and effects of other risks (technology challenges can lead to schedule delays, and budget limitations can lead to fewer technical resources or lower product quality) while other risks, such as budget and schedule risks, are sometimes entirely interchangeable. When accurate categorization is impossible, then individual risks can simply be identified instead.

Identification of different PDPs and different risks invites exploration of potential correlation between PDPs and risk management. The research findings indicate that different PDPs and risk profiles are not random phenomena; rather, different PDPs manage different risks. Given that companies face different risks and employ different PDPs, the findings further indicate that:

1) Design iterations and reviews manage certain risks.
2) All PDPs include design iterations and reviews.
3) Each PDP therefore manages certain risks.

The case studies, existing literature, and common business experience all support the first contention that both iterations and reviews manage certain risks. Iterations can manage risk by providing information feedback. Different types of iterations help manage different risks. As IDe and Microsoft exemplify, cross-phase iterations that encompass

prototyping may provide customer feedback and thereby reduce market risk. The same iterations could reduce technical risk if integrations yield useful feedback to design groups, and could reduce schedule or budget risk if they avert major redesigns near the program delivery date. As Xerox and ITT demonstrate, narrower iterations may allow for clear engineering requirements or good communication between detailed engineering groups and thereby reduce technical risk. The same iterations could reduce schedule or budget risk if they prevent rework due to specification changes.

Case studies and literature suggest that reviews also manage risk in several ways. By dividing a development program into separate and discrete actions, reviews segment complex design problems. Some reviews, such as the Printco reviews, tend to be frequent but not rigid, and therefore serve as assessments. These may help reduce schedule or market risk by providing updates on program progress and improving planning accuracy. Other reviews, such as ITT peer reviews, tend to seek out design difficulties while the goal of the gate reviews is to ensure that there are no technical design problems. These help to improve product quality and reduce technical risk.

The second contention, that all PDPs include design reviews and iterations, is also supported by the research findings. The case study descriptions indicate that reviews and iterations are important parts of all PDPs. They are, in fact, so common that this research proposed metrics based on the kinds of iterations and reviews used. Figure 6.7.1 shows that different iteration and review combinations can actually help *define* company PDPs.

The third contention, which frames PDPs as risk management frameworks, is not only the logical conclusion of the first two contentions, but is supported by the case studies as well. Figure 6.7.2 uses a measure of PDP flexibility based on reviews and iterations, and finds that software-oriented companies or groups are more likely than manufacturing companies to have flexible PDPs. Software developers have significantly different risk profiles than do their manufacturing counterparts: software product lifetimes are shorter, development cycles are more rapid and frequent, manufacturing lead time is less of an issue, and prototyping costs are usually less. It is thus not surprising that, given the

different risks among industries, companies within an industry or general risk profile tend to employ PDPs with similar strengths. The correlation may be due to the role of PDPs in risk management and the ability of companies to build prototypes or perform integrations and tests. The following subsection further explores these explanations.

The case study findings lead to the conclusion that PDP design choices are numerous and that they have major risk management consequences. PDPs are not monolithic or singular entities that can be applied equivalently to all companies with just slight modifications. PDPs must not be merely tweaked or customized to suit the needs of innovating companies; they must be consciously selected or designed to match company risks and attributes.

7.1.1 Risks and integrations: Results from alternative categorization of case studies

Although risk is an important factor distinguishing companies and affecting PD, it is not the only – or perhaps even the most telling – factor. As has been suggested earlier in Chapter 2, a company's ability to integrate and test products can also be a critical descriptor and determinant of PDPs.

Product integration often includes an early model, test, simulation, or prototype involving interdependent components or modules. A key question for companies is whether the value of the information they gain from such a test or feedback loop is worth the cost and time of the integration. The value of perfect information can sometimes be calculated analytically [de Neufville, 1990] but often the information received in imperfect and the value is difficult to predict in advance. Further, sometimes the information cannot be practically gained at all until an initial product is actually produced. Such difficulties are common in industries that produce complex mechanical products that cannot be prototyped without great expense or without significant lead time. One example of integration difficulty can be seen in the SWPG case study, where the company often sells its first "prototype" to a customer willing to take a risk on buying an untested product in exchange for a reduced price or guarantees of SWPG servicing in case of failure. Of course the company uses computer simulations to model overall systems, but the fidelity

and quality of these simulations are not as good as the first actual prototype. Other companies such as DeskArtes, Microsoft, or ATS, can test their products more easily because integrations of their software products require no physical construction or major production expense. Their simulations are not merely models of reality; they are actual parts of the code that later become the final product.

The importance of both risk and integration differences among companies can be seen respectively in the following two charts, Figures 7.1 and 7.2. These charts are similar to Figure 6.7.2, but views companies through the lens of different categories to draw greater insight into why companies use certain PDPs. Figure 7.1 shows the same companies on identical axes to those shown in Figure 6.7.2, but differs in two respects. First, it excludes Printco, the one case study company that also serves as a test application for the PDP design method described in Chapter 8. Printco was dissatisfied with its PDP in part because it recognized that the process did not adequately manage several company risks. The company is redisplayed (in its new location on the locus of reviews and iteration) in Figure 8.3. The remaining case studies are categorized by the preeminent risks faced by the different companies.

Figure 7.1 suggests that two different companies can share the same category of major risks, yet employ different PDPs to manage those risks. For example, both SWPG and ITT (on its SUO program) face primarily technical development risks, yet the companies use significantly different processes. This suggests two major lessons. The first lesson is that different PDPs can manage the same categories of risk. While it is true that "schedule risk" companies tend toward less flexible processes and "market risk" companies tend toward more flexible processes, there is significant scatter among companies facing technical risk. The iteration and review combinations among these companies – and thus emphasis on either flexibility or predictability – are different, so although there is no one-to-one correlation between risk type and PDP, the results show that PDPs manage risks in different ways. The second lesson is that knowledge of the category of major risks faced is not enough to prescribe or predict which kind of PDP a company employs. Additional knowledge about the specific risks, rather than broad

categories, may provide greater insight. This suggests, as Chapter 2 implied, that individual risks must be considered as well categories of risks.

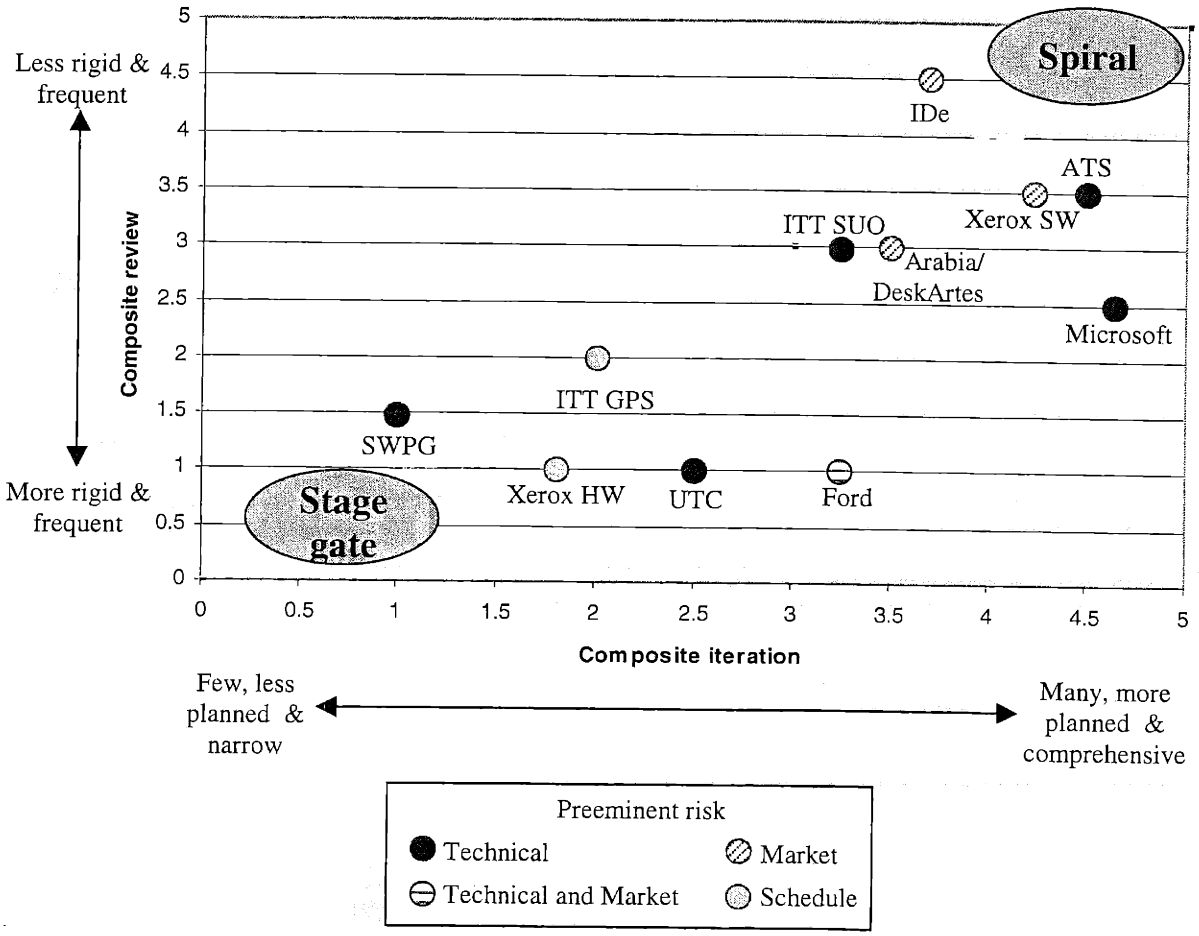


Figure 7.1: Case studies charted by preeminent risk

Since Figure 7.1 shows that different PDPs can manage similar risks, it can be surmised that risk may not be the most telling factor in PDP decisions. Knowledge of risk alone does not resolve the scatter of Figure 7.1, however Figure 7.2 shows a different categorization of companies that resolves the case studies into clear groups or clusters. This latter figure shows the same companies categorized by the ease with which they can integrate sample or test products. The ease of integration is rough measure of the cost and time required for an integration relative to the value gained from the test. Some companies call these integrations "prototypes," while others call them "builds" or

"stubs," but they all represent a form of iteration and attempted risk reduction. Such integrations not only address certain risks by providing information feedback from prototypes, but also represent risks of their own by potentially costing a company time, funds, and effort. As such, these integrations become particularly useful lenses by which to view and categorize companies. Figure 7.2 shows distinct clusters of companies in the two corners of the chart, each grouped with other companies in the same category. This suggests that integration is a powerful determinant of the process type that companies may employ. Ease of integration, among the case study companies, suggests an ability and desire to be more flexible.

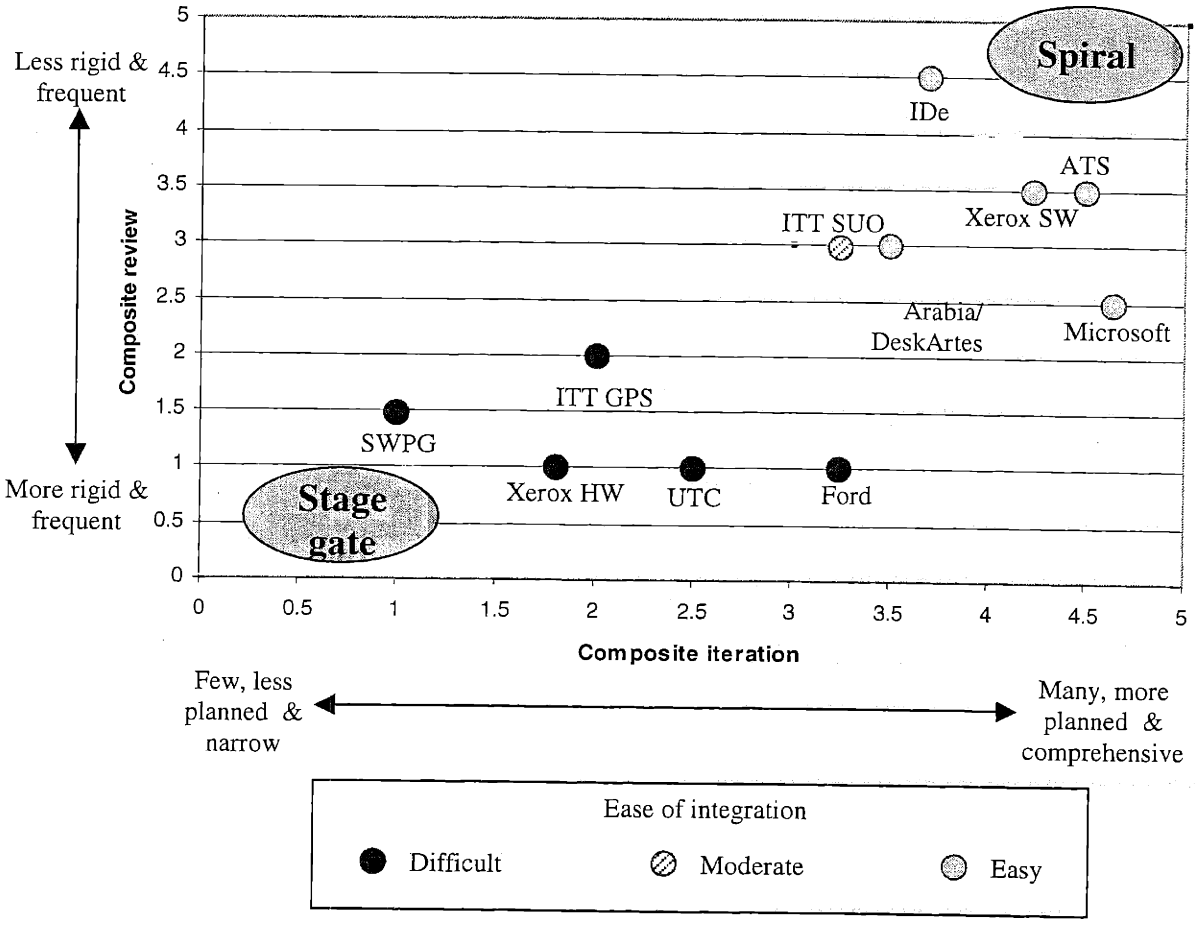


Figure 7.2: Case studies charted by ability to perform integrations and test

Overall, these comparative charts lead to several important results. Most important is the revelation that although risk is a distinctive component of PD, it is not currently a predictively telling characteristic because different PDPs can manage similar risks. Ease

(or difficulty) of integration, on the other hand, is a decisive quality of PD that may help predict the applicability of different PDPs to address risks. Further, most company PDPs cluster towards two corners of Figures 6.7.2, 7.1, and 7.2. Exceptions tend to occupy the lower right hand quadrant of the same charts, suggesting that companies are more flexible with their iterations than their reviews. It may be possible in the future to find companies that would occupy the upper left corner of the charts, but such companies are likely rare because of the difficulty of maintaining rigid iterations while simultaneously loosening reviews.

7.2 Secondary results

7.2.1 The establishment of useful PDP metrics

The establishment of iteration and review as metrics for identifying and comparing PDPs is a fortuitous secondary result of this research. These metrics were initially proposed only as a means to an end: as a tool that could help distinguish PDPs. The tool itself was not assumed to be particularly important except in its assistance in demonstrating how different PDPs could address different risks.

The metrics gained in importance for two reasons. First, they filled a gap in PDP practice in literature. The metrics were necessary because previous literature provided no equitable way of comparing or contrasting PDPs on a common scale. Indeed, most previous literature made no attempt to contrast PDPs at all because many sources considered just one process. Those sources that did acknowledge the existence of PDP choices frequently compared PDPs based on either diagram shape or subjective advantages and disadvantages. These proved to be difficult criteria by which to compare the initial case studies. The iteration and review metrics provided a much-needed common language in which different PDPs could be identified.

The second reason why the metrics became important was because they proved to be practical and useful. Because they were based on key traits shared by all PDPs, they were able to capture basic PDP facts about every case study company. Each PDP encountered could be described in terms of risk or iteration metrics. Once described

quantitatively, the PDPs could be uniquely identified, compared, and contrasted. Moreover, these metrics were both understandable to and welcomed by practitioners, who enthusiastically greeted the metrics as a way to better understand their own processes. The metrics are easy to communicate and access in case studies: managers are frequently able to describe major iterations (often because of the changes they entail) and engineers are intimately familiar with the character of design and development reviews. Together, the conceptual ease of communication and general applicability of the metrics made them useful.

As suggested in Chapter 4, the iteration and review metrics are still of limited use for three reasons. First, PDP descriptions lack fidelity when analyzed in terms of these metrics alone because the characteristics are general. For example, the metric for iteration breadth only indicates how many stages are spanned by an iteration; it does not indicate which stages are included. Thus, when measured by this metric alone, an iteration spanning the front end of a process (including specifications through design) could look identical to an iteration spanning the back end of the same process (including design through integration). A second limitation of the metrics is that, although they could be potentially decomposed in many ways, any decomposition would lead to industry specificity and a lack of general applicability. For example, if metrics were deconstructed to include daily builds or tests, those deconstructed metrics would only be of use in software or related industries where frequent tests and prototypes are possible. A final caveat to these metrics is that there is no guarantee that the metrics can continue to describe every company's process. However, the repeated successes in capturing each of the case study PDPs is a promising sign.

7.2.2 The applicability of the spiral process

Several case studies show that the spiral process is more broadly applicable than earlier thought. Several sources tout the spiral process as effective in software development where quality requirements are only moderate or low. Among the case studies, Microsoft fits this mold because it employs a spiral process to develop software that is "good enough" but invariably released with bugs. However, the spiral process appeared in other

companies as well, where it was less expected. ATS, for example, used the spiral process for mission critical flight control software with high quality requirements. Xerox uses a spiral subprocess for its network (although not control) software requirements as part of a hybrid PDP. ITT is experimenting with spiral-style flexibility in some of its development programs, and one source describes P&W engine product development as following a spiral process. These examples demonstrate that the spiral process is more widespread than anticipated. The repeated instances are unlikely to all be anomalies, and may represent a business trend of introducing greater flexibility to PD.

7.2.3 PDP design and implementation difficulties

The study also confirms the need for better PDP decisionmaking. It was hypothesized earlier that companies have no clear method for designing and steering their PDP efforts. Case studies confirmed that companies were inconsistent in their reasons for selecting or designing PDPs. Some companies designed their PDPs to meet company-specific risks, but the case study histories indicate that several companies made their PDP decisions haphazardly, as a result of management trends, corporate momentum, or company mimicry.

A related discovery is the common gap between official PDP processes and the processes that companies actually implement. One research objective was to overcome this gap and describe the "true" processes; in doing so, the discrepancies between actual and prescribed processes became abundantly clear. Although time-consuming for a case study researcher, the incongruities can be revealing. As exemplified in section 6.7, developers sometimes break from official company policy because the official process does not address actual development needs. These cases of "ground level" and sometimes surreptitious corrections reflect a failure in PDP design and the difficulty inherent in selecting a PDP suitable to each company.

7.3 Threats to validity and other concerns

The PDP comparisons and research results are based on both qualitative findings and quantitative metrics, so both of these bases must withstand scrutiny. This section

addresses questions about and challenges to the research methodology and interpretation of results. First, it addresses the validity and utility of the iteration and review metrics. Second, the section reinforces the importance of case study analysis in this subjective field of research. Finally, it discusses the sensitivity of findings to research assumptions.

7.3.1 Value of metrics

Although the iteration and review metrics are a useful means of identifying, comparing, and contrasting PDPs, they have two major flaws. First, they do not capture the full range of PDP differences and must demonstrate their worth relative to other potential PDP metrics. Second, there is no statistical analysis of how well iterations and review characterize PDPs, so the metrics are not fully validated.

Iteration and review are only two of many potential PDP metrics. The most obvious alternative metric is an outcome-based measure of PD success. However, defining and rating product development success is difficult, and translating a measure of product development success to PDP success is even more problematic. One other researcher tries to compare PDPs by their success, but to limited avail because of the difficulty of this problem. (MacCormack, 2000) Product sales, payback period, profit, time-to-market, and customer satisfaction are all poor ways of rating PDP success because they may be affected by many external variables. By these measures, a successful Ford small vehicle development program with high levels of customer satisfaction may falsely appear to be a failure because of the low profit margins of the product. However, from a strategic view, the company still has an interest in building market share among young consumers in the hopes that they will become repeat customers and buy other Ford products later. The low profit margin would be a poor measure of PD success and an even poorer measure of PDP success, since many other variables – including competitors' products, economic trends, and several other external factors – can lead a good PDP to develop an unsuccessful product or a poor PDP to develop successful products. Alternatively, Microsoft may produce an extraordinarily successful and profitable product because of market conditions, even though the process that developed it may be flawed. Defining and rating PDP success based on outcome remains

problematic and would be a poor substitute for the operation-based metrics of iteration and review.

Other alternative metrics could include fully decomposed PDP factors, such as the number of prototypes and manufacturing lead times. Companies could also be segmented by market and explicitly categorized as start-up or established firms, software or manufacturing companies, and government contractors or commodity developers. Such factors are extremely important, as acknowledged in Chapter 4, because they profoundly affect companies' risk profiles. However, many of these factors are local to certain environments or business areas and lack the universality necessary to compare PDPs from different industries. This approach of this thesis is to consider these factors in risk assessment, but not to use them as metrics for comparing PDPs.

A second shortcoming is that there is no statistical analysis demonstrating how well iteration and review metrics define PDPs. Such analysis would be ineffectual with the small sample size of the case study population, although a future, larger-scale study could test the statistical significance of these variables with respect to different company PDPs. This research builds grounded theory – namely that risk and integration factors are instrumental in determining the applicability of PDPs – and suggests that these iteration and review characteristics would be promising test variables. The grounded theory is based on earlier reasoning that iterations and reviews are common to all PDPs. Each case study supports this theory; all PDPs include iterations and reviews, and all PDPs have different kinds and combinations of iteration breadth, numbers of iteration, level of iteration planning, review rigidity and review frequency. The utility and importance of these variables is further buttressed by the fluidity with which these variables could be discussed during interviews. Thus, this research suggests that iteration and review characteristics are important and telling variables in PDPs. Case studies support the metrics as useful PDP descriptors, but additional and larger-scale research is necessary to prove their significance.

Finally, the weighting of iteration and review characteristics was subjective and could have led to different results if evaluated differently. For example, in creating the composite index of PDP flexibility shown in Figure 6.7.2, iteration breadth and review rigidity were considered equally. It is possible that another researcher would weigh one of these factors more heavily than the other and therefore arrive at different values of overall PDP flexibility. Because these metrics are new quantifications of formerly qualitative data, they may fall victim to subjective interpretations until a large-scale study is able to assign levels of importance to each of the variables. Until then, personal judgment is the only guide to the relative weighting of different iteration and review characteristics.

7.3.2 Value of case study research

As discussed in the Chapter 5 description of methodology, case study research has both strengths and limitations. The strengths of case study methodology correspond well to the needs of this research. Case studies are useful for demonstrating facts, collecting qualitative data, assimilating and balancing subjective views, and building theory that can eventually be tested quantitatively. This research demonstrates facts by displaying several companies' different PDPs and risks. The research also collects qualitative and subjective data in the form of interviews and process documentation and assimilates it from a neutral viewpoint of a disinterested party. Finally, the research builds a grounded theory of PDP risk management and proposes metrics by which large populations can be studied in the future. Case study research and methodologies, including interview technique and observations, are useful in achieving research goals.

Conversely, this research is restricted by the limits of case study methods. The research builds and supports convincing theory and sets the stage for future work, but technically proves little. It demonstrates several different PDPs, which may be regarded as proof of existence and a refutation of the argument that all PDPs are fundamentally the same process, but even this "proof" is open to interpretation. The similarities and differences between PDPs can still be a matter of perspective. Just as all people can be lumped together as one because all humans have hearts and lungs, all PDPs could be classified as

one because they all include basic development actions. However, such a narrow view would miss many important distinctions. This research demonstrates critical differences between PDPs and makes the case that the PDPs themselves vary tremendously. The proof of differences between PDPs is accomplished, in case study format, entirely negatively by showing that PDPs are not the same. Similarly, the case study of ATS demonstrates (negatively) that the spiral process need not be restricted to non-mission critical or low quality products. Except for these two proofs of existence, the results are mostly limited to support of the theory that different PDPs manage different risks. The case studies provide qualitative support for the theory, and the findings indicate the usefulness of new proposed metrics for PDP identification and comparison.

7.3.3 Assumptions of risk and match between PDPs and companies

This research assumes that the case study companies employ at least competent PDPs that do not cause a history of major PD mismatches or failures. This assumption is necessary because otherwise the link between PDP and risk management would be spurious. For example, if all of the case study companies employed PDPs that did not suit their needs, as in the case of Printco, this research might misidentify the companies' major risks and jeopardize the research finding of a link between risks and PDPs. However, the assumption invites the challenge that the research is either tautological or circular: PDPs are discovered to manage company risks, and company risks are discovered based on PDPs. Further, the assumption itself is challenged by the finding that many companies have difficulty designing PDPs and therefore make suboptimal choices.

The research findings withstand these challenges for two main reasons. First, the assumption of PDP competence is acceptable because, as discussed earlier, PDP success is difficult to define or rate. There are many internal and external variables and subjective criteria that can mask the effectiveness of PDPs. However, repeated and sustained PDP failure is more obvious; it would almost certainly lead to major losses or a company's downfall. Research results would be in serious doubt if findings were based on such failing companies. Second, an assumption of competence does not imply

optimality. Current case studies can illustrate important risk management concepts while still leaving room for improvement in the future.

7.4 Other observations

In addition to the results and rebuttals above, there are several other interesting observations stemming from the case studies. These observations are tangential to the main research goals but are still worthy of note. They include the unusual role of cycle time, and a new view of the evolutionary prototyping process.

7.4.1 PD cycle times

PD cycle times can be counterintuitive and misleading. One might assume that companies with long cycle times would be particularly attuned to market risk because market needs are more likely to change over a long time period than over a short period. Thus, companies with long cycle times would emphasize prototyping and customer involvement. Conversely, companies with short cycle times – often software companies – could afford to ignore these market feedback efforts because customer testing would take valuable time and any potential improvements could be included in the next product version, usually already in the pipeline.

Reality proves otherwise. Although many companies face the common difficulty of writing specifications, fast-paced companies tend to favor flexible processes, such as the spiral process or evolutionary delivery process, that incorporate frequent customer interaction or testing. This preference may be because the benefits of market feedback outweigh the relatively slight costs of prototyping a "virtual" product. Meanwhile, manufacturing companies that release products less frequently tend to use fewer planned, cross-phase iterations and therefore build fewer integrated prototypes. This occurs because of product complexity and the steep costs and lead times necessary to build physical models. Both kinds of companies may face difficulties in writing their specifications, but the companies

The counterintuitive result of this mismatch is that companies with the greatest need for market flexibility are sometimes the least likely to generate customer feedback during a PD cycle. Companies that are most impervious to market changes because of short cycle times frequently incorporate market feedback anyway.

7.4.2 Evolutionary delivery in another perspective

The evolutionary delivery process can be viewed as a technical equivalent of the design-to-schedule and design-to-budget processes. (See sections 3.4 and 3.5) Although much of this thesis is devoted to distinguishing between PDPs, in this case the similarities between evolutionary delivery and the "design-to" processes are more interesting than the differences.

The process description in section 3.5 groups the design-to-schedule and design-to-budget processes together because they both work incrementally until a deadline is reached. From a process perspective, it does not matter whether the deadline is temporal or monetary. However, the case studies included evolutionary delivery process examples that operate in much the same way. "Completed" products are introduced, but if they do not reach a certain technical standard of acceptance (i.e. quality issues or a lack of enough features), then they are merely called prototypes and another iteration is included until the product meets the technical standard of customer acceptance. From a process perspective, the difference between a deadline and meeting a standard of acceptability is minimal. If the deadline is a time deadline, then a PD effort will likely not go beyond schedule, but may exceed budget and not include every desired technical feature. If the deadline is a budget deadline, then a PD effort will likely be within budget, but at technical and monetary cost. Similarly, if a "deadline" is a technical standard, then a PD effort will likely meet that standard, but at the cost of budget and schedule. Although there are differences between standards and deadlines, the iteration pattern of evolutionary design processes are similar to those of the design-to-schedule/budget processes.

7.5 Chapter summary

One of the major goals of this theory-building research is to describe how different PDPs manage risk. The case study results show that iterations and reviews serve not only as successful process metrics, but as links between PDPs and risk management. Examples demonstrate how certain iterations manage market risk, how certain reviews manage schedule risk, and several other correlations. The most distinct predictor of PDP selection is the ability of a company to perform integrations. The importance of these integrations is identified thanks to the ubiquity of iteration and reviews characteristics which can be used as PDP metrics.

The establishment of PDP metrics is an important collateral result because, although the purpose of the metrics are to allows for the PDP comparisons necessary to establish their risk management characteristics, the metrics themselves are a significant contributions to the field of research. Other secondary results suggest that successful PDP design and application is not fully understood by either researchers or practitioners, and that these actions might be improved.

Caveats to the results include the limitations of the metrics, subjectivity in the case studies, and sensitivity to assumptions. However, the impacts of these threats are limited and do not invalidate the case study results, which are descriptive and which provide lessons on the advantages of using certain PDPs. The next chapter further describes these lessons and applies them to provide a prescriptive PDP design tool.

8. APPLYING CASE STUDY LESSONS: DESIGNING PDPS TO MANAGE RISK

'Would you tell me, please, which way I ought to go from here?'
'That depends a good deal on where you want to get to.'
– Lewis Carroll's Alice in Wonderland

The case studies demonstrate that companies implement different PDPs, but are inconsistent in designing or selecting their processes. Adopting a process that works for another company or following a management fad is not likely to lead to success. PDPs should be customized to different companies or programs; but PDPs should be methodically customized to different companies or programs. The need for an improved PDP design method is real and immediate. This chapter applies lessons from the case studies to propose and apply a helpful tool: a PDP design method that can assist companies in planning or selecting their PDPs. The PDP design method matches risks to specific iterations and reviews of processes, thus helping companies design processes that suit their own risk profiles. Printco, one of the earlier case study companies, serves as an example of how the PDP design method works.

8.1 PDP design method proposal

In proposing a new PDP design method, relevant lessons from Chapter 7 results include:

- There is a need to improve the way companies design their PDPs.
- Companies face unique sets of individual development risks that should be the basis for PDP design.
- PDPs are amalgamations of iterations and reviews; these are the relevant characteristics that should be analyzed in PDP design.
- Iterations can help manage several risks; the type of risk managed depends on how frequently the iterations occur and what actions the iterations span. For example, technical and market risk can both be handled by several iterations spanning design prototyping stages.
- Design reviews may also manage risks, both in conjunction with and independently of iterations. The types of risks managed depends on the characteristics of the design

reviews. For example, frequent reviews can provide the control necessary to handle schedule and budget risk.

To translate these descriptive lessons into prescriptive actions, four key steps can help companies better design PDPs. Of course, not every PDP needs to be designed from scratch. The same steps can be used to help select which of many existing PDPs may best fit a company, or can be used to modify a process already in use. The steps are:

1. Identify and prioritize project risks
2. Assign each risk to a specific phase or cycle
3. Plan the necessary iteration cycles to address the assigned risks
4. Schedule reviews at the completion of key stages, times, or cycles

These steps are illustrated in Figure 8.1, which diagrams how the proposed PDP design method helps to match companies with PDPs.

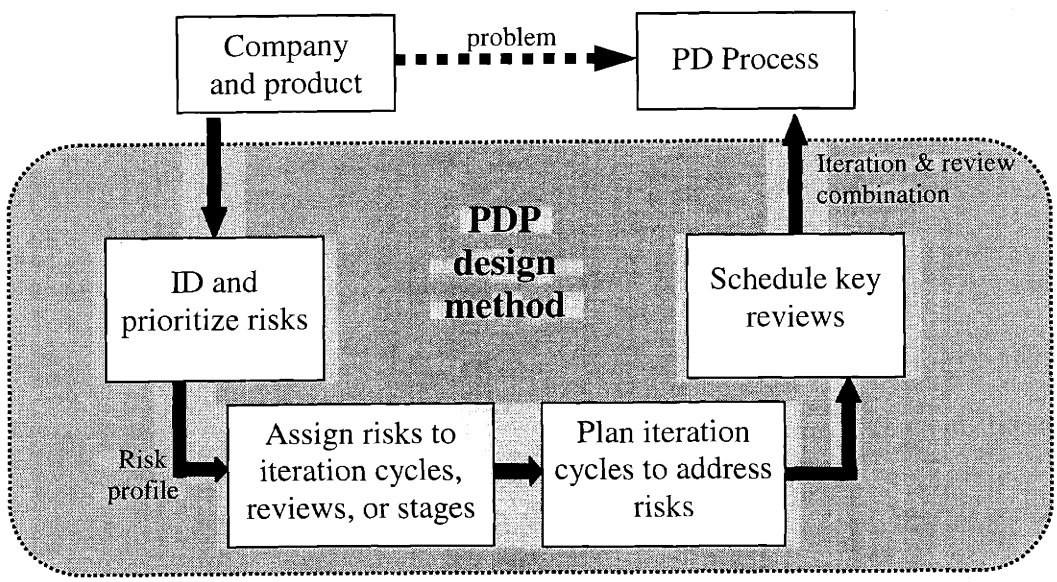


Figure 8.1: PDP design method

A company begins by identifying and prioritizing the risks that it faces in a development program. The risk identification is frequently based on either past experience or recognized uncertainties. Past experiences, such as a history of lateness or product quality issues, are a relatively easy way for companies to estimate which risks are greatest in a current development program. Uncertainties that are recognized, such as the ambiguous customer focus group results or the knowledge that a company is introducing a new product to an untested customer base, can also help identify risks once potential impact costs are assigned to those uncertainties. The company should be able to categorize most risks as technical, market, schedule, or budget risks, although some risks will defy classification. For example, specification definition often falls in the category of market risk, but a hardware/software interface issue that could arise during integration might result in both technical and schedule risk. Risks are then prioritized. There are often one or two "showstoppers," or high-probability risks that are likely to ensure failure if they are not addressed. The resulting risk profile becomes the focus of PDP attention.

Case studies have shown how iterations and reviews can address risks, After risks have been identified and prioritized, the risks are assigned to specific iteration cycles, stages, or reviews for them to manage. The most isolated risks can be simply be assigned to stages rather than to an iteration and review combination. For example, technical risk regarding the design of an isolated product component can be assigned to a detailed design stage with only minor, intra-phase iterations among design engineers. More complicated risks are assigned to iteration cycles and reviews instead. For example, high customer uncertainty and the resulting market risk may be assigned to two planned, cross-phase iterations that incorporate one prototype or customer test per cycle. Each iteration provides feedback that reduces risk in the next round. Schedule risk, which is so often linked to budget risk because of the frequently linear relationship between schedule and the cost of labor, may prompt a company to include reviews at regular time increments rather than at the end of a phase.

Once risks are assigned to iteration cycles and reviews, the iterations and reviews are planned, as shown in Figure 8.2. The figure, which can apply to either a company

process or program-specific process, shows three labeled iteration/review combinations, each of which would have had a particular risk or set of risks assigned to it. Depending on the risk(s) addressed, the iterations may span different phases or include reviews at different times, but the result is that risk is reduced with each stage or cycle.

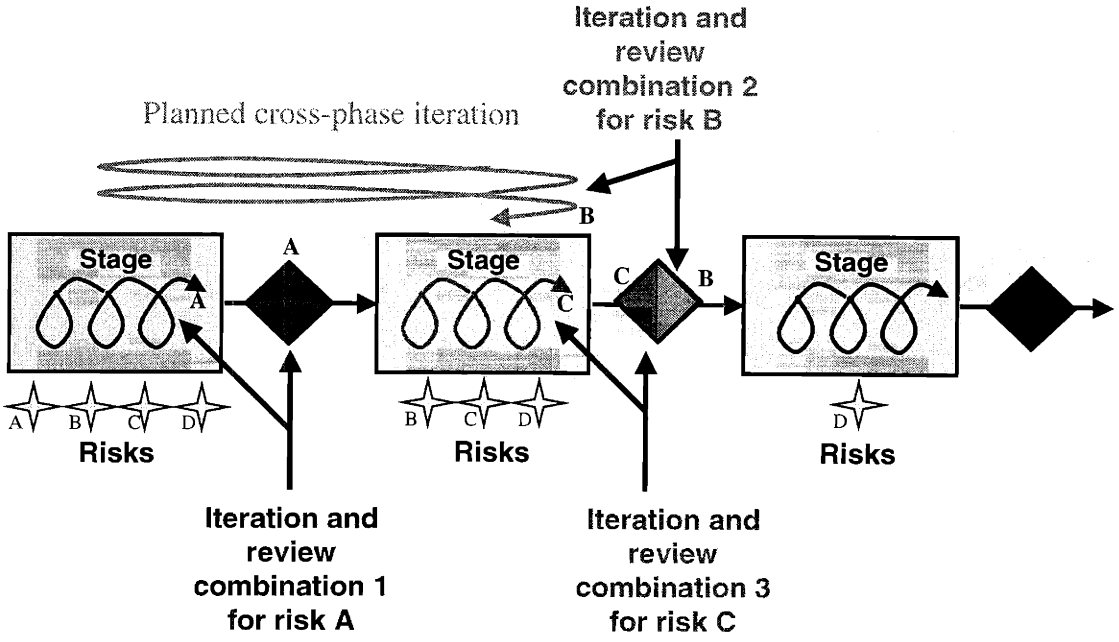


Figure 8.2: Planning iterations and reviews to address risks

Together, the stages linked by a unique combination of iterations and reviews constitute a PDP, reaching the goal of the PDP design method. The method links companies' development programs with ideal PDPs by using risk, iterations, and reviews as intermediaries. The result is a PDP prescription that addresses a company's major development risks.

8.2 Method demonstration: Printco revisited

In order to demonstrate the use of the PDP design method, one company was chosen to serve as an example. Printco, one of the case study companies (see section 6.5), expressed interest in using the lessons of this research immediately because it was in the early stages of reorganizing its own PDP. This section discusses how the method was used to prescribe process improvements for Printco, which is currently making changes based in part on these suggestions.

Printco has earned its reputation for developing good products in the marking and coding industry, but the company has outgrown its PDP. The company acknowledged its need to improve its PDP, and was leaning toward formalizing and enforcing a series of existing stage gates. Several in the company also recognized Printco's need to improve "up front" work so that marketing information can be incorporated early for a well-specified product.

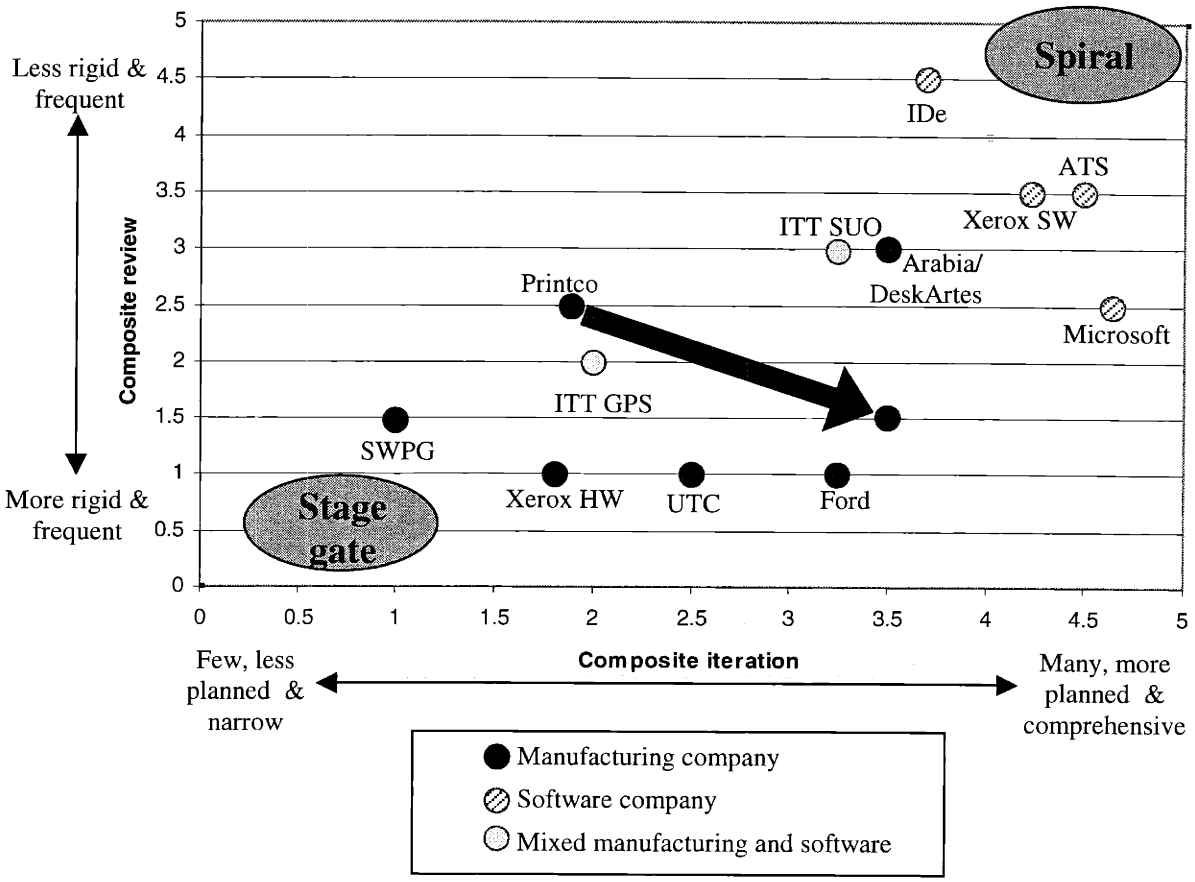
When applied to Printco, the PDP design method identified the same risks described earlier in section 6.5. Printco's various PD efforts face very different – and sometimes unusual – risk profiles. Derivative products like the Model P220/440, which might be expected to face just minor technical risk, faced surprising amounts of market risk. The new 26K Series faces both market and technical risk, and but fell victim to delay because of technical challenges that expanded late in the program. Both have gone over time and over budget. The risk profiles were determined with the help of several employees because of the great sensitivity of the PDP design method to risk identification. If risks are misidentified, the method will prescribe a process that addresses the wrong development risks.

Printco's risk did not match with its existing iteration and review scheme. According to the PDP parameters (the design iteration and reviews metrics seen in Figure 6.7.1) established earlier, the current Printco PDP incorporates design iterations of medium breadth. Most iterations are intraphase with usually only one or two interphase loops which are visible as rework. The degree of planning of these iterations is low because they are neither encouraged nor overtly expected. Printco reviews are more frequent than its iterations but lack rigidity.

The next step in the PDP method called for reassigning Printco's risks to iterations and reviews that would correct the problems of the old process. It was recommended that Printco make some of its reviews more rigid, especially in the first part of the process, after sanctioning, and during early design to establish greater control over market and

schedule risk. It was also suggested that, rather than eliminating the rework-filled, cross phase iterations that currently delay the company's development efforts, the company should acknowledge the need for such iterations and plan market prototypes in them. Other companies have seen success in incorporating both the rigid reviews of stage gate processes and the planned iterations of early prototyping. Printco, on the other hand, has had difficulty in building prototypes early enough to garner the information necessary to substantively change design. Information from "late stages" arrives so late that change is costly and difficult.

Part of the PD problem might be solved by resolving to freeze specifications earlier and to maintain rigid reviews, but this places great faith in the company's marketing abilities. It also suggests an expectation that technical uncertainty will lead to the greatest risks, which is not always true. In fact, the company appears to overestimate the importance of technical risk at the expense of the market risk that it faces. For this reason, the double approach of rigid iteration and modestly planned prototyping iterations may help the company in the likely event that its specifications will not be able to be capably frozen as early as engineers would like. Simultaneously increasing rigidity of early reviews and flexibility of prototyping iterations would help the company deal more effectively with its several risks. Described in terms of the metrics defined in Chapter 4 and used in Figure 6.7.2, the recommended process changes would appear as shown below in Figure 8.3. This would "move" Printco from the upper left quadrant of Figure 6.7.2 to the lower right quadrant, reflecting increased flexibility in iterations and more restrictive reviews.



	Iteration			Review	
	Scope	# of inter-phase loops	Level of planning	Rigidity	Frequency
Current PDP	1-2	1-2	1	4	1-2
Suggested PDP	2-3	2-3	3	2	1

Suggested PDP incorporates more flexible iterations for greater market feedback and reduced market risk

Suggested PDP incorporates more rigid reviews for improved scheduling and reduced tech risk

Key:
Breadth of iterations: Narrow 1 ⟷ 3 Comprehensive
Number of interphase loops: None 0 ⟷ 4 Multiple
Degree of iteration planning: Unexpected 1 ⟷ 5 Planned & scheduled
Review rigidity: More rigid 1 ⟷ 5 Less rigid
Review frequency: Frequency 1 ⟷ 5 Less frequent

Figure 8.3: The current and suggested Printco PDPs in terms of the metrics described in Chapter 4.

These recommendations based on the PDP design method were presented first to Printco's PD manager and then to an assembly of engineers, marketers, and managers. The recommendations generated discussion, because it had been Printco's earlier inclination to move towards a traditional stage gate process. However, these recommendations suggested that, although a stage gate PDP could help Printco if its marketing organization is able to garner the information necessary to define both requirement and design specifications early, this may not always be possible. Further, Printco could not reduce its risks simply by adding stage gates.

Printco is making PDP changes based in part on the PDP design method recommendations. The changes include more rigid reviews in the beginning of product development cycles and the planned incorporation of at least one cross-phase prototyping iteration for customer feedback. The recommendations were well-received by management and by most engineers; however, results of the resulting process changes will not be known until at least the end of the product development cycle in which they are implemented. The Printco demonstration does not yet validate the PDP design method, but the positive reaction and reception to the recommendations suggest that the method is a reasoned application of the iteration and review lessons learned earlier. On a personal level, the application of the method and metrics helped engineers and managers make sense of what they were doing.

9. CONCLUSIONS

The products of the Sirius Cybernetics Corporation have fundamental design flaws that are completely hidden by their superficial design flaws. It is easy to be blinded by the essential uselessness of them by the sense of achievement you get from getting them to work at all.
– The Hitchhiker's Guide to the Galaxy, as written by Douglas Adams

Product development is a risky necessity for many innovating companies. Although it holds the promise of increased sales, market share and profits, it can fail in whirlpools of technical difficulties, cost overruns, and missed market opportunities. PDPs must therefore not only focus on the final outcome – a new product – but also on mitigating the many development risks. This research exhibits and explains PDPs as risk management structures. In exploring the relationships between risk management and PDP design, this research makes four key contributions.

First, the research analyzes several PDPs both theoretically and empirically to demonstrate how PDPs differ substantively. It builds upon previous literature that either does not adequately distinguish between different processes or otherwise makes comparisons based on varying or subjective criteria. This research makes a secondary contribution to the field by proposing and supporting new metrics with which PDPs can be identified, compared and contrasted. The metrics are based on design reviews and iterations, which are characteristics shared by all PDPs.

Second, the thesis describes how iterations and reviews, and thus PDPs, manage different development risks. Planned iterations can generate valuable information that feeds back to early process stages and reduce risk. Reviews further contribute to risk reduction by providing controls over changes, schedules, and information flows. The research also reveals that, although PDPs manage risk differently, companies do not always design or select them based on these characteristics.

Third, the research identifies integrations and tests as telling and predictive components of PDPs. Risk identification remains important, as suggested above, but integrations and

tests help to address risks by providing information feedbacks. Integrations may also generate risks if their costs and time outweigh the value of the information gained.

The fourth major contribution is the proposal and ongoing application of a PDP design method based on risk, iteration, and review. This tool is the most easily applicable contribution of this research, and can provide companies with a framework or path by which they can intelligently design PDPs that suit their needs. The method is simple to follow and allows companies to evaluate their own risks and then deal with those risks with specific parts their PDPs. Just as segmentation is a valuable tool in marketing products, dividing PDPs into reviews and iterations, as well as into their traditional stages, can be helpful to product development.

Case study data strongly support the first three contributions. The PDP design method, although unproven, is exemplified and is being applied at one company. Together, the contributions link PDP and risk management and can improve understanding and implementation of product development.

The research also provides several secondary findings, including the identification of trends in PDP use among several companies. Software companies are more likely to employ PDPs that emphasize flexibility, while manufacturing companies are more likely to use PDPs that emphasize predictability. However, spiral and other iteration-intensive processes are applied more broadly than earlier expected; several companies employ flexible PDPs for either mission-critical or manufactured products.

The secondary findings illustrate the value of considering PDPs with evenhanded metrics and in terms of the risks they address. Alternative explanations for PDP differences would not support the findings of this research. For example, both software and manufactured products can be complex, so complexity alone does not account for PDP differences. Cycle times are also an initially tempting explanation for the differences in PDP preference, but this factor is also misleading because the fast cycle times of software products could imply a need for less, rather than more, flexibility. However, by

examining PDPs in terms of iteration, review, and risks addressed, we can gain a better understanding of which PDPs might apply in different situations.

9.1 Future research

This theory-building research leads to many potential research avenues. The resulting theory and lessons on the relationship between PDPs and risk can be further tested and expanded. Additionally, the proposed tool for improved PDP design should be further evaluated and improved. This section outlines four promising areas of future research.

The most helpful continuation of this research would refine and reinforce the PDP metrics by surveying a large population of companies about their PDPs. This study identified and supported the use of key variables, such as iteration breadth and review frequency, but was based only on several cases and could not assess the statistical significance of each variable. A useful next step would be to test these variables already shown to be promising. This could be accomplished by studying a large sample of companies to assess the five proposed key variables and test if they are significant descriptors and differentiators of PDPs. Additionally, any DSM models of companies' processes might illustrate precisely which types of information flow to expect, and could clarify the stages spanned by key iterations.

A second avenue of further investigation would further decompose variables that this research recommends for distinguishing PDPs. Such a study could focus on some of the alternative variables and factors considered in Chapter 4, including the breakup of iterations according to the included stages, or the use of industry-specific criteria such as prototyping lead time. Such a study would likely have to limit its scope to certain industries, for example either heavy manufacturing or software, to avoid the problem of classifying companies with dissimilar and incomparable characteristics.

A third research option would attempt to rate the relative success of the PDPs defined here. As discussed in Chapter 7, rating the overall success of PDPs is an extraordinarily difficult challenge, in part because of the lack of convincing metrics. Effective PDPs can

sometimes generate failed product offerings due to exogenous variables, and poor PDPs can sometimes succeed despite themselves. Profit, time to market, customer satisfaction, and several other common metrics are of extremely limited use in determining the "success" of PDPs. However, the relative success of managing risk may be easier to measure. Future research could identify critical success factors by comparing matched pairs of companies with similar products and similar risks but different PDPs. These variables could be instrumental in helping companies determine if process changes are actually beneficial.

Finally, further research could improve the classification and transferability of development risks. The current research uses four risk categories whenever possible, but is weakened by risks that defy categorization, at which point it resorts to the more cumbersome approach of individual risk management. Some of those risks, such as manufacturing or labor relations risk, are outside the scope of this research. Other risks are simply difficult to categorize because they span categories or are easily transferable, such as when a particular technical risk is addressed by assigning more engineers to a task and thus threatening a project budget. Research to date has difficulty separating endogenous from exogenous risks and cause-based risks from effect-based risks.

9.2 Final thoughts

Improving product development remains an important goal from an academic, business management, and social perspective. From an academic perspective, the complexity of product development is a fascinating engineering systems challenge. Product development entails multiple actors, numerous designs, potentially millions of components, countless interactions, many possible solutions, and even more possible failures. While individuals and companies may want to design products for profit or use, product development and processes are important scholastic subjects because academia plays an important role in developing analytical tools, understanding the variables, and increasing the comprehension and control of such complex and interactive systems.

From a business management perspective, product development can be gateway to success or failure. Many aspects of management – ranging from corporate strategy to finance and marketing – can impact profit or shareholder value, but product development is among the most important. Product development efforts generate what companies sell, are the basis of marketing, are the end goal of innovation management, are the object of finance, and can help shape corporate direction. Product development poses many challenges: a cynical business saying is "Good...Fast...Cheap. Pick any two," but product development managers must balance all three goals. Understanding and improving product development is a major component of business success.

Finally, improved product development is an important social goal. Development, improvement, and innovation are among the most inspiring parts of human nature. It was once believed that the ability to make tools – the most rudimentary and singular form of product development – was a trait that separated humans from all other creatures. Now we know that some other animals also develop tools by experimenting with and crafting elementary tools, usually to help them eat. Humans have long progressed beyond food-tool making (although forks, and chopsticks are still painstakingly designed and crafted or manufactured around the world) and now make screwdrivers, light bulbs, cars, drugs, computer programs, child safety seats, weapons systems, and duct tape. Many – although certainly not all – of these products are socially beneficial, both intrinsically because of what they do and economically because of the jobs that they indirectly create. Most products also have the potential to create negative externalities in the form of pollution, traffic, or other natural and social costs. It is thus no surprise that society is deeply interested in both encouraging and managing product development. This societal interest manifests itself whenever people read shopping catalogs to learn about the latest gadget, engineers engage in research, or governments enact environmental policies, cloning restrictions, and consumer safety standards.

It is the hope of the author that this research may also be applied to improving the often-strained relationship between regulated businesses and government agencies. Companies frequently bemoan costs imposed by regulations, such as environmental laws designed to

reduce air pollution or spur development of more fuel-efficient designs. It is unfortunate when laws and regulations are onerous to businesses, but the burdens of such laws cannot be entirely eliminated without sacrificing societal interests. However, improved product development may be a partial solution to this problem. More effective development processes is one way in which companies may reduce costs and respond more robustly to both government regulations and competitive challenges. Improved product development should always be viewed as an opportunity.

REFERENCES

Abernathy, W. and Utterback, J. "Patterns of Industrial Innovation," *Innovation/ Technology Review*, 1978, pp. 40-47.

Abernathy, W. and Clark, K. "Innovation: Mapping the Winds of Creative Destruction," *Research Policy*, Vol. 14, North-Holland: Elsevier Science Publishers, 1985 pp. 3-22.

Alic, John; Branscomb, Lewis; Brooks, Harvey; Carter, Ashton; Epstein, Gerald, *Beyond Spinoff: Military and Commercial Technologies in a Changing World*, Boston: Harvard Business School Press, 1992.

Ansell, Jake and Wharton, Frank (ed.) *Risk: Analysis, Assessment and Management*, New York: John Wiley & Sons, 1992.

Appelbaum, Alec, "Europe Cracks Down on E-Waste," *IEEE Spectrum*, May 2002, Vol. 39, No. 5, pp. 46-51.

Ashford, N. and Caldart, C. *Technology, Law, and the Working Environment*, Washington DC: Island Press, 1996.

Bannister et. al. "Evolution of Westinghouse Heavy-Duty Power Generation and Industrial Combustion Turbines," *Transactions of the ASME, Journal of Engineering for Gas Turbines and Power,* Vol. 118. No. 2, April 1996, p. 325.

Bayegan, Markus, "Doing R&D Smarter: A More Effective Approach to Industrial R&D," MIT Lecture by the CTO and Head of Group R&D of ABB, Ltd. March 18, 2002.

Beck, Kent, *Extreme Programming Explained*, Boston: Addison-Wesley, 2000.

Ben-Haim, Yakov, *Information-Gap Decisions Under Sever Uncertainty,* Haifa, Israel: Technion, Academic Press, 2001.

Bernstein, Peter L. *Against the Gods: The Remarkable Story of Risk,* New York: John Wiley and Sons, Inc., 1996.

Blum, B. I. "The Life Cycle – A Debate Over Alternate Models," *ACM SIGSOFT Software Engineering Notes*, Vol. 12, No. 8, 1982, pp. 988-993.

Blum, Fred H. "Getting Individuals to Give Information to the Outsider," *Journal of Social Issues*, Vol. 8, No. 3, 1952 pp. 34-52.

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

Boehm, Barry, "A Spiral Model of Software Development and Enhancement," *IEEE Computer,* 1988, pp. 61-72

Boehm, Barry and Bose, Prasanta, "A Collaborative Spiral Software Process Model Based on Theory W," IEEE, 1994.

Bohn, Roger E. "Measuring and Managing Technological Knowledge," *Sloan Management Review*, Fall 1994, pp. 61-73.

Bower, J. and Christensen, C. "Disruptive Technologies: Catching the Wave," *Harvard Business Review*, January-February 1995.

Buchanan, David, et al. "Getting In, Getting On, Getting Out, and Getting Back," Ch. 3, pp. 53-67, in *Doing Research in Organizations*, Alan Bryman ed., New York: Routledge, 1988.

Burgess, Robert G, *In the Field: An Introduction to Field Research*, Boston: George Allen and Unwin, 1984.

Christensen, "The Limits of the Technology S Curve," Parts I and II, *Production and Operations Management*, Vol. 1, No. 4, Fall 1992. Part 2 on pp. 358-366

Christensen, Clayton, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*, Boston: Harvard Business School Press, 1997.

Christensen, C. M. and Bower, J. L. "Customer Power, Strategic Investment, and the Failure of Leading Firms, *Strategic Management Journal*, Vol 17, 1994, pp. 197-218.

Clark, K. and Fujimoto, T. *Product Development Performance*, Boston: Harvard Business School Press, 1991.

Cleland, David, Project Management, Strategic Design and Implementation, 2nd ed. New York: McGraw-Hill, 1994.

Cohen, Wesley and Levinthal, Daniel, "Absorbtive Capacity: A New Perspective on Learning and Innovation," *Administrative Science Quarterly*, Vol. 35, 1990, pp. 128-152.

Cooper, Robert G. *Winning at New Products, 3rd ed.* Cambridge: Perseus Publishing, 2001.

Crawley, E., "System Architecture" Lecture notes, Cambridge: Massachusetts Institute of Technology, 2001.

Cringely, Robert X. *Accidental Empires: How the Boys of Silicon Valley Make Their Millions, Battle Foreign Competition, and Still Can't Get a Date*, Reading, Mass.: Addison-Wesley, 1992.

Craig, David C. *Promises and Pitfalls of Architectural Strategy in the Printer Industry*, Cambridge, MA: Massachusetts Institute of Technology Master's Thesis, 2001.

Cressy, Donald R. *Other People's Money*, Glencoe Illinois: The Free Press, 1953.

Cusumano, Michael A. The Japanese Automobile Industry, Cambridge: Harvard University Press, 1991.

Cusumano, Michael and Selby, Richard, *Microsoft Secrets*, New York: The Free Press, 1995.

Cusumano, Michael, Mylonadis, Y. and Rosenbloom R. "Strategic Maneuvering and Market Dynamics: The Triumph of VHS over Beta," *Business History Review*, 1992.

Cusumano, Michael and Nobeoka, "Organizational Requirements for Multi-Project Management," Ch. 7 in *Thinking Beyond Lean*, New York: The Free Press, 1999.

Cusumano, M. A. and Smith, S. A., "Beyond the Waterfall: Software Development at Microsoft, in D. B. Yoffie (ed.) *Competing in an Age of Digital Convergence*, Boston: Harvard Business School Press, 1997.

David, P. "The Dynamo and the Computer: A Historical Perspective on the Modern Productivity Paradox," *American Economic Review*, Vol. 80(2), 1990, pp. 355-361.

David, Paul A. "Clio and the Economics of QWERTY," *American Economic Review - AEA Papers and Proceedings: Economic History*, Vol. 75, No. 2, May 1995, pp. 332-337.

Davis, Craig R. "Calculated Risk: A Framework for Evaluating Product Development," *Sloan Management Review*, Summer 2002, Vol. 43, No. 4, pp. 71-77.

Dawkins, Richard, *The Blind Watchmaker*, New York: W.W. Norton & Company, 1996.

De Meyer, Arnoud; Loch, Christoph; and Pich, Michael, "Managing Project Uncertainty: From Variation to Chaos," MIT Sloan Management Review, Winter 2002, Vol. 43, No. 2, pp. 60-67

De Neufville, Richard, *Applied Systems Analysis*, New York: McGraw Hill, 1990.

Dong, Qi, *Predicting and Managing System Interactions at Early Phase of the Product Development Process*, MIT Doctoral Thesis, Cambridge: Massachusetts Institute of Technology, 2002.

Dougherty, Deborah, "Grounded Theory Research Methods," in *Blackwell Companion to Organizations*, Joel Baum, ed., Medford, MA: Blackwell Publishers, 2002.

Eppinger, Steven; Whitney, Daniel, et. al. *Organizing the Tasks in Complex Design Projects*, ASME 2nd International Conference3 on Design Theory and Methodology, 1990.

Eppinger, Steven, et. al. "A Model-Based Method for Organizing Tasks in Product Development," *Research in Engineering Design*, 6:1-13, 1994.

Eppinger, Whitney, et. al. *Generalized Models of Design Iteration Using Signal Flow Graphs*, MIT Sloan Working Paper #3866, 1996.

Eppinger, Steven D. "Innovation at the Speed of Information," *Harvard Business Review*, January 2001, Vol. 79, No. 1, pp. 149-158.

Eppinger, Steven D. and Salminen, V.K. *Patterns of Product Development Interactions,* International Conference on Engineering Design 2001, Glasgow, August 21-23, 2001.

Experian Information Solutions, *Aviation Technology Systems Corporation*, Experian Business Reports E02461812 (1991) and E03802655 (1997)

Fine, Charles and Whitney, Daniel, "Is the Make-Buy Decision Process a Core Competence?" *Sloan School of Management Working Papers*, 1-38, 1996-97. Presented at the MIT Symposium on Technology Supply Chains, May 10-11, 1995.

Ford website A:
ford.com/en/ourcompany/communityandculture/connectingwithsociety/todayschoicesfortomorrow.htm

Foster, R. and Kaplan, S. *Creative Destruction*, "Control, Permission, and Risk," New York: Currency, Ch. 4, 2001.

Foster, R. "The S-Curve: A New Forecasting Tool," *Innovation, The Attacker's Advantage*, Summit Books, Simon and Schuster, New York. Pp. 88-111. (Ch. 4), 1986.

Foster, "Timing Technological Transitions," in Horwitch, Mel (Ed.), *Technology in the Modern Corporation, a Strategic Perspective,* NY: Pergamon Press,1986.

Freedman, David H. "Fuel Cells Vs. The Grid," *Technology Review*, January/February 2002, Vol. 105, No. 1, pp.40-47.

Gilb, Tom, *Principles of Software Engineering Management*, Reading, Mass: Addison-Wesley Publishing Company, 1988.

Gilmour, Peter and Hunt, Robert, *The Management of Technology*, Melbourne: Longman Cheshire Pty Ltd. 1993.

Gordon, Marcy, "SEC head met with Xerox Chief," *The Boston Globe*, May 19, 2002, p. A15.

Griffin, Abbie and Hauser, John, "The Voice of the Customer," *Marketing Science*, Vol. 12, No.1, Winter 1993.

Haveman, Heather, "Between a Rock and a Hard Place: Organizational Change and Performance Under Conditions of Fundamental Environmental Transformation," *Administrative Science Quarterly*, Vol. 37(1), 1992, pp. 48-69.

Haggerty, Patrick E. " Industrial Research and Development, " Chapter 10 in *Science and the Evolution of Public Policy*, Shannon, James (ed.) New York: Rockefeller University Press, 1973.

Hague, Doug, *Description of a Turbofan Engine Product Development Process*, MIT Master's Thesis, Cambridge: Massachusetts Institute of Technology, 2000.

Hartmann, George and Myers, Mark B. "Technical Risk, Product Specifications, and Market Risk," in *Taking Technical Risk*, by Branscomb, Lewis and Auerswald, Philip, Cambridge; The MIT Press, 2001, pp. 30-43.

Hax, Arnoldo C. and Wilde, Dean, *The Delta Project*, New York: Palgrave, 2001.

Hax, Arnoldo C. and Majluf, Nicolas S. *The Strategy Concept and Process: A Pragmatic Approach*, New Jersey: Prentice Hall, 1991.

Heany, Donald, *Cutthroat Teammates*, Homewood, Illinois: Dow Jones-Irwin, 1989.

Hekmatpour, Sharam and Ince, Darrel, *Software Prototyping, Formal Methods and VDM*, Reading, Mass: Addison-Wesley Publishing Company, 1988.

Helo, Petri, et. al. "Software Process Structures, A System Dynamics Analysis," Presented at the Third MIT Design Structure Matrix Workshop, Massachusetts Institute of Technology Center for Innovation in Product Development, Oct. 29, 2001.

Henderson, R. "Managing Innovation in the Information Age," *Harvard Business Review*, January-February, 1994, pp. 100-105.

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

High, Jack, *Regulation: Economic Theory and History*, Ann Arbor: University of Michigan Press, 1991.

Herbsleb, James D. and Moitra, Deependra, "Global Software Development," *IEEE Software*, March/April 2001, pp 16-20

Iansiti and MacCormack, "Developing Products on Internet Time," *Harvard Business Review,* September-October 1997, pp. 108-117.

Jachimowicz, Felek, et. al., "Industrial-academic Partnerships in Research," Chemical Innovation, Sept. 2000, pp. 17-20.

Jootar, Jay, "A System Architecture-based Model for Planning Iterative Development Processes" Singapore-MIT Alliance Paper, Center for Innovation in Product Development, January 2002.

Jootar, Jay, *A Risk Dynamics Model of Complex System Development,* MIT Doctoral Thesis, Cambridge: Massachusetts Institute of Technology, 2002

Judd, C.M., et. al. *Research Methods in Social Relations, 6th ed.* Fort Worth: Harcourt Brace Jovanovich, 1991.

Kleim, Ralph and Ludin, Irwin, *Reducing Project Risk*, Gower: Brookfield, Vermont, 1997.

Krubasik, Edward G. "Customize Your Product Development," *Harvard Business Review*, November-December, 1998, pp. 4-9.

Loch, Christoph, "On Uncertainty, Ambiguity and Complexity in Project Management," MIT Operations Management Seminar, Cambridge, Massachusetts, May 20, 2002.

MacCormack, Alan, *HBS Multimedia Case Studies: Microsoft Office 2000*, Boston: Harvard Business School Press, Case 600-023, 2000.

MacCormack, Alan, "Managing the Sources of Uncertainty: Matching Process and Context in New Product Development", Harvard Business School Working Paper 00-078. (Forthcoming in *JPIM*) http://www.hbs.edu/dor/abstracts/9900/00-078.html

MacCormack, A., Verganti R., and Iansiti, M., "Developing Products on Internet Time: The Anatomy of a Flexible Development Process." *Management Science* 47, No. 1 (January 2001)

MacCormack, Alan. "Towards a Contingent Model of the New Product Development Process: A Comparative Empirical Study." *Harvard Business School Working Paper Series*, No. 00-077, 2000.

MacCrimmon, Kenneth R. and Wehrung, Donald A. *Taking Risks: The Management of Uncertainty*, New York: The Free Press, 1986.

Maier, Mark W. & Rechtin, Eberhardt, *The Art of System Architecting, 2nd Ed.* CRC Press, 2000.

Management Roundtable, *Product Development Best Practices Report*, "Pratt and Whitney Moves from In-House Software Solutions to an Integrated, Vendor-Developed Approach to PDM," Waltham MA: Management Roundtable Inc. 1999. (Online version, 2002)

Mann, Charles C. "Getting Over Oil," *Technology Review*, January/February 2002, Vol. 105, No. 1, pp.33-38

McConnell, Steve, *Rapid Development: Taming Wild Software Schedules*, Ch. 7: Lifecycle Planning, Redmond: Microsoft Press, 1996.

McGrath, Michael E. *Product Strategy for High-Technology Companies, 2nd Ed.* New York: McGraw-Hill, 2001.

Meade, Laura and Presley, Adrien, "R&D Project Selection Using the Analytic Network Process," *IEEE Transactions on Engineering Management*, Vol. 49, No. 1, February 2002, pp. 59-66.

NASA, National Aeronautics and Space Administration, Ames Commercial Technology Office, *Spinoff 2000,* 2000.

NASA, National Aeronautics and Space Administration, Ames Commercial Technology Office, *Partnership Options for NASA and Industry*, Handbook, 2003.

National Research Council (Standing Committee to Review the Research Program of the Partnership for a New Generation of Vehicles, Board on Energy and Environmental Systems, Commission on Engineering and Technical Systems, Transportation Research Board), *Review of the Research Program of the Partnership for a New Generation of Vehicles. Fourth Report,* ISBN: 0-309-06087-7, National Academy Press, 1998.

Nelson, Richard, *National Innovation Systems: A Comparative Analysis*, New York: Oxford University Press, 1993.

Nelson, R. "Why should managers be thinking about technology policy?" *Strategic Management Journal*, Vol. 16, 1995, pp. 581-588.

Nevins, James L. et. al. *Ford Motor Company's Investment Efficiency Initiative: A Case Study*, Alexandria, Virginia: Institute for Defense Analysis, IDA Paper P-3311, April 1999.

New York Times, *"Company News: Xerox to Eliminate about 530 jobs in Rochester Area,"* Section C, Page 4, Column 1, January 24, 2002.

Nichols, Nancy A. "Scientific Management at Merck: An Interview with CFO Judy Lewant," *Harvard Business Review*, January-February, 1994, pp. 88-99.

Ort, Robert; Kelly, Colleen, and Hotchkiss, Marlow, *Lakes, A Journey of Heroes*, (The Human Side of Xerox Product Development), Webster and Santa Fe: Xerox Corporation and LivingSystems, Ltd. 1997.

Osbourne, Sean M. *Product Development Cycle Time Characterization Through Modeling of Process Iteration*, MIT Masters Thesis, Cambridge: Massachusetts Institute of Technology, 1993.

Otto, Kevin and Wood, Kristin, *Product Design*, New Jersey: Prentice Hall, 2001.

Pahl, G. and Beitz, W. *Engineering Design, A Systematic Approach, 2nd Ed.* London: Springer, 1996.

Pindyck, Robert; Rubinfeld, Daniel L. *Microeconomics, 4th ed.* New Jersey: Prentice Hall, 1998.

Porter, Michael, "Towards a Dynamic Theory of Strategy, *Strategic Management Journal*, 1991, Vol 12, pp. 95-117.

Porter, M. "From Competitive Advantage to Corporate Strategy," *Harvard Business Review*, 1987, pp. 43-59.

Porter, Michael and van der Linde, Claas, "Toward a New Conception of the Environment-Competitiveness Relationship," in Jasanoff, Shiela (ed.) *Comparative Science and Technology Policy*, Lyme, NH: Edward Elgar Publishing, 1997, pp. 513-534.

Pepin, Ronald, *Application of Critical Chain to Stages Software Development*, MIT Master's Thesis, 1999.

Rosenau, Milton D. and Moran, John J. *Managing the Development of New Products, Achieving Speed and Quality Simultaneously Through Multifunctional Teamwork*, New York: Van Nostrand Reinhold, 1993.

Rosenbloom, Richard and Cusumano, Michael, "Technological Pioneering and Competitive Advantage: The Birth of the VCR Industry," *California Management Review*, Vol. 29, No. 4, Summer 1987, pp. 51-76."

Schumpeter, Joseph, *Capitalism, Socialism, and Democracy*, New York: Harper and Brothers, 1942, p.83.

Siegel, Donald; Waldman, David; Silberman, Jonathan, and Link, Albert, *Assessing the Impact of Organizational Practices on the Performance of University Technology Transfer Offices: Quantitative and Qualitative Evidence*, National Bureau of Economic Research (NBER) Conference on Organizational Change and Performance Improvement, Santa Rosa, CA, April 23, 1999. (Also NBER Working Paper 7256)

Swanekamp, Robert, "Raising the reliability of advanced gas turbines," Power, Vol. 146, No. 2, March/April 2002, pp. 24-34.

Shiba, Graham and Walden, *A New American TQM*, Cambridge: Productivity Press, 1993.

Smith, Preston G. and Reinertsen, Donald G. "Shortening the Product Development Cycle," *Research-Technology Management*, May-June, 1992, pp. 44-49

Solo, Sally, "How to Listen to Consumers," *Fortune*, January 11, 1993, pp. 77-78.

Stein, Hank, "Deregulation and Its Effect on Gas Turbines, *Turbomachinery International*, Vol. 38, No. 7, November/December 1997, pp. 32-35

Teece, D.J. (1987) "Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing, and Public Policy," *The Competitive Challenge*, ed. D. Teece, Ballinger Publishing, Cambridge, Ch. 9, pp. 185-219.

Teece, D. "Capturing value form knowledge assets: the new economy, markets for know-how, and intangible assets," *California Management Review*, 1998.

Trumbull, J. Gunnar, *The Politics of Product Market Regulation and its Impact on Product Innovation*, Doctoral Disseration, Cambridge: Massachusetts Institute of Technology, 1998.

Ulrich, Karl T. and Eppinger, Steven D. *Product Design and Development, 2nd ed.* Boston: McGraw Hill, 2000.

Usher, Abbott P. *A History of Mechanical Inventions*, Revised Edition, New York: Dover Publications, Inc., 1954.

US DOE/Department of Energy, "Advanced Turbine Systems: The Next Generation of Gas Turbines" Federal Energy Technology Center, http://www.fetc.doe.gov. Also, published document by same title through US DOE, Federal Energy Technology Center.

US DOE/Department of Energy, Energy Information Administration, 1995
Utility Data: Form EIA-860, "Annual Electric Generator Report,"
Nonutility Data: Form EIA-867, "Annual Nonutility Power Producer Report,"

US DOE/ Department of Energy, Energy Information Administration:
ftp://ftp.eia.doe.gov/pub/energy.overview/aer/aer8-8.txt Table 8.8. (1998)

US DOE/Department of Energy, Energy Information Administration, Inventory of Electric Utility Power Plants in the United States 2000, March 2002 DDOE/EIA-0095 (2000), http://www.eia.doe.gov/cneaf/electricity/ipp/ipp_sum.html

US FAA/Federal Aviation Administration, Press Release 10/7/1995, "New Oceanic Air Traffic Control Technology Becomes Operational" http://www.faa.gov/and/and300/datalink/news/press3.htm (2002)

Unger, Darian, *Energy Policy and Environmental Technology: The Development of Natural Gas Turbine Technology in Power Generation*, Cambridge: Massachusetts Institute of Technology Master's Thesis, 1999.

Unger, Darian, *Management of Engineering Innovation in the Power Turbine Industry*, 2000 North American Power Symposium, Waterloo, Canada, IEEE Power Engineering Society, 2000. Vol. 1, Ch. 3, pp. 20-27.

Unger, Darian, "Information Technology Impacts on the US Energy Demand Profile," *E-Vision 2000 – Electronics and Energy Use*, US Department of Energy, October 2000.

Unger, Darian, *Energy policy and Environmental Technology: The Development of Turbine Technology in Power Generation*, 5th Annual Conference on Technology Policy and Innovation, Innovation and Environment Section, Delft, The Netherlands, June 2001.

Van Haste, Clara, "What's Happening to the Gas Turbine Market?" *Electrical World*, Vol. 210, No. 11, November 1996, pp. 61-64.

Von Hippel, Eric, *The Sources of Innovation*, New York: Oxford University Press, 1988.

Ward, Allen, et. al. "The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster," *Sloan Management Review*, Vol 36, Issue 3, Spring, 1995, pp.43-61.

Watson, W. J. *"The 'Success' of the Combined Cycle Gas Turbine" Opportunities and Advances in International Power Generation,* University of Durham Conference Publication, March 18-20, 1996, London: Institution of Electrical Engineers, 1996.

Weston, Kenneth C., *Energy Conversion*, New York: West Publishing Company, 1992.

Wheelwright, S. and Clark, K. "Creating Project Plans to Focus Product Development," *Harvard Business Review*, March-April, 1992, pp. 70-82.

Whitney, Daniel E. *Physical Limits to Modularity*, MIT Engineering Systems Division Internal Symposium, Massachusetts Institute of Technology, May 29-30, 2002, pp. 527-543.

Womack, James P. et. al. *The Machine That Changed the World, How Japan's secret weapon in the global auto wars will revolutionize Western industry*, New York: Harper Perennial, 1991

Woodward, Charles, "Computer Aided Industrial Design for Ceramics and Glass Industries," Deskartes website, www.deskartes.com, 2003.

Xerox Corporation, *Xerox Corporation Annual Report* and SEC Forms 10-K for 2000 and 2001, Stamford: Xerox.

Xerox Corporation – Newsroom, "Xerox Leverages History of Adapting Products for Disabled to Meet New Federal Requirements, " News Release, Stamford: Xerox, June 25, 2001.

Xerox Corporation CEHSS (Xerox Customer Environment, Health and Safety Support), *2001 Environment Health and Safety Highlights Report*, Webster NY: Xerox, 2001

Yassine, Ali, and Whitney, Daniel, *Do-It-Right-First-Time Approach to Design Structure Matrix Restructuring,* Proceedings of DETC '00: ASME 2000 International Design Engineering Technical Conferences, Sept. 2000, DETC2000/DTM-14547

Yassine, Ali, et. al. "A Decision Analytic Framework for Evaluating Concurrent Engineering," *IEEE Transactions on Engineering Management*, Vol 46, No. 2. May 1999.

Yoffie, David and Cusumano, Michael, "Judo Strategy: The Competitive Dynamics of Internet Time," *Harvard Business Review*, 1999.

Young, Peter C. and Tippins, Steven C. *Managing Business Risk, An Organization-Wide Approach to Risk Management*, Boston: AMACOM/American Management Association, 2001.

APPENDIX A: QUESTIONNAIRE AND INTERVIEW EXAMPLES

Printco interview guide:
November/December 2002
(Interviewed 14 engineers and managers)

What was the budget and schedule of this development effort?

Who is on the core design team? How are they organized?

What were the major technical risks for this product?

What were the major market risks for this product?

How dependent is this product on earlier Printco products, components, or knowledge?

How much design work occurred before sanctioning is complete?

Can you describe in detail a big design problem or change in the product during development? Which problem or challenge was the biggest?

How do the reviews work? Does process control reviews?

What is the success rate at design reviews?

Who constitutes design review team?

How effective is your change control process? How much feedback do you get from customers? How early/late do integration efforts occur?

When and where are the prototypes evaluated?

Does one design group usually have to wait for another? Where do the schedule delays occur?

How has your PDP changed over time? (Over the last 5-10 years?)

What are you trying to improve in your process?

How has quality control and MTBF improved?

How successful is the final product? Technically? In the market?

Are there specific design for manufacturing or design for cost efforts?

Compare this project to another project...how much do the people matter? Does the PDP make different projects more alike or consistent with each other?

Center for Innovation in Product Development
April, 2001

PRODUCT DEVELOPMENT PROCESS SURVEY FOR XEROX – "CLEAN SHEET" PROJECT WITH SOFTWARE ORIENTATION

Background: This research is being conducted by the Center for Innovation in Product Development (CIPD), a research center at the Massachusetts Institute of Technology in Cambridge, Massachusetts. CIPD is sponsored by Xerox and has a nondisclosure agreement regarding the information acquired here. The goal of this study is a comparison of product development processes between different parts of Xerox and between Xerox and other companies. For any questions or comments, please contact Darian Unger at Unger@mit.edu or (617)-253-4735. Thank you for your prompt attention and assistance.

Please send response to Unger@mit.edu or to
Darian Unger
Center for Innovation in Product Development
Office E60-246
Massachusetts Institute of Technology
Cambridge, MA 02139

Name(s) of person/people completing this survey _____

Email address(es) _____

Phone number(s) _____

Basic description of project:
1. Please outline the main function of the product (i.e. Endeavor controller for use in document center)

2. Please list the major features that make this a "clean sheet" product.

3. Please list any major architectures or features retained from earlier products.

4. Were the architectural specifications for this product significantly different than for previous versions?
Please explain.

5. Approximately when was the product or system delivered? (approximate month and year)

Size of project:

6. What is the total time required for project (in weeks)? _____

7. What are the total person-hours required for project? _____

8. What was the development budget? ($) _____

9. What was the approximate size of the completed software product (in lines of code)?
_____

9a. If this figure includes comments, what was the percentage of comments?
_____

10. What was the division of labor and resources among development team? (Please fill out your choice of column – only one column is necessary.)

	Staff resources (person-years)	**Budget ($)**	%
Project management			
Architectural and high-level design			
Detailed design/programming			
Testing, QA, and integration			
Other (please explain)			
Total			Should total 100%

Design reviews:

11. What were the dates of the design reviews?

12. Did any reviews lead to rejection or rework? If so, please explain briefly.

13. Please provide copies of review documentation (Transfer Advisory Team assessments) for the 3.1, 3.2, 3.3, and 3.4 reviews. (Please send with this completed survey to Unger@mit.edu. If your response is not electronic, please mail it to:
Darian Unger
Center for Innovation in Product Development
Office E60-246
Massachusetts Institute of Technology
Cambridge, MA 02139

Design changes, prototypes, and testing:

14. What was the origin of the software code in the finished release?

	%
Off-the-shelf code retained from previous version of the same product.	
Off-the-shelf code from other sources	
New code developed for this product	
Other (please explain)	
Total	Total should be 100%

15. Were any major features included in the product that were not originally planned for?

15a. If yes, at what cost did those additional improvements come? (i.e. schedule lag, increased budget, reduction of pad-time)

15b. If yes, at what point in the development process did those changes or feature additions become necessary? (i.e. middle of time-to-market phase 3.3, etc.)

16. What percentage of the features that were implemented in the final product were contained in the original functional specification?

_____

17. What percentage of features in the final product are new features that were not on the original features list or that were changed due to market or technical feedback?

_____

18. During the design stage, how frequently was the system "built?" (i.e. how often were design changes, including bug fixes, integrated into the code base and recompiled, e.g. twice per week, once per month, once prior to 3.3 review, etc.)

19. How many cycles were there in phase 3.3?

20. What was the relative emphasis on different types of testing during the project?

	% of total testing time
Focus on testing	
Component testing	
Integration testing	
System testing (complete product)	
Other (please explain)	
Total	Total should be 100%

21. How many prototypes were made?

22. How many prototypes were tested with customers? If prototype was partial, please explain.

23. For the following question, please use a 5-point scale where 1= poor, 2=worse than expectations, 3=met expectations, 4=exceeded expectations, 5=excellent
Please indicate the extent to which you perceive the project met expectations in terms of:

Schedule performance _____
Budget performance _____
Customer satisfaction with end product _____
Financial returns from product as a whole _____

24. If rework occurred, what percentage was due to:

Rework due to changes in architecture _____
Rework due to changes in specifications _____
Rework due to technical feedback or technological changes _____
Rework due to customer feedback _____
Other (please explain) _____
(Total should be 100%)

25. What was the "cost" of any rework in the project?

In calendar time?

In person-time?

In budget?

On morale?

Opinion:

26. Is the architecture modular or monolithic?

27. Using the following 5-point scale, what is the flexibility of the design team in

incorporating feedback from prototypes or market surveys once design begins?

_____

(1=poor, 2=low, 3=acceptable, 4=good, 5= excellent)

28. What are the major risks in software development at Xerox? Does the Time-To-Market process adequately manage those risks?

28. Please write any additional comments you wish or list any concerns you have about this survey.

Thank you very much for your time and response.

Please send to Unger@MIT.edu or
617-253-4735

Darian Unger
Center for Innovation in Product Development
Office E60-246
Massachusetts Institute of Technology
Cambridge, MA 02139

APPENDIX B: LIST OF PEOPLE INTERVIEWED

Dick Arra (ITT)
Mike Barrett (Xerox)
David Benjamin (Printco)
Frank Bevc (SWPG)
Eileen Blanchette (IDE)
Ralph Brown (IDE)
John C. (Printco)
Daniel Caprioni (ITT)
Jim C. (Printco)
Randall.Cole (Xerox)
Jason D. (Printco)
Rodrigo Dobry (Microsoft)
Anne Donelan (ATS & IDE)
Ed F. (Printco)
Charles G. (Printco)
Jeffrey Gramowski (Xerox)
Doug Hague (UTC)
Rachel Happe (IDE)
Shelly Hayes (Xerox)
Joel Haynes (GE)
Qi Van Eikema Hommes (Ford)
Jim Irvine (ITT)
Chris Kagic (ITT)
Brian Kalita (IDE)
Johanne Korrie (Xerox)
Walt Kreucher (Ford)
David Lackner (NASA)
Dave L. (Printco)
Bob Martino (ITT)
Lee McLurin (SWPG)
David Miller (SWPG)
Richard Moore (IDE)
Ray Narramore (Xerox)
Jonathan Niemeyer (UTC)
Jennie N. (Printco)
Scott P. (Printco)
Dawn Paluszny (Ford)
Joseph P. (Printco)
Patrick Pendell (Xerox)
John Penny (Ford)
Bill Phillips (Ford)
Mike Policano (ITT)
Mike P. (Printco)
John Rajan (ITT)
Elenora Rakover (Xerox)
Michael Ray (Xerox)
George Roller (Xerox)
Tony Russo (ITT)
Neal Salante (ITT)
Joe Schlepko (SWPG)
Anne.Schneider (SWPG)
Mike Simco (ITT)
Bob Smit (ITT)
Kirk Speer (SWPG)
Mike S. (Printco)
Craig Tedmon (GE & ABB)
Graham W. (Printco)
Jim Weisheit (UTC)
Donald Wegeng (Xerox)
Kelly Zechel (Ford)

APPENDIX C: SAMPLE TRANSCRIPT EXCERPT

Below is a sample excerpt from an April, 2001 interview with a Xerox marketing manager. Included are the middle 4 of the 12 transcript pages from one 45 minute interview. Questions are in italics, responses are in block print. For references to the Xerox TTM process steps, see Figure 6.4.2.

So marketing and strategy get together to decide what's going to go into the next product, and that happens for both variant and clean sheet products?
Yes exactly.

What if something is important enough to not wait for the next generation...the next variant...but either it's worth paying extra money to get an extra feature in a current product or possibly in some extreme cases, pushing out a schedule going late just to get a feature in. Does that happen?
That does happen.

How often does that happen?
Not too often

Like 1 in a 100? A couple of times a year?
It depends. Moving up a schedule is like,...there really has to be a shift in the marketplace to do that...like someone has leapfrogged you and you would be at such a disadvantage that it wouldn't be viable from a business standpoint to launch a product. It would have to be pretty severe to get a new feature, something you didn't commit to...if it's a big feature. Now if it's a small feature that the engineering teams believe they can get it in time or if it's a small change in the schedule, I'm talking the difference between a week or a month, it really depends on what the cost/benefit tradeoff is. And there are things, for example, we have in the past and what we do do is where something might not be going to schedule...so that we try to keep the products so that they launch when they're supposed to, especially the variants when were at a little faster pace. What we've done is we launch the product and then come out with what we call a Service Maintenance Pack, and that's kind of a misnomer, because we also put, have in the past put features in it...where a feature is not...when 99% of the things are on schedule, there's this one 1% that's not getting on schedule. It's still very important, but in most cases you don't have a business case to justify slipping the program to get this one feature but its still worth a significant amount of money so you release it in a later release.

Let me ask about the worst case scenario...lets say the specs have been laid out...you're in the [Xerox Time-To-Market Procedure] 3.2-3.3 area...and then you in marketing is the first to discover that there's a significant shift in the market...probably because of competition?
Usually it's competition

So which would be worse...having to add features or having to change features?
Having to add features, usually. Because one of the things we've done is that we've made tradeoffs between features when we defined the spec one way and we've made tradeoffs to still meet the schedule but to limit the amount of changes or amount of (unintelligible) that the marketplace would have and what I mean by that you could make some changes to a feature but still get 95% of that feature and still hit the sweet spot by doing some different things that might save engineering or our supply chain a significant amount of resources. And that's quite common, actually. That's a big part of what I do. Well, not common, but...well,...the devil is the details, right.

You told me with no hesitation that adding features is more problematic than changing features. Why is that?...In some other circumstances, changes create more trouble because of unexpected interactions between components, but that's not what came to your mind...
No...most changes are to simplify something, not to make it more complex...and that's what my thing is about hitting the sweet spot. We could have, performance engineers and systems engineers could have defined something that we'd love to have as a marketing team, but there could be some things on that that we took because we thought there would be no cost or because we can spin something out of it...but there is some give there and that's what I'm saying about changing. We really not adding new features, but we do, we do. And if we do add a feature, I call a systems engineer and they do a change request that you were speaking about. And we make sure that the rest of the...they have a requirements review board, engineering team do...and when I ask for a change, I'm going to ask for a change in the software release that we have in July because I strongly believe we need it...and they're doing an analysis to see if it's indeed possible and if we can still meet July. So that does happen, but that goes though the engineering team process, change request, the system engineer analyzes it to get an idea how much its going to impact everybody. Everybody's included in the RMB, the system engineers, the engineers who are doing the work, the testers, the people in documentation. There's a whole bunch of checkoffs. In that respect, we have a pretty well-defined process.
Yeah...they have a little leeway but not much. Most of our incentives are to launch on

Do you find yourself on program teams that are not your own projects?
Yeah, we have...there is an external team that reviews each of the phase gates to make sure that we do things consistently with Xerox's TTM process. So I would have a counterpart, a marketing manager in a different group who would review my outputs.

I'd like to ask about prototyping and how you view from marketing...Do you do Beta testing with customers?
Yes we do.

Does everything get beta tested with customers?

Pretty much. At least 6-8 weeks...getting feedback is part of marketing. I spend a lot of time trying to recruit customers and then I get the feedback too. I'm not the only one

You have a sales force to do that?

Yeah, the sales force makes the initial customers and they know who the customers are and what they're doing and I go in and communicate what is new and why they want it and what we do for them and what they've got to do. It's a customer engagement/test arrangement? There's a manager that I interface directly with who basically does it. I'm accountable...I sign off on who ends up getting to be a tester.

When you beta test with prototypes in these companies, are they the same prototypes? Are they different levels of prototypes?
They have the same the same prototypes but they have different environments and they use it in different ways.

Let me switch to your opinion on the flexibility of the TTM process. You always get these tradeoffs of flexibility vs. hardening....when you decide to go through on schedule regardless of the changes or features that you want to edit. Do you think that Xerox has it right? Based on most of the projects that you're working on, are comfortable with the way it is?
Yeah, I guess I can't complain. You always want more, but you have to realize or understand that we do have a limited amount of resource. As long as we're doing the things that are the most important and with the highest priority I can live with the ones that are lower down that might not make it.

When you want something in there, you rank this as a very high priority. It goes if necessary to the core group and they weigh you weighting this as a 10 and engineering weighting this as an 8 difficulty, and hopefully it works out in your favor because it's more important to you than it is to them. Is that a fair assessment?
Yes. That's a fair assessment

They're going to weigh your interest against theirs. Do you lobby hard on that front?
Oh for sure. If I really believe that we're either missing out on something and missing out on an opportunity significantly or we're going to be at a significant competitive advantage, it's basically what I call laying on the tracks. I'll lay out on the tracks for that. And one of the things....we are a team.. all the team has to agree to pass or reach the phase gates. I can theoretically, and I could, say "we can't go and here is the reason why. You've got to do this and this. You can't launch because of this." That's when you get a lot of exposure. There might be decision probably with a (garbled) manager saying that they thought that my call was correct or not.

At what stage to these prototypes happen? For example, it would be easier if the feedback that you get from prototypes or beta test...if that happened earlier.
Yes..it's never early enough.

So the earlier a prototype, the earlier a beta test from your perspective, the better. Has there been improvement in doing that or has that been pretty much been the same for the past few years. Better or worse, is there trend?
I think all of the above. There are certain things that are done better and getting out there earlier and getting feedback early. And there are other things that we've in the last few years let slip a bit.

Let me get more specific with prototyping and numbers. Does it happen between 2.2 and 2.3. Where is the first time that you'll get a prototype that you can give to a customer.
Usually it's past 3.3. Passing 3.3 means that it really has all the features/functions that we expect to launch with. Usually that's the case. I have seen cases where we've gone out to customers with a product at 3.2...very friendly customers. But typically not...

But all these prototypes and beta test are with entire machines. You don't take, say, code, and put that into the existing machine of a customer and say 'we're beta-testing on the machine that you've already got'?
Oh yes we do.

Oh, you do that? And that doesn't happen any earlier, though? You'd think that would ease things up, that you could get more prototypes in faster?
Well, since we have our defined process, that code has a TTM process just like a product so we try to get it past 3.3 –2 before we get it the customer. Because basically a 3.3 says that you've done some testing on it, you have all the features in it and its reasonably reliable...reliable enough so that you feel customers could really like it.

On certain products we have...done 3.2. It really depends what the software is. With print drivers, they go out fairly quickly, there's not really an issue there. It depends on the complexity of it...whether a prototype goes out early or not. And what I mean by complexity is not only the complexity of the software , but how complex is it to be supported in our value chain, within our field organization, our technicians, our analysts. Because we have 15,000 technicians out in the field and 5000 sales people and 4000 analyst and you have to communicate to anyone who's going to be in touch with that product (garbled) be sure that they know how to handle it. But it depends on the complexity. We have a customer that requires a certain driver in a certain print environment we push that out relatively quickly...very quickly.

APPENDIX D: TERMS, ACRONYMS AND ABBREVIATIONS

AA	Appearance Approval
ABB	Asea Brown Boveri
ATS	Aviation Technology Systems *or* Advanced Turbine System
BACT	Best Available Control Technology
Btu	British Thermal Unit
CAD	Computer Aided Design
DARPA	Defense Advanced Research Planning Agency
DOD	Drop on Demand *or* Department of Defense
DOE	Department of Energy
CIPD	Center for Innovation in Product Development
CC	Combined Cycle *or* Change Cut-off
CDR	Critical Design Review
CP	Confirmation Prototype
CPID	Color Printing and Imaging Division
CT	Combustion Turbine (simple cycle)
DOE	Department of Energy
EC	European Community
EIA	Energy Information Administration
EPA	Environmental Protection Agency
FAA	Federal Aviation Administration
FANS-1	Future Air Navigation System
FBN	Fuel Bound Nitrogen
Fig.	Figure
FPDS	Ford Product Development System
G#	Gate number
GE	General Electric
GPS	Global Positioning Satellite
GT	Gas Turbine
GTB	Grounded Theory Building
GUI	Graphical User Interface
GW	Gigawatt
HRSG	Heat Recovery Steam Generator
HW	Hardware (including all non-electronics)
IDe	Integrated Development Enterprise
IEEE	Institute of Electric and Electronic Engineers
IGCC	Integrated Gasification Combined Cycle
IMP	Integrated Management Plan
IPD	Integrated Product Development
IPP	Independent Power Producer
ISO	Independent System Operator
ITT	ITT Industries, Inc.
J1	Job 1
kW	Kilowatt
kWh	Kilowatt-hour
LAER	Lowest Achievable Emissions Rate
LHV	Lower Heating Value
MHI	Mitsubishi Heavy Industries
MS	Microsoft
MW	Megawatt
MWh	Megawatt-hour
NASA	National Aeronautics & Space Administration
NGPA	Natural Gas Policy Act
NO_x	Oxides of Nitrogen
OECD	Organization for Economic Cooperation & Development (Industrialized nations)
O&M	Operation and Maintenance
P&W	Pratt and Whitney
PA	Program Approval
PC	Personal Computer
PCB	Printed Circuit Board
PD	Product Development
PDCA	Plan, Do, Check, Act
PDP	Product Development Process
PDR	Preliminary Design Review
TRR	Test Readiness Review
PR	Product readiness
PUHCA	Public Utilities Holding Company Act
PURPA	Public Utilities Regulatory Policy Act
QA	Quality Assurance
R#	Review number
R&D	Research and Development
RIT	Rotor Inlet Temperature
RCSA	Reverse Case Study Analysis
SECD	Securities and Exchange Commission
SI	Strategic Intent
SO_x	Oxides of Sulfur, often SO_2
Spec	Specification
SUO	Small Unit Operations
SW	Software
SWPG	Siemens Westinghouse Power Generation
TQM	Total Quality Management
TTM	Time to Market
UTC	United Technologies Corporation