Comparing Product Development Processes and Managing Risk

Comparing product development processes and managing risk

Darian W. Unger*

School of Business
Howard University
2600 6th St. NW, Office 453
Washington, DC 20059, USA
Fax: 1+(202)–806–1642
E-mail: dwunger@howard.edu
*Corresponding author

Steven D. Eppinger

Sloan School of Management
Massachusetts Institute of Technology
50 Memorial Drive, Room E52-474
Cambridge, MA 02142, USA
E-mail: eppinger@mit.edu

Abstract: Product Development Processes (PDPs) require careful design to reduce development time, create better products and manage the risks of bringing new products to market. This paper investigates the relationship between product development risk and PDP management. We begin by identifying risks and proposing several iteration- and review-based metrics by which PDPs can be effectively identified and compared.

Data from ten company case studies demonstrate the utility of the proposed metrics and exemplify how different PDPs manage different risks. The cases also show that software companies face different risks and employ more flexible PDPs than manufacturing companies. We conclude that PDPs vary more than previously documented, that the proposed metrics are useful in distinguishing PDPs and that companies can tailor their PDP designs to suit their unique risk profiles.

Keywords: Product Development Process; PDP; risk management; innovation; technology management.

Reference to this paper should be made as follows: Unger, D.W. and Eppinger, S.D. (2009) 'Comparing product development processes and managing risk', *Int. J. Product Development*, Vol. 8, No. 4, pp.382–402.

Biographical notes: Dr. Darian W. Unger earned his PhD from the Massachusetts Institute of Technology (MIT) and is an Assistant Professor at the Howard University School of Business. His research and teaching interests include project management, technology strategy and new product development in the energy industry.

Dr. Steven D. Eppinger is the Deputy Dean and Professor of Management Science at the MIT Sloan School of Management. He also holds the General Motors Leaders for Manufacturing Chair and has a joint appointment in the

MIT's Engineering Systems Division. Prof. Eppinger is the co-author of *Product Design and Development*, a widely used textbook published by McGraw-Hill. Prof. Eppinger holds SB, SM and ScD degrees in Engineering from the MIT and has published articles in *Management Science, IEEE Transactions on Engineering Management*, the *Journal of Mechanical Design, Research in Engineering Design* and *Harvard Business Review*.

1 Introduction

Successful Product Development (PD) is critical to industrial performance. Rapid and innovative PD can provide critical competitive advantages to firms (Jachimowicz *et al.*, 2000; Ulrich and Eppinger, 2004). Despite the importance of PD, companies currently have difficulty designing or choosing from an extensive array of PD processes. If companies design their processes 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 PD processes, nor is any single process ideal for all circumstances and companies.

This article explains a variety of Product Development Processes (PDPs) and aims to help companies better design their own PDPs. A literature review reveals that current categorisations of PDPs are insufficient for effective management comparison or application. Using a combination of existing literature and case study research, we propose that two risk management activities – development iterations and reviews – can be used as metrics to describe and compare different PDPs. We then use the case studies to examine the variation among PDPs and to demonstrate the utility of the proposed metrics.

2 Background and literature review

Published literature and industry practice in product development management provide useful background for this study. This review examines PDP characteristics and explains different PDPs.

2.1 PDP steps and risks

PDPs are not uniform, but they often use similar actions to manage development risks. Prevailing literature and industry practices present PDPs that involve a common series of actions, steps or stages. Most companies follow at least some form of the following steps: product planning, project planning, concept creation, system-level design, detailed design, testing/prototyping, and release. The purpose of PDPs that include these steps is to provide a structure for managing the many uncertainties and risks that companies face. Segmenting the process into smaller actions is one way of controlling risks.

Risk management is a fundamental PD concern because risk, defined as exposure to danger or loss, is prevalent in all development projects. Balancing risks and potential rewards is an enduring theme of engineering and programme management (Ansell and

Wharton, 1992; Foster and Kaplan, 2001; MacCrimmon and Wehrung, 1986). The risks of PD can lead to several forms 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 fulfil customer needs and therefore completely miss a market niche (Awny, 2006).

Existing literature suggests several ways of categorising PD risks. This research uses a traditional categorisation of risk by source of uncertainty underlying the risk (Cross and Sivaloganathan, 2005). A successful PD process should be able to manage or mitigate the following four major types of risk:

1 Technical risk – uncertainty regarding whether a new product is technologically feasible and whether it will perform as expected, given clear and valid product specifications.

2 Market risk – uncertainty regarding whether a new product accurately addresses changing customer needs and whether the product is well positioned relative to competition. Unlike the technical difficulty of building ‘to a specification’, market risk concerns whether an achievable specification brings the wrong product to market.

3 Schedule risk – uncertainty regarding whether a new product can be developed in the time available.

4 Financial risk – uncertainty regarding whether a new product can be developed on budget and whether the project will pay back the investment.

These four general types of risk 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 (Bstieler, 2005). For example, quality assurance or integration risk may be considered 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 tends to increase technical, schedule, and budget risks. It is therefore impossible to completely separate the types of risks faced in PD, although the categorisations are useful in planning PDPs.

Prior research explores the roles, categorisations, and management of risk. De Meyer *et al*. organise risk by type and warn of the need to observe these risks carefully in order to improve project and development management (De Meyer *et al*., 2002; Hartmann and Myers, 2001). More general risk literature stresses the importance of maximising expected values and introduces traditional risk management methods such as hedging, decision analysis, and parallel development (Ansell and Wharton, 1992; De Neufville, 1990). Other sources point towards the importance of managing information flows in mitigating risks and improving design efficiency (Varma Citrin *et al*., 2007; Blanco *et al*., 2007).

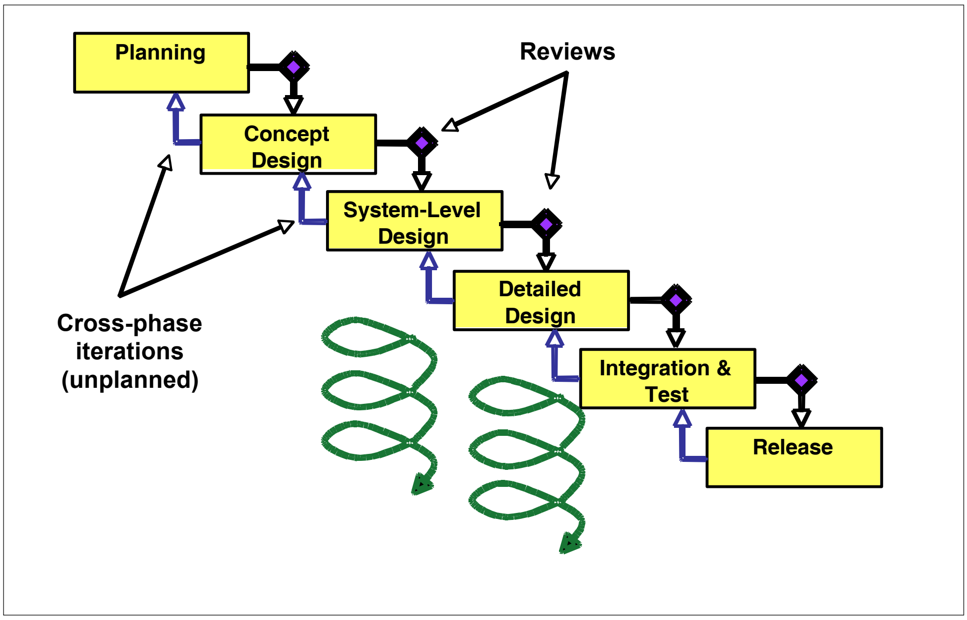
Building on this literature, we will consider how different PDPs address risk through product development iterations, integrations, and reviews. Iterations often address market risk while reviews often address technical risk. Planned iterations – often in the form of early prototypes, simulations, or analytical models – provide feedback for improved design. As later sections show, the cost, time, and fidelity or quality of integrations vary widely across industries. For example, some hardware-based prototypes are difficult or expensive to build because they require tooling, construction, and complex electro-mechanical integrations. In contrast, computer-based, soft prototypes may be easier to build and integrate, but may not provide as much quality feedback if the computer models do not capture key real-world aspects of the product. PD managers must weigh the benefits and costs of system 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 system integrations, tests, and feedback generally improves the evolving product. Similarly, performance validations and testing are critical to reducing companies' technical risks (Boehm, 1988; Otto and Wood, 2001; Cooper, 2001). Both integrations and validations manage risk, although the risks they manage are often different.

2.2 *The spectrum of PDPs*

PD literature provides many examples of how companies manage development risks. This section presents and describes two common PDPs that constitute the two ends of a spectrum of PDPs. At one end of the spectrum is the staged process, the traditional and dominant PDP in American industry. The spiral process, at the other end, incorporates cross-phase iteration and is commonly used in the software industry.

The most widely used type of PDP, and the standard for comparison in this research, is the traditional staged process shown in Figure 1 (Cooper, 2001; Smith and Reinertsen, 1992; Ulrich and Eppinger, 2004). This process, also called waterfall, stage gate, phase gate, toll gate, checkpoint, life cycle, or structured PD by various authors and practitioners, has been dominant in US industry for almost 30 years. Variants of this process have also evolved into design-to-schedule and design-to-budget processes (McConnell, 1996).

Figure 1 The traditional, staged PDP (see online version for colours)



The ideal staged process proceeds in distinct 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 review, or gate, to evaluate whether the previous phase was successfully completed. If the project is reviewed positively, work proceeds to the next phase. If not, then the project iterates within that phase until it can successfully pass the review or the project may be terminated.

In Figure 1, the reverse arrows, or cross-phase iterations, indicate that it is possible to revisit earlier phases, but such iterations are difficult and costly. These major unplanned iterations are generally avoided whenever possible. Instead, most iterations occur within stages; these narrow iterations result in both advantages and disadvantages.

One major advantage is that staged processes impose structure on development by reaching sharp product definitions and specifications early in PD. Technical risk is reduced because narrow iterations and reviews freeze specifications early. Rigid specifications and stable product definitions help to avoid errors by avoiding midstream corrections.

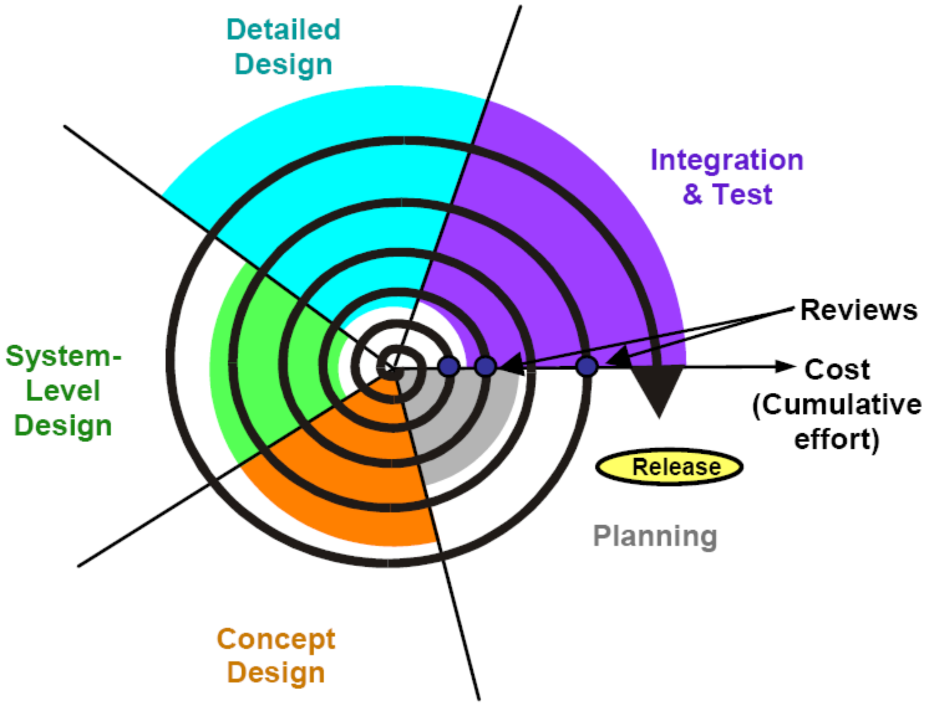
Staged processes perform well in cases when products have stable product definitions and use well understood technologies (as in the case of upgrades or maintenance improvements to existing product). In these cases, staged processes help to find errors in the early stages of a project, when costs of changes are low (McConnell, 1996). Staged processes also work 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 specification freeze and no subsequent changes that increase the likelihood of mistakes.

Inflexibility is the main disadvantage of narrow iterations constrained within phases. Because they do not cross phase boundaries, narrow iterations cannot incorporate feedback from later phases. It is 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 specifications and assumptions are proven wrong by subsequent market research or prototyping. The staged process does not handle these midstream changes well and can be ill-suited for projects in which requirements are poorly understood in the beginning. They may also face difficulty handling parallel tasks within stages (Smith and Reinertsen, 1992).

The spiral PDP differs from the staged process because of its emphasis on flexibility and comprehensive iteration. Unlike the staged processes, the spiral process includes a series of planned iterations that span several phases of development. It is a relatively recent PDP that has been adopted by many in the software industry, with its notably different design requirements (Easingwood *et al*., 2006). 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. As shown in Figure 2, its spiralling form repeats regular steps, including concept development, system level design, detailed design, and integration and testing. The process is flexible; the actual number and span of loops can vary.

Figure 2 The spiral PDP (see online version for colours)



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 (Boehm, 1988).

A simple spiral process with minimal uncertainty and only one loop would closely resemble a staged 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 staged processes. 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 staged process.

The spiral process has several disadvantages. First, it is more sophisticated and complex than other processes, and thus requires more management attention. Second, the lack of rigid specifications can potentially lead to delays in developing complex subsystems. Finally, the spiral process may be overkill for simple projects that could use a simpler waterfall process (Boehm and Bose, 1994).

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 staged processes, and unlike the large but unplanned and unwanted feedback loops which can occur in less successful staged 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 staged process if the milestones and deliverables between each spiral round act merely as stage reviews.

Recent literature sources often recognise PDPs as risk management structures but often focus on one process rather than comparative PDPs. For example, Cooper argues persuasively for the effectiveness of the stage gate PDP (Cooper, 2001). Other sources, including those general sources mentioned in the beginning of this section, implicitly endorse this point of view (Pahl and Beitz, 1996; Smith and Reinertsen, 1992). Boehm advocates the use of the spiral process in software development, and is joined by others who denounce the deficiencies of rigid waterfall processes and call for flexible prototyping (Boehm, 1988; Boehm and Bose, 1994; Gilb, 1988; Hekmatpour and Ince, 1988). This stream of PD literature is strengthened by many studies of individual companies' PD efforts, ranging from software designers to automobile manufacturers (Cusumano and Selby, 1995; Cusumano and Nobeoka, 1998; MacCormack, 2000; Ward *et al*., 1995).

Finally, some sources begin to compare different PDPs. Krubasik (1998) argues for the need to customise 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 offer brief and balanced comparisons of different PDPs, but limit the scope to theoretical examples (McConnell, 1996). Finally, some sources use comparative empirical studies to suggest a method of matching PDPs and context, but do not relate these to PD success (MacCormack, 2000; MacCormack *et al*., 2001).

2.3 PDP variation and problem definition

Staged PDPs facilitate managerial control while spiral PDPs allow more flexibility, and there are many other PDP variations that fall between these extremes. The array of variants includes modified staged processes, evolutionary prototyping and delivery processes, and design-to-schedule or design-to-budget processes. Each of these PDP variants has distinct advantages and disadvantages, but PDP differences are poorly understood and not yet fully acknowledged in existing literature and practice. As a result, companies have difficulty designing or selecting PDPs.

Our research has two goals, both of which help to bridge the knowledge gap in existing literature and industrial decision making. First we seek to identify different PDPs and establish that variety exists. To do so, we define parameters that allow for evenhanded comparisons between PDPs. Second, we demonstrate how different PDPs can address different risks through integrations, iterations, and reviews. Our overall research goal is to help academics and business managers with the difficult task of identifying, comparing, and successfully designing PDPs for risk management.

3 Characteristics for specific PDP comparison

This section proposes characteristics by which different PDPs can be defined, compared, and contrasted. Companies try to balance structure and flexibility in their PDPs, but have difficulty measuring degrees of either structure or flexibility. Characterising 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 between design groups or project phases. Characterising 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. These two characteristics are useful metrics for PDP comparison because all PDPs include some combination of iterations and reviews.

3.1 Design/Integrate/Test cycles

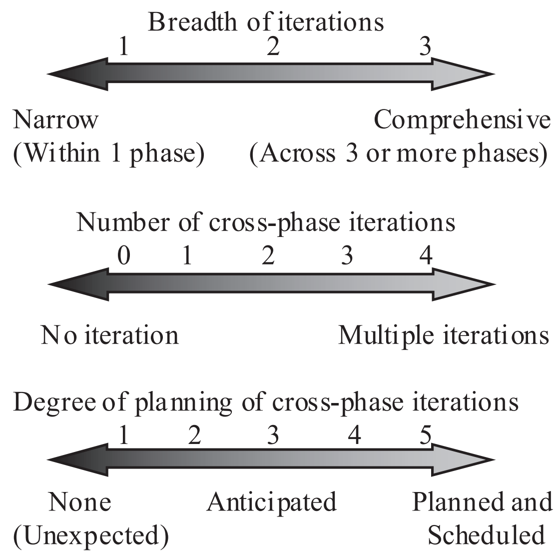
We focus our attention on the design-build-test-redesign and the design-analyse-redesign cycles (the hard and soft forms of design iteration, respectively). Given the uncertainties inherent in PD, iteration is inevitable and must be managed effectively. Iteration is technically defined as the repetition of an action or process. This definition can be perceived positively (as in renewal and improvement) or negatively (as in wasteful repetition). Our 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 methods of iteration that allow feedback from early versions of products.

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.

Iterations in PD 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. These three parameters are shown in Figure 3, along with the scales we use to measure each one.

The first parameter, the breadth or scope of iteration, is a critical descriptor of a company's PDP. Breadth can range from narrow to comprehensive. Narrow iteration is within phases, exemplified by several rounds of interdependent detailed design tasks. Comprehensive iteration is across phases, exemplified by processes that cycle not just around a specific stage, but rather over a range of process stages from concept to prototyping.

Figure 3 Three parameters for measuring PDP iterations



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 cross-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 cross-phase iterations are planned also varies. Processes 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 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 cross-phase cycles may be planned, may be subject to time and budget constraints, or may be dependent on customer satisfaction and quality assurance.

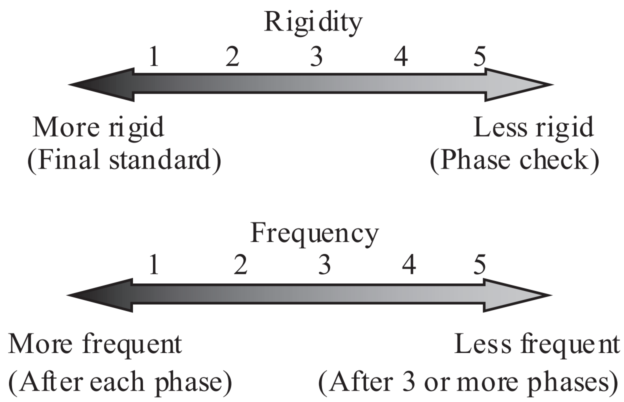
3.2 Design reviews

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

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 peers from other projects. The level of formality of the reviews also varies dramatically.

Figure 4 shows two parameter scales, rigidity and frequency, which we use as metrics to characterise design reviews. Rigidity of review is defined by the degree to which deliverables are held to previously-established criteria. In a rigid review, a project is probed for problems and not allowed to continue until each deliverable meets established criteria. 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 merely a team check-in or a project status report.

Figure 4 Parameters for measuring PDP reviews



Frequency of reviews also affects the character of the PDP. Some companies have reviews at rigid 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 certain deliverables. 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 staged PDPs, reviews occur after each stage. In spiral PDPs, reviews may occur after each spiral, or series of stages.

3.3 Identifying and distinguishing PDPs

We distinguish the variety of PDPs as combinations of iterations and reviews. For example, staged processes entail narrow iterations and rigid reviews after each stage. Conversely, spiral processes employ more comprehensive iterations and flexible reviews after several stages. These measures of iteration and review allow PDPs to be compared more precisely than before. Earlier investigations of PDPs either identified only one main process or identified a few and distinguished them only with descriptions of their diagram shapes or broad generalisations of their perceived strengths and weaknesses. Here, the characterisations of iterations and reviews become the basis on which all PDPs can be distinguished.

Each iteration/review combination also manages risk differently; no single PDP is suitable for all risk circumstances. 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. 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 specifications accurately reflect rapidly changing customer needs.

4 Research method

This section explains the methodology of the company case studies that underlie our research findings. Case study methodology suits the goals of this research for four reasons. First, it provides empirical data to help build theory about the complex and poorly understood relationship between PDPs and risk. Second, it demonstrates the utility of using quantitative iteration and review metrics to characterise PDPs and distinguish them from each other. Third, the resulting understanding of several real PDPs provides counterexamples to conventional wisdom regarding the applicability of certain processes. Finally, case study research is useful for understanding phenomena and building theory, especially in the immature field of PD management. The case study methodology also supports our proposal of new, quantitative characteristics that describe and distinguish different PDPs while comparing them to earlier qualitative process information (Judd *et al.*, 2001). The limitations of case study research were of relatively minor consequence to this research. Case study methodology has difficulty in proving causality because cases demonstrate only their own existence. However, this research does not attempt to prove causality between development risk management and PDP design. Rather, its main goal is to establish the existence and identities of different PDPs and to build grounded theory relating PDP design to effective management (Dougherty, 2002).

4.1 Case study method

The goal of each case study was to gain a rich understanding of the company's risks and PDPs. 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.

Interviews followed the procedures for semi-structured "interview-conversations" described by Blum (1952), Buchanan *et al*. (1988) and Burgess (1984). Some common PDP questions were asked consistently in all interviews, but in most interviews the latter half was conversational and varied according to the person interviewed. Areas of questioning included both the PDP and the 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, programme 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 criticised by interviewees, who were more at ease making admissions or accusations because they were assured that they would not be personally identified. In addition to private interviews, case studies at two companies 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 were investigated with the help of public data in addition to interviews. In these cases, such as Microsoft and Ford, existing literature and previous sources were considered first, followed by data from interviews.

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 section reach beyond formal company descriptions to include individual engineers' and managers' assessments of how the PDPs are actually implemented.

5 PDP case studies and analysis of results

This section presents nine of our company PDP case studies. The primary company case studies examine Siemens Westinghouse Power Generation (SWPG), Integrated Development Enterprise (IDe), ITT Industries, Aviation Technology Systems (ATS), Ford Motor Company, United Technologies Corporation (UTC), DeskArtes, and Microsoft. A tenth company was used for method validation.

These case study companies represent several different industries and operating environments. Four of the case study companies produce mostly software. Five of the case study companies produce mostly manufactured goods, although several include important software components in their products. Most case study subjects are large corporations, although three of them are smaller companies with hundreds of employees rather than tens of thousands. Some of the companies provided multiple case studies. ITT Industries and Xerox, for example, have units following different processes. In those cases, two different projects were investigated. ITT was also 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. Two of the case studies, Ford and

Microsoft, were selected in part because of the availability of public PDP information. This use of public data provides other researchers or reviewers with the means of independently examining source data. It also allows readers who are familiar with these companies' PDPs – which have been extensively investigated by several other researchers – to compare these research findings to their own knowledge and interpretations. The case study companies can be seen together in Table 1.

5.1 Company and process descriptions

The first case study company, SWPG, is a large engineering and manufacturing company that employs a strict staged PDP to develop turbomachinery for power generation. It faces major technical risk, especially in the areas of quality assurance and thermal efficiency. Market risks are mitigated by early contracts and system of guaranteed liquidated damages, which effectively transfer market risk to technical risk by driving up engineering requirements. Cycle times for this company are slow, with up to several years between the introduction of new products.

The second case study company, IDe, is a small software company that employs an evolutionary delivery PDP to develop its internet-based development management products. It faces major market risk, frequently must customise products to customers' specifications, and operates with a very fast cycle time of only a few months.

The third case study company, ITT Industries, is a large defense contractor whose products include military electronics. ITT faces technical and schedule risks, but market risk is often limited by the monopsonistic nature of the defense industry. The company uses a staged PDP with 'progressive freezes', but applies it differently to different products. Progressive freezes mean that specifications can be set in a piecewise fashion without delaying the entire development programme. Subsequent work can start on those requirements or design aspects that are known to be solidly defined and unlikely to change. ITT's experimentation with PDPs yielded two different results because one process was used for development of a Global Positioning Satellite (GPS) product while another process was used in developing a military Special Unit Operations (SUO) radio.

The fourth case study company, Xerox, is a large manufacturing and software company that develops copiers and document centres. Its considerable market risk forces a corporate culture of on-time delivery and thus translates to schedule risk among design engineering groups. The company uses a hybrid PDP that employs a staged process to develop the electro-mechanical systems and a spiral process for the software systems.

The fifth case study, a collaboration of DeskArtes and Arabia, investigates small software and manufacturing companies that use ray-tracing CAD software developed by DeskArtes to design ceramic tableware manufactured by Arabia. The major risks include market risk inherent in visual product aesthetics and industrial design. Use of the software allows an evolutionary prototyping PDP and extensive customer testing of electronic prototypes.

Table 1 Comparative summary of findings

Company		*SWPG*	*IDe*	*ITT*	*Xerox*	*ATS*	*Markem*	*UTC*	*Ford*	*DeskArtes/ Arabia*	*Microsoft*
PDP description		Strict stage gate	Evol. delivery	Progressive freeze	Hybrid	Spiral	Aspiring stage gate	Stage gate	Stage gate with quasi prototypes	Evol. delivery	Spiral
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 (*i.e.*, heat rate). Mild risk muted by contract structure	Major risk is market – new company is highly customer sensitive	Major risks are tech and schedule, depending on product. Military contracting limits market risk	Market risk translates to schedule concerns	Major risks are tech, primarily quality assurance	Market on one project, tech on the other from late integration	Major risks are tech – FAA regulations and quality requirements	Market risk greatest, but complex tech and budget risk also high	Market risk dominate – customer aesthetics	All risks muted by market dominance; features driven by mkt. risk, tech risk dominates

The sixth and seventh case studies, ATS and Microsoft, both examine companies that use spiral PDPs to develop software. Both faced primarily technical risk because industry dynamics subdued market risk for both companies. The final case studies, UTC and Ford, both analyse large manufacturing companies which employ staged PDPs despite facing considerably different risks. Ford is more concerned with market risk and meeting disparate customer needs, while UTC is more concerned with technical risks stemming from quality assurance.

5.2 Comparative case study findings

Qualitative comparison between individual case studies reaffirms the difficulty companies have in designing PDPs. The quantitative data applies the parameters proposed in Section 3 to actual company processes. The resulting view demonstrates the existence of considerable variety among company processes. The PDP distinctions also suggest that PDPs address risk and integration differently, a principal finding that will be used later in the development of a PDP design method.

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

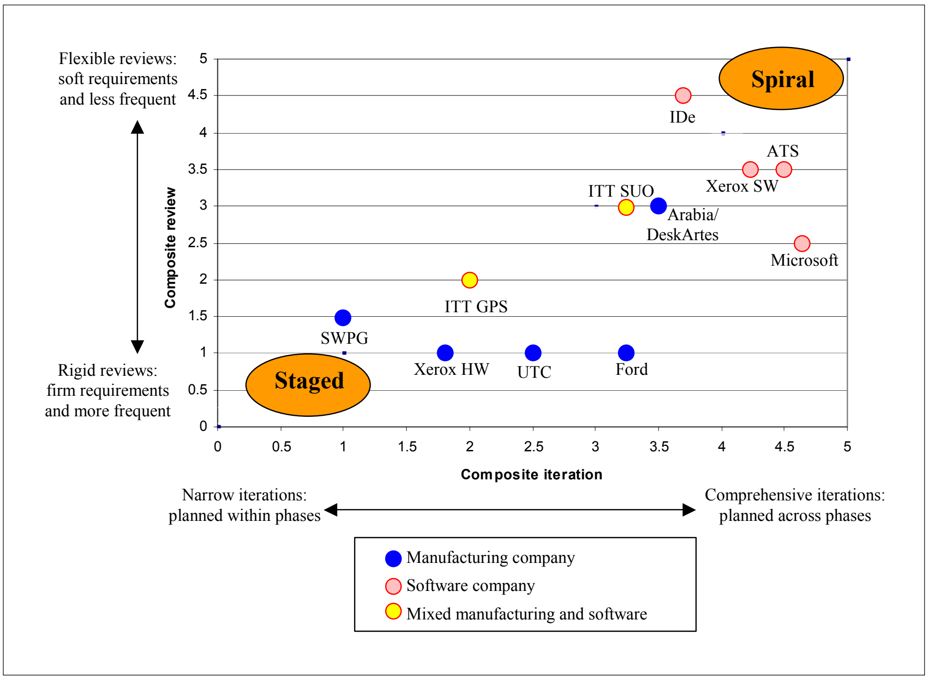
There appears to be no consistent method by which companies design or select their PDPs. Although the case studies did not examine the underlying philosophy of management decisions that led to PDP definitions, several disparate paths were evident. First, some companies changed their PDPs due to organisational shifts. For example, SWPG formalised and added rigidity to its process after a corporate acquisition and merger. 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 by the company's chief engineer in part to overcome persistent PD delays. 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 promoted by a consultant. On the other hand, Microsoft modelled its PDP after the culture of its developers by retaining 'hacker' traits of frequent changes in development code. In summary, some companies carefully consider the PDPs they implement, but others employ PDPs with little regard for the suitability of those processes to company-specific risks or challenges. Companies seem to have based their PDP decisions on many different factors, including their disparate risk scenarios, but none had an analytical process to follow.

We observed that 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 due to informal improvements to the written PDP, such as when ITT allows programme managers to omit process sections that they deem extraneous. Other times, differences between written and actual PDPs are harmful and the result of poor implementation of good ideas. These discrepancies must be noted in order to gain accurate understanding of companies' PDPs.

Quantitatively, the case studies demonstrate the utility of the proposed metrics and display differences among multiple, distinct processes. Section 3 reasoned that all companies use iterations and reviews, and these findings confirm that this is true for each of the case study companies. No two columns of defining characteristics are identical in Table 1, suggesting that the corresponding PDPs are also different. As described in Section 3.3, each column represents a PDP's 'signature' and identifies a different PDP.

Figure 5 plots company process data by composite review flexibility and iteration. High iteration and review values indicate a process favouring flexibility while low iteration and review values indicate a process favouring control. Companies are plotted individually, except for Xerox and ITT, which are represented by two points each due to their internal divisions. In the figure, manufacturing companies tend to be positioned in 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 to be positioned in the upper right of the chart, while companies with mixed manufacturing and software components fall in the middle. However, the considerable scatter in the plot that suggests a key finding: although software developers are more likely than manufacturers to favour flexibility in their PDPs, the software versus manufacturing distinction alone does not predict PDP flexibility.

Figure 5 Overall PDP flexibility by iteration and review (see online version for colours)



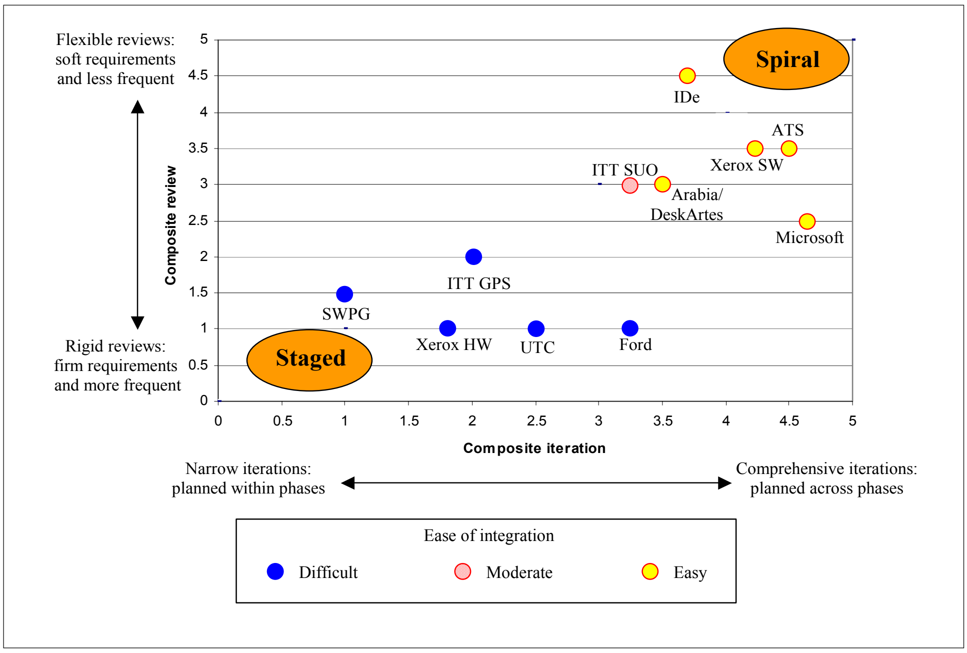
5.3 Case study analysis

Our case studies suggest that a company's ability to integrate and test products can be a critical descriptor and determinant of PDPs. Our analysis moves beyond the useful but imperfect software/manufacturing distinctions of Figure 5 to better understand how companies can operate more widely across the spectrum of PDPs. Product integration often includes an early model, test, simulation, or proto type 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. Some integrations, such as prototypes of complex mechanical products, may require too much time or be too costly to afford multiple tests. 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 service guarantees. SWPG of course also 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/Arabia, Microsoft and 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 integration differences among companies can be seen in Figure 6, which offers additional insight into why companies use certain PDPs. Figure 6 resolves the case studies into clear groups or clusters by categorising the companies by the ease with which they can integrate and test prototypes as part of the PDP. The ease of integration categorisation reflects our understanding of the feasibility, cost, and time required for integrations relative to the value of information gained from such tests. 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. Thus, these integrations become particularly useful lenses by which to view and categorise companies. Figure 6 shows distinct clusters of companies in the two corners of the chart, each grouped with other companies in the same category. This suggests that ease of integration is a powerful determinant of the process type that companies may employ.

These comparative charts of Figures 5 and 6 suggest that many PDPs in commercial practice cluster towards two corners. Exceptions tend to occupy the lower right hand quadrant, suggesting that companies are more flexible with their iterations than their reviews. It may be possible in the future to find PDPs that would occupy the upper left corner of the charts, but such are likely rare because of the difficulty of maintaining rigid iterations while simultaneously loosening reviews.

Figure 6 Case studies charted by ability to perform integrations (see online version for colours)



5.4 *Additional research findings*

The case studies and application of PDP metrics have already demonstrated the existence of multiple PDP variants. Our research also suggests that risk and integration characteristics are useful indicators of which process can be applied most effectively. The ensuing section points out two additional, but related, findings. First, the proposed metrics are a useful means of comparing processes. Second, the relationship between PD cycle times and design flexibility is counterintuitive.

The proposed iteration and review parameters are found to be useful metrics for several reasons. First, they fill a gap in PDP practice in literature. The metrics are necessary because previous literature provided no equitable way of comparing or contrasting PDPs on a common scale. Indeed, prior literature that attempted to compare PDPs did so based on either diagram shape or subjective advantages and disadvantages. These proved to be difficult criteria by which to compare the initial case studies. These iteration and review metrics provided a much-needed common language in which different PDPs could be identified. Every PDP encountered could be described in terms of review and iteration metrics. Once described quantitatively, the PDPs could be uniquely identified, compared, and contrasted. Finally, these metrics have shown to be both understood and welcomed by practitioners, who valued metrics as a way to better understand their own processes. The metrics are easy to communicate and access: managers are frequently able to describe major iterations 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.

The data also suggest that PD cycle time can be a misleading indicator of PDP choice. One might expect that companies with long cycle times would be particularly attuned to market risk because market needs can change over the duration of PD. Thus, companies with long cycle times would emphasise prototyping, customer involvement, and cross-phase iterations. Conversely, one would expect that companies with short cycle times, software companies for example, could afford to avoid such 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.

Such assumptions about PDP choice would be misleading. Although most companies face the common difficulty of writing specifications, companies in fast-paced markets tend to favour 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 costs of prototyping and testing the 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, steep prototyping costs, and the long lead times necessary to build physical models. 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 less sensitive to market changes because of short cycle times nevertheless frequently incorporate market feedback.

6 Implications and conclusions

Product development is a necessary risk for innovative companies. Although it holds the promise of increased sales, market share and profits, PD can fail due to 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. We exhibit and explain PDPs as risk management structures. In exploring the relationships between risk management and PDP design, we make two key contributions.

First, we analyse several PDPs both theoretically and empirically to demonstrate how PDPs substantively differ from each other. We build upon previous literature that either does not adequately distinguish between different processes or makes comparisons based on subjective criteria. We also contribute to the field by proposing and supporting new metrics with which PDPs can be identified and compared. The metrics are based on design reviews and iterations, which are characteristics of all PDPs.

Second, we describe how various PDPs manage different development risks. PDPs with planned iterations and integrations can generate valuable data that are fed back to early process stages and reduce risk. Software companies tend to favour such processes, but the root cause of differences in PDP applicability lies not in whether a product is manufactured with parts or written with code, but rather in a company's ability to integrate or prototype effectively.

References

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

Awny, M.M. (2006) 'Product development strategy: a perspective of enterprises', *Int. J. Product Development*, Vol. 3, No. 2, pp.143–151.

Blanco, E., Grebici, K. and Rieu, D. (2007) 'A unified framework to manage information maturity in design process', *Int. J. Product Development,* Vol. 4, Nos. 3–4, pp.255–279.

Blum, F.H. (1952) 'Getting individuals to give information to the outsider', *Journal of Social Issues*, Vol. 8, No. 3, pp.34–52.

Boehm, B. (1988) 'A spiral model of software development and enhancement', *IEEE Computer*, pp.61–72.

Boehm, B. and Bose, P. (1994) 'A collaborative spiral software process model based on theory W', *3rd International Conference on the Software Process, Applying the Software Process*, IEEE, Reston, Virginia, USA.

Bstieler, L. (2005) 'The moderating effect of environmental uncertainty on new product development and time efficiency', *Journal of Product Innovation Management*, Vol. 22, No. 3, pp.267–284.

Buchanan, D., *et al*. (1988) 'Getting in, getting on, getting out, and getting back', in A. Bryman (Ed.) *Doing Research in Organizations*, New York: Routledge, pp.53–67.

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

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

Cross, M. and Sivaloganathan, S. (2005) 'A methodology for developing company-specific design process models', *Proceedings of the Institution of Mechanical Engineers, Part B. Journal of Engineering Manufacture*, Vol. 219, No. 3, pp.265–282.

Cusumano, M. and Nobeoka, K. (1998) *Thinking Beyond Lean*, New York: The Free Press, pp.157–182.

Cusumano, M. and Selby, R. (1995) *Microsoft Secrets*, New York: The Free Press.

De Meyer, A., Loch, C. and Pich, M. (2002) 'Managing project uncertainty: from variation to chaos', *MIT Sloan Management Review*, Winter, Vol. 43, No. 2, pp.60–67.

De Neufville, R. (1990) *Applied Systems Analysis*, New York: McGraw Hill.

Dougherty, D. (2002) 'Grounded theory research methods', in J. Baum (Ed.) *Blackwell Companion to Organizations*, Medford, MA: Blackwell Publishers.

Easingwood, C., Moxey, S. and Capleton, H. (2006) 'Bringing high technology to market: successful strategies employed in the worldwide software industry', *Journal of Product Innovation Management*, Vol. 23, No. 6, p.498.

Foster, R. and Kaplan, S. (2001) *Creative Destruction*, Chap. 4, New York: Currency.

Gilb, T. (1988) *Principles of Software Engineering Management*, Reading, MA: Addison-Wesley Publishing Company.

Hartmann, G. and Myers, M.B. (2001) 'Technical risk, product specifications, and market risk', in L. Branscomb and P. Auerswald (Eds.) *Taking Technical Risk*, Cambridge: The MIT Press, pp.30–43.

Hekmatpour, S. and Ince, D. (1988) *Software Prototyping, Formal Methods and VDM*, Reading, MA: Addison-Wesley Publishing Company.

Jachimowicz, F., *et al*. (2000) 'Industrial-academic partnerships in research', *Chemical Innovation*, September, pp.17–20.

Judd, C.M., *et al*. (2001) *Research Methods in Social Relations*, 6th ed., Fort Worth: Harcourt Brace Jovanovich.

Krubasik, E.G. (1998) 'Customize your product development', *Harvard Business Review*, November–December, pp.4–9.

MacCormack, A. (2000) 'Towards a contingent model of the new product development process: a comparative empirical study', *Harvard Business School Working Paper Series*, 00-077.

MacCormack, A., Verganti, R. and Iansiti, M. (2001) 'Developing products on internet time: the anatomy of a flexible development process', *Management Science*, January, Vol. 47, No. 1.

MacCrimmon, K.R. and Wehrung, D.A. (1986) *Taking Risks: The Management of Uncertainty*, New York: The Free Press.

McConnell, S. (1996) *Rapid Development: Taming Wild Software Schedules*, Redmond: Microsoft Press, pp.133–162.

Otto, K. and Wood, K. (2001) *Product Design*, New Jersey: Prentice Hall.

Pahl, G. and Beitz, W. (1996) *Engineering Design, A Systematic Approach*, 2nd ed., London: Springer.

Smith, P.G. and Reinertsen, D.G. (1992) 'Shortening the product development cycle', *Research-Technology Management*, May–June, pp.44–49.

Ulrich, K.T. and Eppinger, S.D. (2004) *Product Design and Development*, 3rd ed., New York: McGraw Hill.

Varma Citrin, A., Lee, R.P. and McCullough, J. (2007) 'Information use and new product outcomes: the contingent role of strategy', *Journal of Product Innovation Management*, Vol. 24, No. 3, p.259.

Ward, A., *et al*. (1995) 'The second toyota paradox: how delaying decisions can make better cars faster', *Sloan Management Review*, Spring, Vol. 36, No. 3, pp.43–61.