Planning Design Iterations

Darian W. Unger, Steven D. Eppinger

Abstract – **Companies developing new products have a wide variety of product development (PD) processes from which to choose. Each process offers a different method of iteration to manage risk. Companies must therefore consider the nature and level of risks they face in order to determine the most appropriate iteration and PD process.**

This paper identifies principles of risk and iteration inherent in product development and then explains how several different PD processes manage risk through iteration. It explains current research on PD decision criteria and concludes by proposing a framework to help companies better select PD processes.

Keywords – **Product development, iteration, risk reduction**

I. Introduction

Successful product development (PD) is critical to industrial performance. The speed and quality of PD can provide critical competitive advantages to firms, but the development costs must also be contained. The pressure to improve product development is evident in the words of the Vice President of R&D of Grace Performance Chemicals (W.R. Grace & Co.):

> 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.[1]

Customers, competitors, and regulations can all drive companies to develop new products. Despite the prevalence and importance of PD, this area of technology management leaves considerable room for improvement. Companies can choose from a variety of existing PD processes, but each of those processes manages risk by iteration differently, so choosing the best one can be difficult. Problems can occur if the selected process poorly suits the company. Companies could improve their PD with better criteria for selecting PD processes.

II. Principles of Risk and Iteration

Although PD processes differ across firms, a common problem is that development involves risks. A successful PD process should be able to manage or mitigate the following four major types of risk:

Technical – uncertainty regarding whether a new product is technologically feasible and will perform as expected.

Schedule – uncertainty regarding whether a new product can be developed in the time allowed.

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

Market – uncertainty regarding whether a new product accurately addresses changing customer needs and product positioning with respect to dynamic competition.

These four major risks are neither comprehensive nor entirely independent. Many other factors – such as quality assurance – may also present uncertainty, but they may be subsumed by the larger risks detailed above. Quality risk, for example, may sometimes be a subset of technical risk. The risks also affect each other: technical uncertainty may give rise to a lag in schedule. Also, market uncertainty may lead to the need to build additional prototypes, thus increasing the budget. It is therefore impossible to completely separate the types of risks faced in PD, although the categorizations are useful in planning.

Given these uncertainties, iteration is inevitable and must be managed effectively. Here, iteration is defined broadly to include almost any kind of work that involves correction, feedback or interdependencies. An example of interdependent design tasks can be seen in Fig. 1

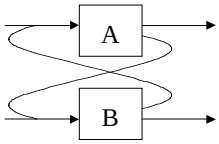

Fig. 1. An interdependent design task configuration

The two tasks in Figure 1 are interdependent because each requires information about the other. Many design processes have hundreds or thousands of such cyclically dependent tasks. These feedback cycles, or iterations, have been successfully modeled by the design structure matrix, a system analysis and project management tool useful in mapping iterations, as shown in previous work by Eppinger.[2]

Interdependent tasks that require feedback are complex and introduce the potential of burdensome and expensive rework if poorly managed. However, iteration is not synonymous with rework. Instead, well-managed design iteration can *prevent* rework and therefore reduce technical, schedule and budget risk. Other types of iteration, such as presenting a customer with a prototype to gage 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 that is 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 are not wastes of effort if they are chosen strategically.

The iterations in this simple example are analogous to well-managed iterations in product development, although PD processes are more complex. There are many types of iterations in product development. They vary in scope, number, level of planning, and type of uncertainty that they are trying to address.

The scope of iteration can be a telling component of a company's PD process. *Narrow* iteration is intra-phase, exemplified by several rounds of interdependent detailed design tasks. *Comprehensive* iteration is cross-phase, exemplified by processes that do not just cycle around a specific part, but rather over a range of process stages from concept to prototyping. Both types of iteration are demonstrated in Figure 2. There is a continuum between these two types of iteration, and processes vary in their iteration scope.

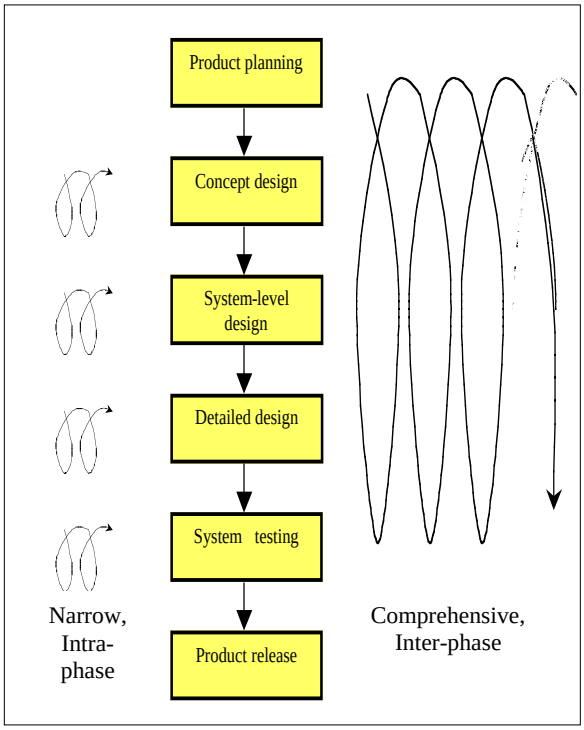

Fig. 2. The two ends of the PD iteration spectrum

The number of iterations also varies by process. Companies manage the iterations in their PD processes by deciding how many cycles to perform before product release. The number of cycles may be preplanned, 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.

Finally, companies differ in their iterations because of the different risks that they face. Iterating over different parts of the PD process can have a wide range of effects. For example, building several prototypes may mitigate technical risk by determining if the product performs to the level of quality promised by design. It may also address risk by providing information on whether the product will satisfy customer needs. However, the iteration may not contribute significantly to mitigating schedule risk because if the prototype is built late in the process. 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 market risk.

The four types of risk are threats to successful PD, so most PD processes attempt to mitigate risk by iterating. As the next section demonstrates, the kinds of iteration used varies by process.

III. Types of Processes to Manage Risk by Iteration

Just as reasons and risks for product development differ, so do PD processes themselves. Companies choose from a variety of processes and methods to iterate through development to mitigate risk and manage PD effectively. This section describes some of the iterative PD process choices. These processes all iterate in some way, but differ widely in their type of iteration and the types of risk they address.

A. The waterfall/stage-gate process

The most widely-used type of product development process, and the basis for comparison in this research, is the traditional stage-gate, or waterfall, process shown in Figure 3.

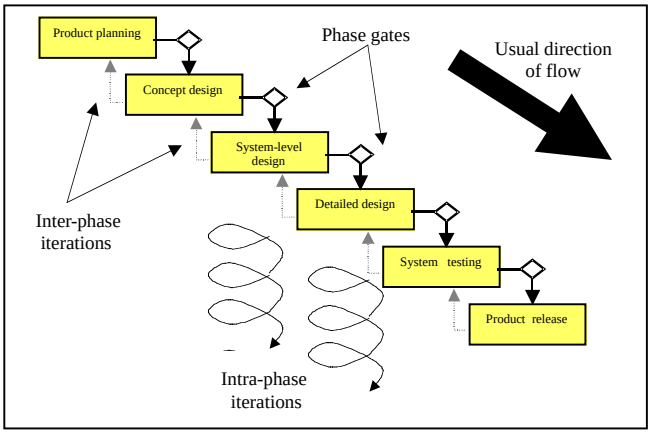

Fig. 3. The traditional stage-gate, or waterfall, product development process[3]

This ideal waterfall process proceeds in discrete stages, or phases, from product planning to product release. The interceding 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 that evaluates whether the action of 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 inter-phase iterations, in Figure 3 indicate that it is possible to reverse course and make changes in earlier phases, but this is generally difficult in a waterfall process. 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 result in substantial rework. These major, and generally unexpected, feedback loops are accepted if necessary, but are difficult and generally confined to successive stages to minimize the expensive rework involved in feedback across many stages.

Stage gates and the difficulties of the reversal process 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.

The advantages of waterfall processes include the structure they impose on development by reaching sharp definitions early in product development. Narrow iterations and phase gates lead traditional waterfall processes to freeze specifications early. Firm 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 waterfall process performs well in cases when the 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, low-cost stages of a project. 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, mistake-inducing changes.

Narrow iterations and phase gates also have the disadvantage of inflexibility. Because they do not cross phase boundaries, narrow iterations cannot incorporate feedback from later process steps. This leads to problems in trying to fully specify requirements in the beginning of a project, especially in a dynamic market. Poor or misleading

specifications can lead to great difficulty later. 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.

Waterfall processes are also sometimes poor matches for companies when speed and time-to-market are more important than added functionality or total quality. Its documentation can be burdensome. In addition, traditional waterfall processes have difficulty incorporating cross-phase processes that don't fit neatly into individual process stages. The waterfall process also has difficulty handling parallel tasks within stages. As a result the length of each stage may be associated with the slowest discipline within that stage, thus lengthening the development process.

The waterfall process has strengths as well as disadvantages. It is clear and solid, but may lack broad feedback and flexibility.

B. Modified waterfalls

The "classic" waterfall is one of several different waterfall processes. Some of its problems can be mitigated by slightly changing the process. This section examines two such modifications.

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 4.

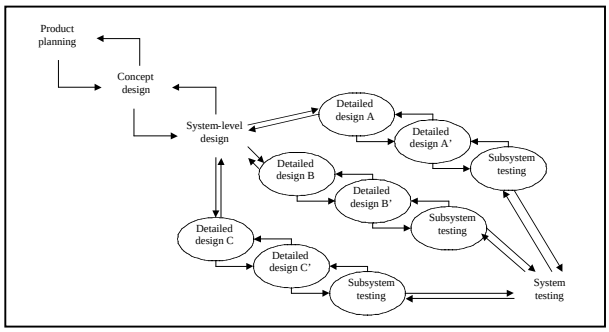

Fig. 4. The waterfall-with-subprojects product development process

Figure 4 shows a waterfall with subprojects. If an 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 forcing the each subproject to finish at simultaneously when some may be completed earlier.

Another modification can be seen in Figure 5, which demonstrates the "overlap" waterfall process where 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 hard divisions between phases. The lack of continuity can lead to some difficulty if different personnel are involved in each step, which remains a problem in some companies. The lack of continuity and can also be problematic if an unforeseen difficulty becomes manifest one stage too late, forcing a potentially unfortunate or expensive reversal of course.

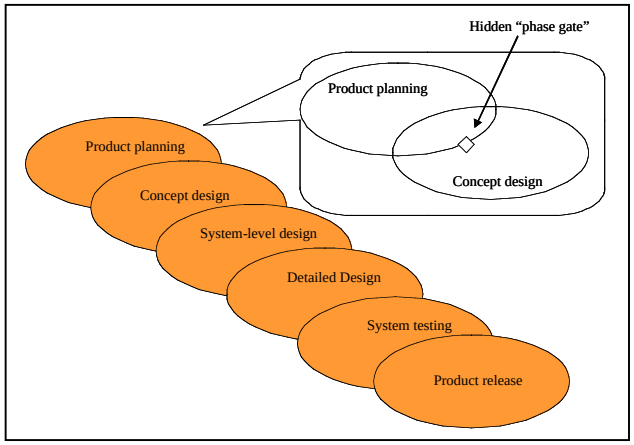

Fig. 5. The overlapping waterfall product development process

The process in Figure 5 can help overcome some of these problems. By allowing stages to overlap, some knowledge – and perhaps feedback – from the next stage can allow for more effective work. Other benefits can include improved teamwork and a more project- (rather than function-) oriented environment.

A problem with both of the modified waterfalls is that they both lead to parallel work. As noted above in Figure 1, 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 more difficult to track.[5]

C. The spiral process

The spiral PD process differs from waterfalls by emphasizing comprehensive iteration. Its proponents assert that the process reduced burdensome and expensive rework in software, and thus lowered development time and cost.[6] The process does appear to have advantages that are demonstrated both theoretically in literature and empirically from interviews. It does, at least at times, lead to successful product development, where success is defined as the development of a competitive product on schedule and within budget.

The spiral process demonstrated theoretically in Figure 6 is a relatively recent product development process that has been adopted by many in the software industry. Unlike the waterfall process, it includes a series of planned iterations that span several phases of development. Despite its circular form, it has five regular steps:

1) Determine objectives, alternatives, and constraints
2) Identify and resolve risks
3) Evaluate alternatives
4) Develop the deliverables for that iteration, and verify that they are correct
5) Plan the next iteration (if there is one.)

The radial dimension in Figure 6 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 inwards, each loop brings it closer to completion, while each movement towards the center reflects additional cost.

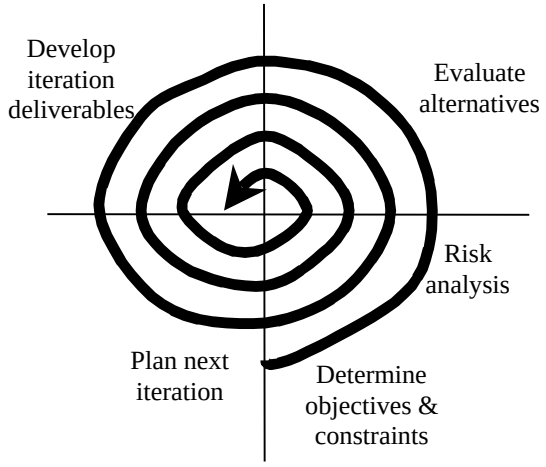

Fig. 6. An adaptation of the spiral product development process – Spiraling towards a completed product

The spiral process is a risk-managing process that allows managers to evaluate risk early in the project when costs are still relatively low. "Risk" in this context entails all four major risks described earlier, including poorly understood requirements and architecture, performance problems, market changes, and potential problems in developing specific technologies. All these uncertainties entail risks that can threaten a project, but the spiral process helps to screen them early, before major costs are incurred. Using the spiral process can be desirable in rapid product development because as costs increase later in the project, the risks decrease.[7]

The initial emphasis on risk analysis is important to the flexibility of the process; in some cases of minimal uncertainty and no iterations, it may collapse to become a waterfall process and have only one radial spiral. In other cases, significant risks can be evaluated early and the development process can be amended (or ended) in successive spirals to better suit the company.

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 slower waterfall process. This glimpse yields information that can be incorporated in early concepts, requirement specs, 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, the author of the spiral

process himself acknowledges difficulties in the first spiral step of determining objectives, alternatives, and constraints. Later scholarly work extends the spiral process by suggesting a split of this first step into several others.[8] In addition to these technical disadvantages, there may well be additional barriers to the adoption of the spiral process, including corporate momentum and potential difficulty in switching processes gradually.

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 expected small iterations which occur within individual stages of waterfall processes, and unlike large but unplanned and unwanted feedback loops which can occur in less successful waterfall processes, iterations in the spiral process are *planned* and *span at least three stages* of the development process. Despite this distinction, critics may consider it similar to a waterfall process if the milestones and deliverables between each spiral round act merely as old-fashioned phase gates.

D. Evolutionary prototyping and delivery

The evolutionary prototyping PD process differs from the waterfall and spiral processes by concentrating on the visible prototypes of a product. As with the other processes, iteration is acknowledged to occur, but the iterations focus on prototyping and refining prototypes until release. Figure 7 demonstrates the process.

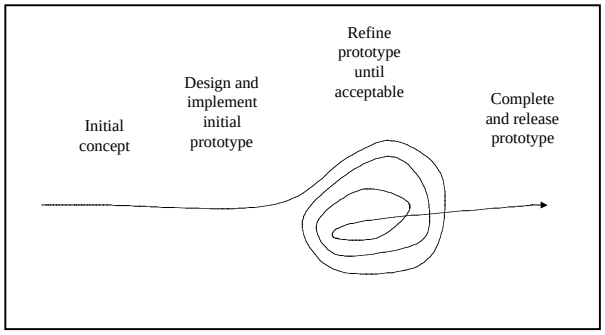

Fig. 7. The evolutionary prototyping product development process

The evolutionary prototyping process can handle changing 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 acceptable outcome is reached. Because it is not possible to know how long each project will be and because building an unspecified number of prototypes can be expensive, the schedule and budget risk can be high.

Evolutionary delivery, as pictured in Figure 8, is similar to evolutionary prototyping, but has added emphasis on the core design.

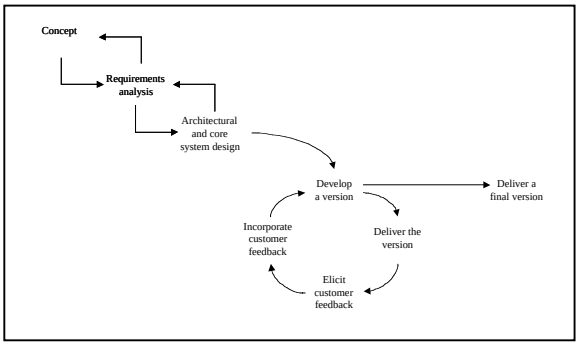

Fig. 8. The evolutionary delivery product development process[9]

This method also includes a series of iterations in the latter half of the process, but does not mix design and prototyping as thoroughly. The 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 may 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 late customer suggestions.

E. Design to schedule/budget

The final PD process demonstrated here involves yet another type of iteration. 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 9.

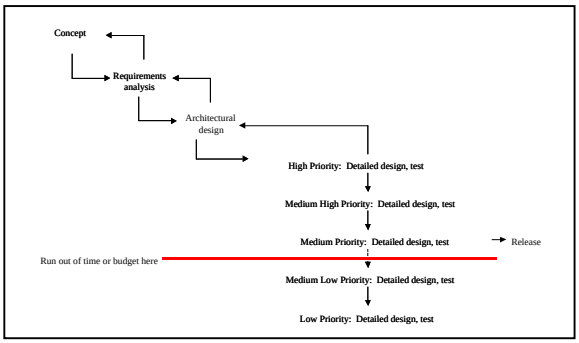

Fig. 9. Design to schedule or budget

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 method does away with the rigidity of waterfall-style gates between phases and adds 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 – where each new, would-be release is an improvement over an earlier prototype – may occur until a budget or schedule limit is reached. If daily or periodic builds have been practiced, 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, those risks are minimized 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 occur 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 the features expected.

As demonstrated in all of the examples above, PD processes vary widely and call for different types of iterations as they attempt to manage risk. The iterations range from narrow to comprehensive and sometime focus on mitigating one of the four major types of risk. These variations create an array of product development processes that companies can choose to follow. The following section describes research that may help companies make these selections more effectively.

IV. Research and Methodology

Current research focuses on improving PD process selection by examining the differences between processes and the differences between companies developing new products. This research stems from curiosity about an interesting phenomenon: companies vary widely in their PD processes but some processes are used more widely in certain industries. Also, there is no methodical guide for companies to follow in selecting the PD processes that best suit them. Given the importance of PD to companies, improvement should be possible.

This research attempts to examine several key PD questions, including

1) How do different PD processes address iteration and risk? What factors set PD processes apart?
2) Is there a relationship between process, iteration, risk, and company or product?
3) Can a series of criteria be developed by which companies can better select their PD processes?

The previous section addressed the first question; this paper proposes that there are a variety of different PD processes, each of which iterates differently as part of risk management. Thus, variations of iteration are key distinguishing features, including the number, scope, and level of planning of iterations.

Case study research addresses the remaining two questions. This empirical study is currently demonstrating the range of PD processes used by a variety of companies – each with different products, different organizational structures, and facing different risks. The case studies include software companies such as Integrated Development Enterprise (IDE) and Aviation Technology Systems (ATS) as well as manufacturing companies such as Ford Motor Co. and Pratt & Whitney (P&W). Case studies also include "mixed" companies whose product include both software and manufactured components, such as Xerox and ITT Industries. Ongoing research with these companies examines their PD procedures and surveys their PD mangers and employees to gage their development efforts. Figure 10 demonstrates how these companies span a range of processes and iteration types.

Fig. 10. Companies use different processes to span a range of iteration types and risk mitigation.

Although still in progress, the empirical research already suggests several key findings. First, one size does not fit all; companies with different products, organizations, iteration preferences and risk profiles cannot all use the same PD process with the same degree of effectiveness. Some processes will suit companies better than others, suggesting that there is a relationship between process and company or product.

Second, selecting the best PD process can be difficult for managers. Although a relationship between process and company exists, the tenets of that relation are still unclear. Managers are not quite haphazard in choosing PD processes, but they may select processes suboptimally because there are few guides that suggest which processes would best fit their own companies. Given the wide range of PD processes available, it is unfortunate yet understandable that there are some mismatches between companies and processes.

Third, it may be possible to help managers improve their difficult PD process selection by clarifying the relationships between processes and companies. Understanding the relationship requires an understanding of its links. The two most important links are method of iteration and risk profile, which help categorize PD processes and companies so that they can be matched more effectively. Process iterations may include varying numbers of planned and unplanned iterations, which can also vary in scope from narrow to comprehensive. Company risks to be addressed by these processes can include technical, market, schedule and budget risk. Together, iteration and risk can be key indictors in companies' attempts to improve PD.

The case studies also suggest several subsidiary criteria that may help companies better select their PD processes. Iteration for the management of risk remains instrumental in matching processes with companies, but five other criteria can further refine the alignment. The first is the type of product or process decomposition possible. Decomposition can help determine what companies can iterate across, as can be seen in Figure 11. If decomposition type limits the ability of a company to iterate in certain ways as part of PD, then decomposition can be used as a selection criterion in choosing a PD process. Related work by Jootar suggests how partitioning can make substantial differences in how much work is necessary in successful PD.[10]

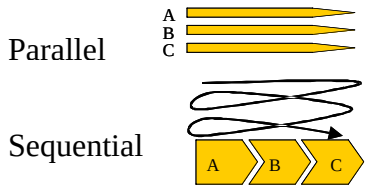

Fig. 11. A parallel decomposition may not allow for iteration.

The second additional criterion is the level of interactivity of changes. Some product changes are highly interactive because they affect other components, while others are relatively independent because they are add-on or plug-in features that require no changes to other parts.

The third criterion is the platform nature of the product being developed. Whether a product is a new generation of a platform or merely an improvement on an older version may have profound impacts on the speed and type of PD iterations performed, and therefore on PD process selected.

The fourth criterion is specification rigidity. If specifications are required to be firm by company policy, customer requirements, or government regulations, iteration may be narrowed in scope. These effects on iteration again may affect process choice.

Finally, the fifth additional criterion is lead time. Iterations may be reduced in scope or number if, for example, prototypes take a long time to manufacture. Alternatively, if ramp-up production does not require the creation of new tooling or long-lead-time items, new or different iterations offer the possibility of improving PD, thus affecting PD process choice.

The ongoing research demonstrates that PD processes use different forms of iterations to address different types of risk. Not all PD processes are suitable for every company, so iteration and risk are two major criteria that companies can choose suitable PD processes. In addition to these two major criteria, type of decomposition, interactivity of changes, platform nature of the product, specification rigidity, and lead time are also important measures that can

help companies improve their PD process selection. These criteria for matching companies and PD processes are demonstrated in Figure 12.

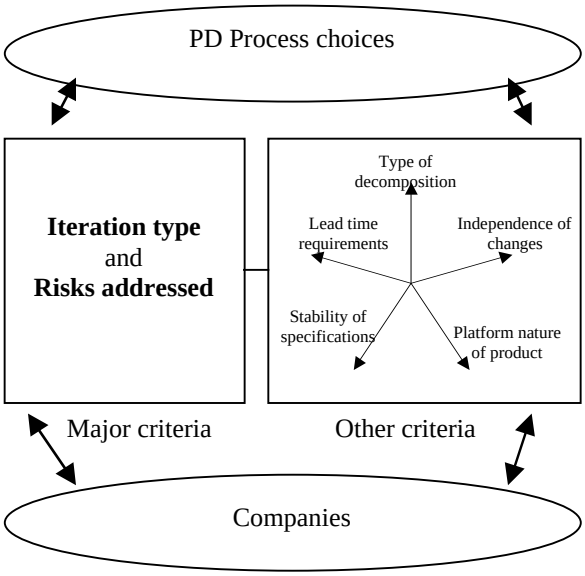

Fig. 12. Decision criteria linking companies and suitable PD processes.

Figure 12 is the first proposed map to help managers select specific PD processes that suit their companies. It traces major and minor criteria to corresponding product development processes. Furthermore, the axes in the spider graph of "other criteria" define the axes so that greater quantities of each lend themselves to processes with narrower iteration of the kind seen in strict waterfall processes, while closing in on the origin suggests comprehensive iteration of the type seen in spiral processes.

Although rudimentary, the map and criteria in Figure 12 can be a useful tool in helping companies select PD processes that are appropriate to their own circumstances. Future research is necessary to refine this map of decision criteria, weigh the differing criteria appropriately, and find boundaries and quantifications along the spider graph so that companies' best options can be clearly marked.

V. Conclusions

Companies can improve their product development if they have better methods for selecting their PD processes. A large array of PD processes manage development risk by iterating, but differ from each other in the number and scope of their iterations, as well as in the types of risk they mitigate.

Current case study research demonstrates that PD processes differ tremendously between companies and that companies could benefit from better PD process selection. Furthermore, risk and iteration methods are important criteria for matching companies with PD processes that suit them and their products. Additional criteria, including the type of decomposition, interactivity of changes, platform nature of the product, specification rigidity, and lead time are also important measures that can help companies improve their PD process selection.

The proposed map of these PD decision criteria can help align companies with advantageous product development processes

Acknowledgements

The authors would like to acknowledge the help and suggestions of Daniel Whitney, Chris Magee, Maurice Holmes, and Jay Jootar.

References

[1] Felek Jachimowicz, et. al., "Industrial-academic Partnerships in Research," Chemical Innovation, Sept. 2000, pp. 17-20.
[2] Steven D. Eppinger, "Innovation at the Speed of Information," *Harvard Business Review*, Jan. 2001, Vol. 79, No. 1, pp. 149-158.
[3] Adapted from: Steve McConnell, *Rapid Development: Taming Wild Software Schedules*, Microsoft Press, Ch. 7: Lifecycle Planning, pp. 136-137.
[4] Preston G. Smith and Donald G. Reinertsen, "Shortening the Product Development Cycle," *Research-Technology Management*, May-June, 1992, pp. 44-49
[5] Op. Cit. McConnell, pp. 144-145.
[6] Barry M. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, 1988, pp. 61-72
[7] Op. Cit. Boehm, p. 64.
[8] Barry Boehm and Prasanta Bose, "A Collaborative Spiral Software Process Model Based on Theory W," IEEE, 1994.
[9] Op. Cit. McConnell, pp. 147-151.

[10] Jay Jootar, "A System Architecture-based Model for Planning Iterative Development Processes" SMA Paper, Center for Innovation in Product Development, January 2002.