Proceedings of DETC '04
ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
Salt Lake City, Utah USA, September 28-October 2, 2004

DRAFT

DETC2004-57351

INFORMATION INCORPORATION POLICIES IN PRODUCT DEVELOPMENT

Jian Zhu[+], Ali A. Yassine, Ramavarapu S. Sreenivas
Department of General Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801

ABSTRACT

In the present paper, we develop a dynamic programming (DP) model of the product development (PD) process. We conceptualize PD as a sequence of decisions: whether to incorporate a piece of information that just arrived (i.e. became available) or wait longer. We, then, utilize this formulation to analyze several different situations depending on the type and nature of information being exchanged: external versus internal information, and stationary versus dynamic information. We derive optimal decision rules to determine whether (and when) to incorporate or not for each case. An analysis of the model results in several important findings. First, we must not seek to incorporate all available information that is related to our design activity. Once the information collection exceeds certain value, the design team should stop collecting further information. Second, only when past design work accumulates a certain threshold value should the team include the latest information and perform rework. Large uncertainty of the information and large sensitivity of the design activity make the information less likely to be incorporated. Lastly, the managerial implications are discussed with several numerical examples.

(*Keywords: Product Development, Overlapping, Design Iteration, Information, Concurrent Engineering*)

1. INTRODUCTION

Product development (PD) has been regarded as a key source of competitive advantage since the early 1990s and it is no longer a sole engineering responsibility (Clark and Fujimoto 1991, Wheelwright and Clark 1992). In today's fast-changing and highly-competitive markets, companies are not only focusing design efforts on premium quality, but also putting great efforts forward to other objectives such as time-to-market, development cost, and production cost as well. As a consequence, PD has many different types of decisions made by both engineers and mangers.

We define the decisions related to PD into two broad categories: product design decisions and development project decisions. Product design decisions are made to transfer customer needs into salable products or services. Decisions range from physical product specifications, manufacturing processes, market channels to after-sales services. Development project decisions enable the development project to proceed in a timely and cost efficient way. Such decisions control the progress of PD projects by specifying project team organization, timing and sequence of design activities, allocation of resources, and communication mechanism among team members. We will only focus on the second type of decisions in this paper.

Concurrent engineering (CE), sometimes called overlapping, has been recognized as a useful technique to reduce the lead time by performing PD activities concurrently with the use of preliminary design information (Clark and Fujimoto 1989, Rosenthal 1992, Smith and Reinertsen 1995, and Stalk and Hout 1990). Many research efforts have been put forward to modeling CE process in PD (Ha and Porteus 1995, Krishnan et al. 1997a, Loch and Terwiesch 1998, and Roemer et al. 1999). However, most of this research treats CE from an activity-based perspective. Instead of using this approach, we model the PD process by an information-processing plus decision-making perspective.

[+] Corresponding author: Product Development Research Laboratory,
University of Illinois at Urbana-Champaign,
Tel. (217) 333-7621, email: jianzhu@uiuc.edu

We view PD as a complex process comprising activities that make decisions under time and budget constraints (see Fig. 1). Each activity gathers the information it needs as inputs at the beginning, then make decisions upon analysis of the received information. Finally, the decisions are released as outputs by this activity and become available input information for other activities. In an information processing view, information is created when the decisions are made. Then it is communicated to those activities that require this information and it is used when the designers of those activities analyze the problems.

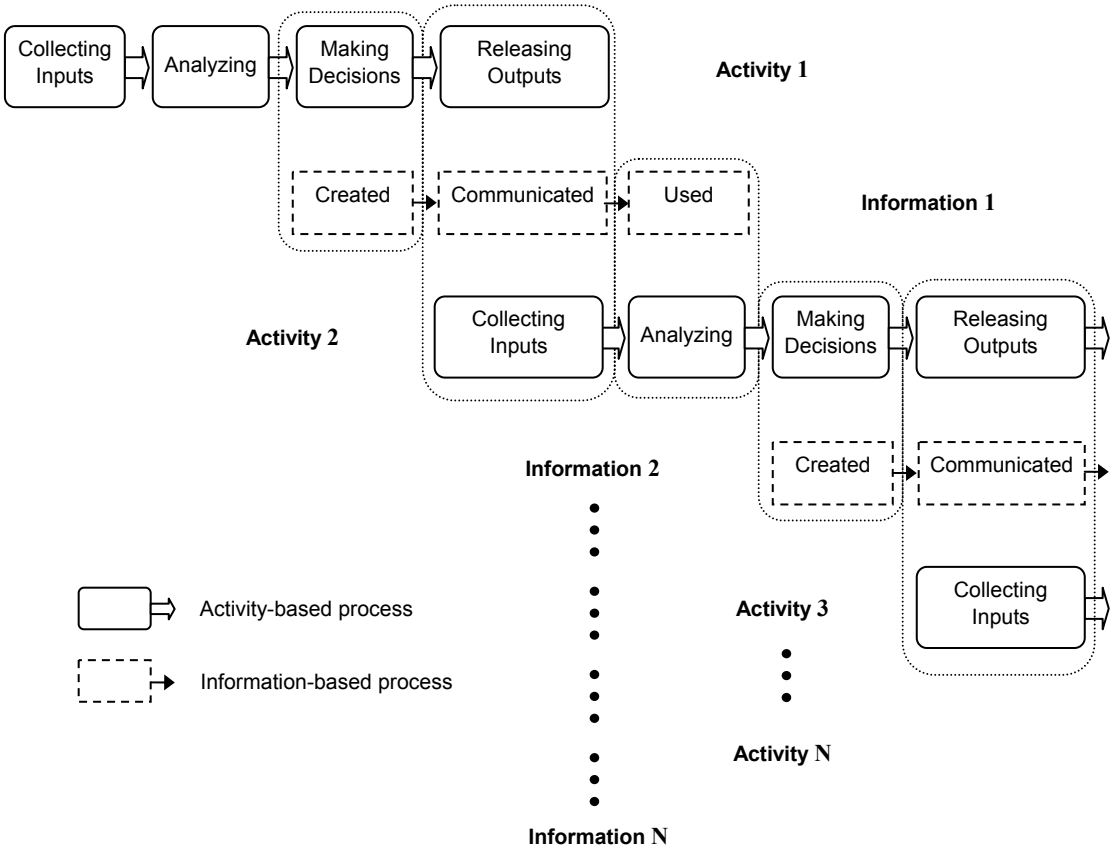
In this paper, we provide strategies for information incorporation under uncertainty. A development team is faced with a lot of information which is somewhat related to its design task. Due to the use of CE, some information is unfinalized and tends to change in later stages. The team may consider the preliminary information which accelerates development process but offers the potential of rework. We develop a dynamic programming model for this PD environment and analyze several different situations depending on the type and nature of the information being exchanged. An analysis of the model provides several important managerial insights to determine whether (and when) to incorporate new design information.

In the next section, we present a brief survey of the previous studies in PD area that are relevant to our work.

2. LITERATURE REVIEW

The literature on CE is voluminous. Takeuchi and Nonaka (1986) characterized concurrency as a "rugby team" approach to design compared to the sequential "relay race" approach. Later Clark and Fujimoto (1991) observed that in automobile industry companies with shorter development lead time frequently overlapped their development activities. Due to these early works, CE has become a core technique for saving development time (Smith and Reinertsen 1995). In spite of its popularity, the risks associated with CE should not be overlooked. Because it utilizes incomplete information, it is widely recognized that overlapping is more costly due to more frequent communications and increased iterations. Hoedemaker et al. (1999) shows that concurrency is not always a better proposition. Then the questions are in what kind of situations CE is favorable and how managers should plan and control CE processes.

Figure 1 Product Development Process from an Information Processing Perspective



Several existing theories and models are developed to address these questions. Ha and Porteus (1995) analyze a product design for manufacturability project that has a product design phase and a process design phase, which are executed in a parallel fashion. Their purpose is to find a progress review strategy that minimizes the total expected project completion time. A progress review can prevent design flaws in the previous stage from moving to later stages, which would result in extensive rework. However, reviewing too frequently can cause too much time, possibly delaying the project completion time. They formulate the problem as a dynamic program and characterize the optimal progress review strategy that minimizes the total expected project completion time. Our model differs from the work of Ha and Porteus by conceptualizing the information much wider, while the information in their model is design flaws. Furthermore, we model the value of the information and the risk associated with incorporating the information.

Krishnan et al. (1997a) investigate the overlapping of two sequentially dependent PD activities. They model the upstream information as an interval value and introduce two important notions, upstream evolution and downstream sensitivity. Upstream evolution can be interpreted as how close the preliminary upstream information is to its final value, and downstream sensitivity is referred to the relationship between the duration of downstream iteration and the magnitude of the change in the upstream information. Based on that, they formulate a mathematical program and determine the optimal number of iterations and the starting time for each iteration. Then they present a conceptual framework to indicate in which cases overlapping is more favorable. However, they do not consider the uncertainty within the PD process. Our model, in contrast, fits better in practice, where decisions are made facing uncertainty.

Similarly, Loch and Terwiesch (1998) present an analytical model for overlapped upstream and downstream tasks to minimize time-to-market. They model the preliminary information as engineering changes happen to the upstream task. These changes are released to downstream task in batch and impose rework on the downstream task. They derive an optimal concurrency and communication policy which is affected by uncertainty and dependence.

Our work differs from the work of above papers in three major aspects. First, all the above three papers analyze the development project without including the development performance. It is the common assumption in these models that the product performance is not influenced by how the development project is organized. Krishnan et al. (1997b) discuss how the sequence of decision making leads to depreciation of product performance. Based on this point, in our model, the decisions on how the development process proceeds are made with consideration of the impact on both time and performance.

Second, we focus on only one activity, but those papers investigate the circumstance of two dependent design activities, one of which (upstream activity) feeds information to the other (downstream activity). Essentially, the downstream task not only incorporates the information from the upstream task, it also incorporates a lot of other information that may not come from the development project itself. For example, the fluctuation of interest rate, the change of supplier parts' prices, new customers' preferences, new technology and so on. Upstream design information is just one type of information that is needed to enable downstream tasks. Accordingly, we only focus on one particular activity (i.e., a downstream task in other models), ignoring the details of the upstream activity, but only considering the information it releases, and analyze the impact of various types of information on this activity.

Third, with no exception, all the above models are aimed at time-to-market reduction. However, in practice, most PD projects have pre-specified deadline. Consequently, we consider a development project in a more realistic way and seek to maximize the development performance subject to the deadline constraint (Joglekar et al. 2002).

The remainder of the paper is organized as follows. In section 3, we discuss the role and value of information in PD. Section 4 introduces the general mathematical model, and section 5 analyzes the model and characterizes the optimal strategy. Finally, we summarize the managerial insights provided by the model and discuss the opportunities for further research.

3. INFORMATION IN PRODUCT DEVELOPMENT

As stated earlier, the present paper views PD as a complex process involving hundreds of decisions in strategy, marketing, finance, engineering, manufacturing and customer service. Hence, information plays a very important role in facilitating and influencing decision making in PD. Large amount of information is involved. Krishnan and Ulrich (2001) provide a long list of typical decisions within the context of PD in seven categories. For example, information about customer needs, available technology and production cost is used in product concept phase to establish product specifications. Information about these specifications is then important input to further detailed design and prototyping.

Since it is dealing with information, PD is very different from other processes, such as manufacturing a car or building a house. Although these processes can be complicated, scheduling these processes is much easier. If two activities are irrelevant, then they can be carried out in parallel without any interactions and iterations; otherwise, they have to be executed in sequence. Consider manufacturing a car, Body-In-White assembly cannot be started until all the parts (bodies, doors, bonnets, floor-pans) are cut out of sheet metal. Paintwork cannot be done until the body-shell is assembled and cleaned. However, other activities can be executed simultaneously. Seats, wheels and engine can be produced at the same time when the body is assembled.

PD differs because of the use of preliminary information. Preliminary information is used to start downstream activities earlier, which may accelerate the PD project. But this approach is not fit for manufacturing processes. Imagine the door as preliminary output of body assembly process. We cannot paint the door before the whole body is assembled. Therefore, an information-based perspective rather than an activity-based perspective can provide us better insights in PD decision-making. Activity-based perspective tells us what and when activities should be started and completed (i.e. sequencing and timing of activities). But Information-based perspective addresses the problems such as "What information do we need to complete an activity?" and "When should we use this information?"

To answer these questions, it is worthwhile to study how information is created, communicated and used during the development process and what impacts information has on the design activities. By understanding the value and risk of information, better decisions can be made to develop products more efficiently. In this section, we represent mathematically the properties of information in PD. In the next section, we use these notations to develop models of information flow in PD.

3.1 Types of Development Information

PD projects often involve many professionals from different departments in the company, sometimes at different geographic locations. Due to the inherent complexity of development projects, there are hundreds of different kinds of information exchanged in PD. As a consequence of this diversity, it is impossible for us to build a model for each specific type of information. Hence, it is necessary to characterize information into few generic types. For the purpose of our analysis, we classify the information in PD in the following two manners.

With respect to the source, information in PD can be classified into two types, internal and external. Internal information refers to information that is generated by design activities, completed or undergoing, within the development project (e.g. budget and time constraints, component specification, and test data, etc.), and external information is the information that is provided by sources outside of the project (e.g. customer needs, market demand, and technology options etc.).

By another perspective, information can also be categorized into stationary and dynamic. Stationary means that the information will not change in the short term, at least within the duration of the development project, while dynamic means that the information tends to alter. Internal information from completed activities is stationary and information from undergoing activities is dynamic. External information can also be either stationary or dynamic. For example, information about a prospective technology (Krishnan and Bhattacharya 2002) is dynamic, while information about a proven technology is stationary.

3.2 Value of Information

It is widely acknowledged that if a design activity proceeds without incorporating information from its predecessor activities, design flaws will arise and development performance will deteriorate (Ha and Porteus 1995). This fact suggests that information benefits the development performance. Hence, if we measure the value of a design activity in term of the performance, then each piece of information has a value to that activity. The value of information is defined as the normalized difference between the performance obtained by incorporating this information and the nominal performance.

PROPOSITION 1: *For any information A and design activity B, the value of information, υ_{AB}, is such that:*

$$\upsilon_{AB} = \left(Q_B|A - Q_B\right)/Q_B \quad (1)$$

where $Q_B|A$ and Q_B are the expected performances that can be obtained by incorporating information A and ignoring it, respectively.

Note that it is possible for the value of information to be negative. Conventional wisdom seems to dictate that taking more information is better than less information when a decision is made. But it is not necessarily true in PD. Incorporating information is not free; it has an "opportunity cost". For an activity that has been ongoing for a while before the information arrives, it requires extra time to make modifications on the work already done due to the new information. On the other hand, the team can also improve the performance in isolation without this information. We call the incremental improvement by working in isolation for a certain amount of time as the opportunity cost.

If the information is not very relevant, then the performance is improved just a little after incorporation. The amount of time spent in obtaining this (marginal) improvement could have been better used by staying on the original course to product improvement. It stands to reason that if the incremental improvement caused by the information is less than the opportunity cost, the information should not be incorporated.

3.3 Risk of Information

PD projects are subject to uncertainty. As a result, unfavorable outcomes, that are not expected at the time decisions are made, frequently occur in development projects. There are two components of risk associated with decision-making in PD.

First, there is the risk of interdependence. Most decisions in PD are intertwined. Decisions of one design activity usually set targets and constraints to other activities and vice versa. Thus, incorporating new information (i.e., making changes to a task may improve its performance, but may also lead to degradation of performance in other dependent tasks. As a result, the overall system's performance may be worse off.

The other component is the risk of incorporating incomplete information. Due to the increasing pressure of competition, designers try to commence design activities earlier by using

incomplete information. In light of the interdependence of tasks mentioned earlier, it is very rare that assumptions regarding incomplete information made at the start of an activity are ever changed as the activity progresses. Modifications to these assumptions are necessarily accompanied by rework.

3.4 Cost of Incorporating Information

Our model postulates that each time incorporation takes place, a certain amount of cost is required. The cost consists of information acquisition cost and incorporation cost. Acquisition cost is the cost that one needs to pay to obtain relevant information. We assume that there is no acquisition cost in our model. This assumption is practical in advanced information technology environments. Designers can transfer design information through many media in an almost costless and seamless fashion, for example, telephone, fax, email, web conference etc.

Incorporation cost represents the cost that is required to modify current design based on the information that is incorporated. In general, the cost of rework can be incurred in two ways, financial cost and time delays. Time delay is inevitable, once it is decided that the information is to be incorporated and the rework is started. However, if overtime were allowed, the time delay would be transformed to an expense. Hence, in our model, for simplicity we view the incorporation cost being purely financial.

The cost of rework is determined by three factors. First, as more time is spent on an activity using old information, the amount of cumulative work that must be modified will be larger. Hence, rework time is an increasing function of the time between two successive incorporations. Second, rework time is also dependent on how much the information and the activity are related. In general, a change in a major input needs longer rework time than a change in a minor input. The third factor that influences rework time is the degree of sensitivity of the activity. Degree of sensitivity implies the robustness of the activity to changes. In other words, a larger degree of sensitivity indicates a longer rework time. For example, let us consider analytical and physical prototyping. A change in one dimension is easy to be made in an analytical prototype. By contrast, much more efforts would be required for the same change occurring to a physical prototype. Then we say that physical prototyping has a greater degree of sensitivity than analytical prototyping. Moreover, sensitivity of an activity is also affected by the design method. Toyota's set-based approach (Ward et al. 1995), in which the team works with a set of prospective designs together without making a commitment until necessary, is less sensitive to changes than point-based approach.

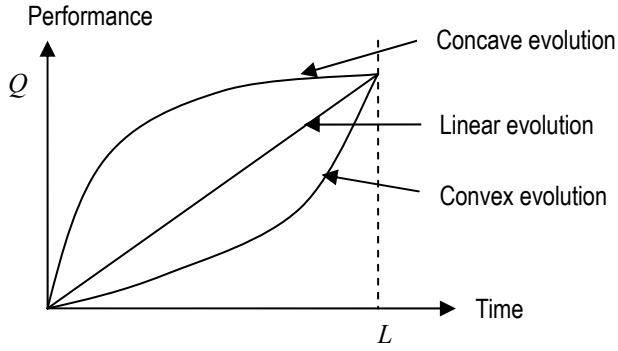
4. THE GENERAL MODEL

Our work is focused on a PD project that has a fixed deadline. The deadline for the whole project can be the launch date of a new product. The new product should be developed by the time within this deadline. For a given design activity, the deadline can be an intermediate milestone.

Consider a design activity that has a deadline L. The team must complete this activity by the deadline. Otherwise, subsequent activities will be delayed and the total project duration will be increased. Subject to the deadline constraint, the team is trying to maximize the development performance. In addition, we assume that the team will not stop before the deadline even if required performance target is achieved. Designers will keep working on their tasks, improving the performance until the due date.

First, let us understand how the performance of a development activity progresses if no new information arrives during the course of this activity. The model is called zero-information model. The reader may imagine the zero-information model as an independent activity, which does not need information from other activities.

Figure 2 Performance Evolution

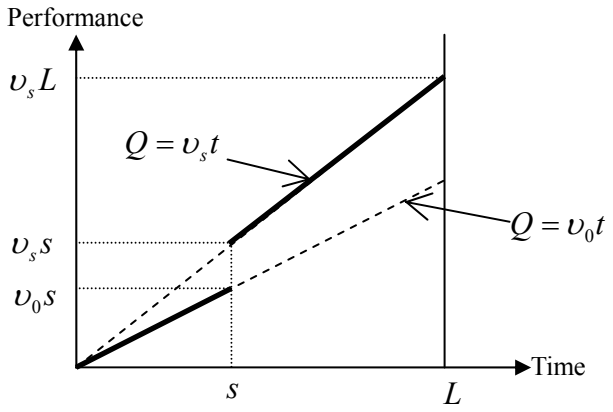


Let Q be the performance of the design activity, which is a function of time t. In general, there are three possible shapes that the performance evolution might follow as shown in Fig. 2. If the performance function has a concave shape, the team works with a diminishing rate. Specifically, the evolution is very fast at the beginning and slows down as the activity progresses. The concave evolution performance represents those activities that are carried out early and further improvement becomes more and more difficult as performance grows, the team gets tired or the resources become limited. The activities with linear evolution are similar to a manufacturing process. The performance increases at a constant rate. The convex evolution is also frequent seen in PD projects. Some activities need a lot of setup, hard thinking and analysis, so the evolution is slow at first. Once the preparation work is done, the performance then progresses rapidly.

Under the zero-information model, no information-related decisions are involved in the development process. The performance is determined by the parameters which represent the properties of the designers and activities, such as designers' skills, due date, the degree of difficulty of the activity and so on.

Now, let us investigate the case where the activity sees new information during the development process. For simplicity, we assume that the performance evolution is linear. At the beginning, the design team collects the available information that values υ_0. Based on the information, the team works on the design problems of this activity at rate υ_0. If the information does not change and no new information arrives, it is same as the zero-information model, and the deadline performance would be $\upsilon_0 L$.

Figure 3 Performance Evolution with Information Incorporation



However, the information collected at the beginning may change during the course of development. Or some new information may arrive at certain time s, which can add value to the activity. After evaluating its value υ_s, the team may decide to incorporate the updated or new information into the current design, which incurs certain amount of cost to adjust its former work. The new information shifts the performance up to $\upsilon_s s$ and makes the performance increases at the rate υ_s later on. If no incorporation occurs later, the deadline performance would be $\upsilon_s L$. The team may also decide to ignore this information for the current product and use it in the follow-up products, in order to save the rework cost. For instance, a new microprocessor technology is introduced when a computer manufacturer is developing a new model of desktop. The new microprocessor can improve the performance of the new desktop. However, it takes extra cost to make modifications on all work that have been done. Then the design team may either stick on its current design and use the new microprocessor in later products or redesign the current product to take advantage of the new technology.

Suppose that the development performance is measured as the expected profit that the company would obtain by the new product. Then, the designer wants to maximize the expected return (the expected deadline performance less the total rework cost, $\sum r_{ij}$, where r_{ij} is the cost of rework incurred if the team decides to perform rework at time i and the last rework performed at time j). Note that other costs rather than rework cost are common and independent of the incorporation decision.

The decision process is modeled as a dynamic program. We divide the time from the beginning of the activity to the deadline into N periods. At the end of each period, the designer checks whether the information changes in the last period, and then makes a decision whether to incorporate it or not. We say that we are at stage k if there are k periods remaining. Let υ_k be the largest value among all information that has arrived since the latest incorporation to stage k. The team is at stage k and in state (υ_k, υ_i), if i is the last stage at which the last incorporation was done and the value of the last incorporated information is υ_i. There are two options: continue working on the activity based on the information incorporated at stage i or incorporate the latest information immediately.

Suppose the team decides to incorporate the latest information. It costs the team r_{ki} to perform the rework. Then the team goes to the state (υ_k, υ_k). If the team decides to ignore the current information and continue working, then the process proceeds to the next stage and the state becomes $(\upsilon_{k-1}, \upsilon_i)$.

We are interested in finding an optimal information incorporation strategy that maximizes the expected return. Let $J_k(\upsilon_k, \upsilon_i)$ be the maximum expected performance at the deadline minus the total rework cost spent from stage k onwards, when at stage k and in state (υ_k, υ_i). Thus, the optimality equation can be written as:

$$J_k(\upsilon_k, \upsilon_i) = \max\{J_{k-1}(\upsilon_{k-1}, \upsilon_i), J_k(\upsilon_k, \upsilon_k) - r_{ki}\} \quad (2)$$

5. MODEL ANALYSIS

In this section, we will tailor the general model developed in section 4 to tailored models for different types of information. We will characterize the form of the optimal incorporation strategy and derive the explicit formula for the optimal solution wherever possible to gain managerial insights.

5.1 Stationary Information Model

In this subsection, we will study the case of stationary information. As defined earlier, stationary information will not change after it arrives. Specifically, the value of the information is constant. Therefore, the team needs to make the decision only once. Compared to dynamic information, which will be discussed later, stationary information is easier to analyze.

Consider a piece of information that is fed to a design activity. Suppose the information is stationary and arrives at stage k. Its value to the activity is υ, which means the deadline performance of the activity will be increased by υL, if the information is incorporated. If the team decides to

incorporate at time s, $k \le s \le 0$. It will cost the team $r(s)$ for rework.

THEOREM 1. *The following results hold for each stage k.*

(a) The optimal incorporation policy for the team receiving stationary information is given as following: If $\upsilon L \ge r(k)$, the team should incorporate the information immediately; otherwise, ignore the information.

(b) The performance at the deadline is increasing in k.

The proof of Theorem 1 is straightforward. Since $r(s)$ is increasing in s and the information will not change again, incorporating the information earlier is always better than incorporating later. Therefore, the team should either incorporate the information immediately after it arrives ($s = k$) or ignore it forever. The benefit of incorporating the information is $\upsilon L - r(k)$. If the benefit of incorporating is greater than zero, the team should incorporate the information immediately; otherwise, just ignore the information.

Furthermore, it is interesting to note that the performance achieved at the deadline is decreasing in the arrival time of the new information. So it is not only the information itself that provides value, but the timing of information generation as it can change the performance of a development project. The analysis indicates that the earlier the new information becomes available, the more benefit it offers to the activity. In order to maximize the expected performance at the deadline, the information that is important input to downstream activities should be frozen early in the development process.

5.2 External Dynamic Information Model

Dynamic information occurs frequently in practice. To accelerate the PD process, upstream activities often release preliminary information to enable downstream activities to start earlier. This information keeps changing until it is finalized. Consequently, downstream iteration takes place in order to update past work. Oftentimes, external information is dynamic. For example, a prospective technology, that may improve development performance but is not fully certain, may be introduced before it is finally proven (Krishnan and Bhattacharya 2002).

For the activity that faces dynamic information input, incorporation policy is especially important. Incorporating information can be risky if information may change substantially. And such kind of changes may cause too many iterations that results in poor design performance at the deadline. On the other hand, if the team does not incorporate until the information is finalized, large amount of rework will be needed.

We will first study external dynamic information and analyze internal dynamic information in the next section. External information is from sources outside the development project team. At each stage, the team checks the information that has arrived. If no information appears in the current period, we say that a piece of information of zero value has arrived. Let the value of the information arriving at each period be υ_k and given by a random variable X with probability mass function $P(X = j) = p_j$. At each stage, the team decides whether to use the information with the largest value or wait for more information. Suppose, at stage k, if the team chooses to incorporate the information that has the largest value, the deadline performance of the activity would be υ and the team pays $r(N-k)$ to perform rework. We assume that $r(\cdot)$ is a convex function due to the effect of learning, forgetting and relearning (Ash and Smith-Daniels 1999). If the team decides to wait and collects more information, the process goes to the next stage $k-1$. Therefore, the optimality equation (2) can be rewritten as:

$$J_k(\upsilon) = \max\{\textstyle\sum_{j=0}^{\infty}(P_{\upsilon j} \cdot J_{k-1}(j)), \upsilon - r(N-k)\} \quad (3)$$

where $P_{\upsilon j}$ is the one-stage transition probability from state υ to state j.

THEOREM 2. *The following results hold for each stage k.*

(a) The team would only incorporate the newest information that arrives in the most recent period.

(b) It is optimal to incorporate the newest information, if its value υ_k is at least as large as a cutoff value ψ.

(c) The cutoff value is the smallest number ψ that satisfies

$$E[(X-\psi)^+] \le r(N-k+1) - r(N-k) \quad (4)$$

where $(X-\psi)^+ = \max\{X-\psi, 0\}$.

For easier readability of the paper, the proof of Theorem 2, as well as other main results, appears in the appendix. Theorem 2 provides explicit conditions that are sufficient to make the optimal decision. Under this cutoff value policy, the process terminates at the first stage where condition (4) is met. It is interesting that condition (4) actually compares incorporating the information ($\upsilon - r(N-k)$) with waiting for exactly one more stage and then incorporating ($\upsilon + E[(X-\upsilon)^+] - r(N-k+1)$). The left hand side of condition (4) can be interpreted as the expected extra benefit of waiting one more stage. Since the state cannot decrease (the largest value ever received cannot decrease in time), the expected extra benefit of waiting decrease because $(X-\upsilon)^+$ decreases in υ. The right hand side of condition (4) represents the additional cost required for waiting one more stage, which is increasing in k. It can be intuitively seen that the additional cost is small at the beginning and can be compensated by the extra benefit. The overall gain is positive, and the expected return increases. As the process proceeds, the extra benefit decreases, and, however, the additional cost increases. At the first stage when the extra benefit is smaller than the additional

cost, the team should not wait any more and incorporate the information at once.

Theorem 2 implies that, in order to achieve a better outcome of the development activity, it is important to have a good estimate of the external information. If the information is underestimated, then the expected extra benefit gained by waiting one more stage is smaller than its actual value. As a result, the team would incorporate the information earlier than the optimal time. If the information is overestimated, the design team will see a false large extra benefit that induces the decision maker to delay incorporation more and leads him to miss the optimal incorporation time. Both situations make the design outcome worse off.

5.3 Internal Dynamic Information Model

Internal information differs from external information by two ways: (1) internal information has a certain point of time to be finalized, and (2) the design team must include the finalized internal information. Internal information is produced by upstream activities. Therefore, the deadline of the upstream activity is the date when the information is finalized. Once reaching the deadline, the design team of the upstream activity will stop working on its current task and moves to other tasks. As a result, the output information will not change after that. Also, the final upstream information is usually a set of constraints to the downstream activity. So if the designer does not include the finalized information, his design would be of no value.

Since the finalized upstream information must be incorporated and the performance is only determined by the value of the latest incorporated information, the value of preliminary information does not play any role in this model. Thus, the maximizing problem in Section 4 is transformed to a problem that minimizes the total rework cost.

In this case, the incorporation strategy will be useful only in the overlapping period, which is from the beginning of the activity to the finalization date of the information, because no change will happen after that. As in the earlier sections, we divide the overlapping period into N stages. We set the state to be (i,u), where i is the number of stages that have elapsed since the last incorporation, and u indicates whether the current information is same as the last incorporated information. If the information changes, we let $u=1$; otherwise, $u=0$.

Let $J_k(i,u)$ denote the minimum expected total rework cost will be incurred from current stage k to the terminal stage N if in state (i,u). Then the optimality equation can be written as:

$$J_k(i,1) = \min\{J_{k-1}(i+1,1), r(i) + J_k(0,0)\} \quad (5)$$

$$J_k(i,0) = p_{k-1} \cdot J_{k-1}(i+1,1) + (1-p_{k-1}) \cdot J_{k-1}(i+1,0) \quad (6)$$

$$J_N(i,u) = u \cdot r(i) \quad (7)$$

where p_{k-1} is the probability that the information will change in stage $(k-1)$, and $r(i)$ is assumed to be a convex function of i.

THEOREM 3. *There exists a set of $N-1$ integers, $s_1, s_2, \cdots, s_k, \cdots, s_{N-1}$, such that if the process is at stage k and the last incorporation happened i stages ago, then it is optimal to incorporate the information if and only if $i \geq s_k$.*

Theorem 3 provides an easy-to-execute policy to the design team. At each stage, the decision maker needs only to check how many stages have passed since the last incorporation. If the number of stages exceeds the cutoff value, the team should incorporate the incorporate the information; otherwise, it can wait until the next stage and repeat the process again. Although the explicit formula for s_k is hard to obtain, s_k can be computed as long as the specific values of parameters are given. This issue will be revisited in the future research section of this paper.

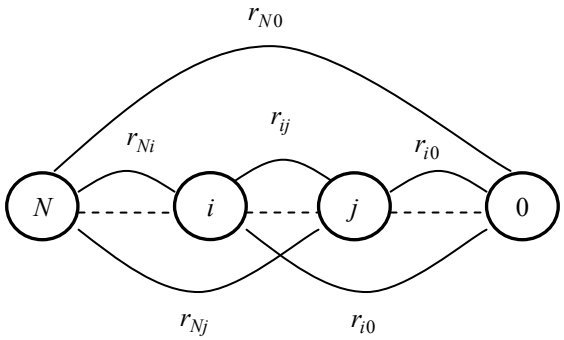
To see how the parameters affect the optimal solution, we use a numerical example where we assume that the cost of rework $r(i) = G + \alpha i^{\gamma}$. G is the fixed cost for each incorporation and αi^{γ} is the variable cost that depends on the number of stages since the last incorporation. For simplicity, we assume the probability that information changes in stage k to be constant p. We calculate the optimal cutoff values, $s_1, s_2, \cdots, s_5$ for the last five stages to derive managerial insights by changing one parameters and fixing the others.

As shown in Table 1, s_k increases as p increases. This is intuitive that, if p is small, the designer is more willing to incorporate information when i is small in order to reduce the variable incorporation cost, as the chance that the information change again is small. If p is large, that means that the information is likely to change later, then the designer would like to accumulate i to share the fixed cost by more stages. When $p=0$, the information stays constant, and the stationary information model applies. The team should incorporate the information immediately, that is $s_k = 0$.

Table 1 Optimal Cutoff Values as a Function of p, Given $G=10$, $\alpha=0.4$, and $\gamma=2$

p	s_1	s_2	s_3	s_4	s_5
0.0	1	1	1	1	1
0.2	3	3	3	3	4
0.4	5	4	4	4	5
0.6	8	6	4	4	5
0.8	10	6	5	4	5

Figure 4 The Shortest Path Problem for $p=1$



If $p=1$, the information changes at each stage, the problem becomes a classical shortest path problem as shown in Fig. 4. The distance between node i and node j is the rework cost incurred if the information is incorporated at stage i, given the last incorporation happens at stage j. The objective is to find the shortest path from node N to node 0, which is equivalent to minimize the total rework cost.

Table 2 suggests that the optimal cutoff value is also increasing in G. When $G=0$, the rework cost $r(i)=\alpha i^{\gamma}$. If $\gamma \geq 1$, the cost function is convex. To minimize the total rework cost, the designer will incorporate every change immediately after it arrives. When $G>0$, it is not economic to incorporate the information at every stage. The designer would wait until $i \geq s_k$, and as a matter of fact, as G increases, the cutoff value s_k becomes greater.

Table 2 Optimal Cutoff Values as a Function of α, Given $G=10$, $p=0.4$, and $\gamma=2$

G	s_1	s_2	s_3	s_4	s_5
0	1	1	1	1	1
5	3	3	3	3	3
10	5	4	4	5	5
20	10	8	7	6	6

Table 3 Optimal Cutoff Values as a Function of p, Given $G=10$, $\alpha=0.4$, and $\gamma=2$

α	s_1	s_2	s_3	s_4	s_5
0.0	∞	∞	∞	∞	∞
0.2	10	8	7	6	6
0.4	5	4	4	5	5
0.6	4	3	3	4	4
0.8	3	3	3	3	3

Table 4 Optimal Cutoff Values as a Function of α, Given $G=10$, $p=0.4$, and $\gamma=2$

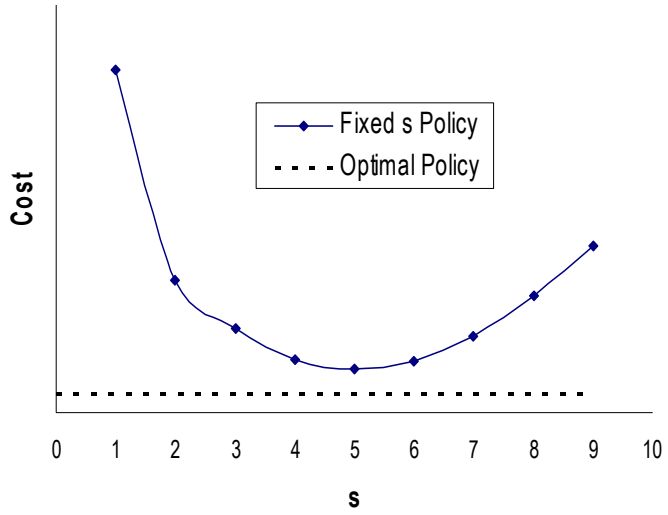
γ	s_1	s_2	s_3	s_4	s_5
0	∞	∞	∞	∞	∞
2	5	4	4	5	5
3	2	2	2	2	3

Table 3 demonstrates the relationship between the optimal cutoff value and the sensitivity of the activity. If the task is more sensitive (has a larger α), it would cost the designer more to do rework. Note that if $\alpha=0$, s_k becomes infinite, that means the team will always incorporate the information at each stage as long as a change arrives. While α increases, the variable cost becomes more significant such that the optimal cutoff value decreases. Similarly, the coefficient γ has the same impact on the cutoff values (See Table 4). When $\gamma \leq 1$, $s_k=\infty$. Once γ exceeds 1, the cutoff values decrease in γ.

5.4 Fixed *s* Policy

In the previous subsection, we presented the structural form of the optimal incorporation policy for the internal dynamic information. That is, it is optimal for the team to ignore new information at stage k as long as the stages elapsed since the last incorporation, i, is less than s_k. The computation of the value of s_k is essentially done by unfolding the recursion in Eq. (5), (6) and (7) backwards from the deadline. This can be computationally intensive if the number of stages N is large. An alternative approach is to use a constant s for all stages. We call this fixed s policy. To provide practical applications, we now focus our attention on the fixed s policy.

Figure 5 Total Rework Cost VS. Incorporation Policy, Given $N=10, G=7, \alpha=0.4, \gamma=2$, and $p=10$.



Using fixed s policy, the team does not care how many stages are remaining. Once s stages have elapsed since the latest incorporation, the team checks whether any change occurred during this period. If there is a change in information over this period, the team would incorporate it by initiating a rework; otherwise, the team would wait for the next change and incorporate it immediately. We run a Monte Carlo simulation for the total rework cost incurred by using the optimal incorporation policy and fixed s policies for a 10-stage problem (see Fig. 5). The dotted line represents the cost generated by using the optimal incorporation policy developed in Section 5.3. We find that the total rework cost, as a function of s, displays a U-shape curve, and for this particular example, $s = 5$ is the best among all s. That implies that there would be such an s that minimized the expected total rework cost.

THEOREM 4. *If a product development activity has a large number of stages and a fixed s policy is used, then the following results hold:*

(a) If $p < 1/T$, *then any change should be incorporated immediately after it occurs.*

(b) If $p \geq 1/T$, *then the optimal s, that minimizes the expected total rework cost, satisfies:*

$$s^* + (1-p)^{s^*} / p = T \tag{8}$$

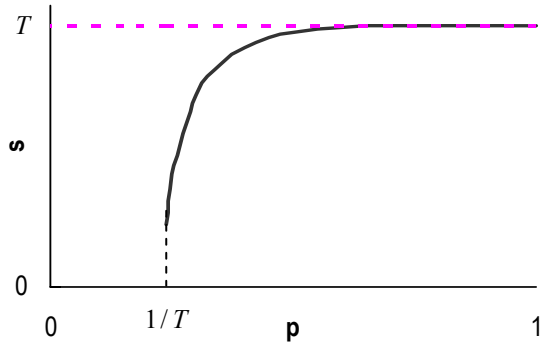
where $T = \sqrt[\gamma]{G/\alpha(\gamma-1)}$.

Part (a) says that if the probability, that a change occurs in one stage, is small enough that is, the information revision from the upstream activity is a rare event, then the team does not need to use the fixed s policy. For such information, once a change occurs, the team should incorporate it immediately. When the information is quite uncertain, part (b) suggests an optimal s that minimizes the expected total rework cost.

Note that the second term on the left hand side of Eq. (8) is greater than or equal to zero. Thus, s^* is less than or equal to T. Also, s is integer and must be greater than or equal to 1. Thus, we know that s^* could be equal to $1, 2, \cdots$ or $[T]$, $[T]$ is the smallest integer greater than T. We calculate the left hand side of Eq. (8) and compare it to T. Starting with $s = 1$ and increasing s by 1 each time, we pick s^* to be the first s that makes the left hand side of Eq. (8) be greater than T. Note that T is constant for any given pair of design activity and feeding information.

Figure 6 demonstrates the relationship between the optimal s and the probability that a change will occur in one stage. It is not surprising that the relationship follows the same pattern as that between the dynamic s_k and the probability. When $p = 1$, $s^* = T$. s^* decreases as p decreases. This is quite intuitive: If the probability that a change happens is small, the benefit of waiting for more stages becomes small. Thus, the team would be willing to incorporate the change more frequently.

Figure 6 Optimal s as a Function of p.



6. DISCUSSION

In this section, we discuss several important managerial implications based on our proposed analytical model. The first finding is that the team must not necessarily always incorporate all the relevant available information that may seem useful for the design activity. Clearly, the more information incorporated, the better the new product. Hence, design teams tend to incorporate every piece of information. However, it may not be optimal. It is not only the information itself that offers value, but also the timing of information arrival. Incorporating a piece of information that arrives approaching the deadline could cost huge amount of resources and time to modify the design. Yassine et al. (2003) study the "design churn" phenomenon and argue that information delay is the major cause of churn. Our result suggests that the team should give up pursuing further information when its current information collection exceeds a critical value. By using this policy, the team sacrifices the prospective design improvement by including further information to avoid unnecessary design churn. As explained in Section 5.2, the critical value depends on estimation of future information and the stage that the decision process is in. If the information arrives in the early stages, it is of little use, then the design team would expect a large increment in the future information and is more likely to wait longer to collect more information. In contrast, if the information received in the early stages is of great value, then the team would expect just small improvement that the future information could offer. Thus, the team is unwilling to pay extra cost that cannot be compensated by the benefit.

Second, the analysis of the numerical example gives several interesting insights. The information incorporation policy is mainly determined by two factors: the uncertainty of the information and the cost of incorporating the information. The cost of incorporation can be further divided into fixed cost and variable cost.

When the uncertainty of the information is very high (i.e., depending on the type of product being developed and/or

technology being used), it is essential to balance the tradeoff between the fixed and variable costs. For the case of high fixed cost and low variable cost, the cutoff value for each stage will be large and, in turn, incorporation happens less frequently. The opposite is the case of small fixed cost and large variable cost. For this type of activity, the cutoff value will be small and the team incorporates the information more frequently.

If the uncertainty of the information is low, the team has more confidence in the current information. Then it is more willing to incorporate information early and save the incorporating cost. Thus, the cutoff values would be smaller than the case of receiving more uncertain information. In our model, we set the uncertainty to be constant. However, information produced by different activities has different properties. For the information generated by a design activity with convex evolution, the uncertainty is small at the early stages and become large in the later stages. Therefore, the team using this information is more willing to incorporate it in the early stages and more reluctant to incorporate in the later stages. On the contrary, if the information is produced by a design activity with concave evolution, changes are more likely to happen in the early stages. Then the cutoff value would be large at the beginning and small at the end.

Lastly, we investigate an alternative strategy, fixed s policy. It is interesting that using fixed s policy does not provide any improvement when the information is quite certain. For this case, the team should simply incorporate any change immediately after it occurs. When the uncertainty of the information is high, we suggest an optimal fixed s policy that minimizes the expected total rework cost, and the optimal s is determined by the structure of the rework cost and the uncertainty of the information.

7. CONCLUSION

In this paper, we formulate the decision process of whether or not to incorporate new information in product development as a dynamic programming model. We first develop a general model and then tailor it to three stylized models: stationary information, external dynamic information, and internal dynamic information. For the internal dynamic information model, we analyze the relationship between the optimal policy and model parameters with a numerical example.

We are focusing on obtaining managerial insights and, thus, have kept the model as simple as possible. Extensions of the model are possible in several approaches. First, the model can be modified to be a continuous-time problem. Second, sometimes the time delay by rework cannot be transformed to a financial cost. Thus, tradeoffs between time, cost, and performance would be more complicated and worth further research. Lastly, in our model, we have only two options reacting to new information: incorporate or not. Other options may be also available. For example, one group of the team continues working with the old information and the other working with the new information. Thus, the model becomes a dynamic resource allocation problem.

In addition, we suggest using the fixed s policy for projects that have a large number of stages and provide the optimal s. The cost gap between the optimal incorporation policy and the fixed s policy deserves both further theoretical and empirical research.

APPENDIX

Proof of Theorem 2

For part (b) and (c): Let

$$S = \{\psi : E[(X-\psi)^+] \le r(N-k+1) - r(N-k)\}$$

Thus S is a closed set of states, because $(X-\psi)^+$ decreases (ψ cannot decrease) and $r(N-k+1)-r(N-k)$ increases ($r(\cdot)$ is convex). We shall show that $J_k(\upsilon) = \upsilon - r(N-k)$ for all $\upsilon \in S$ and all k. It is immediate for $k=0$, so suppose it for $k=n-1$. Then for $\upsilon \in S$,

$$\begin{aligned} J_n(\upsilon) &= \max\{\textstyle\sum_{j=0}^{\infty}(P_{\upsilon j}\cdot J_{n-1}(j)), \upsilon - r(N-n)\} \\ &= \max\{\textstyle\sum_{j\in S}(P_{\upsilon j}\cdot J_{n-1}(j)), \upsilon - r(N-n)\} \\ &= \max\{\textstyle\sum_{j\in S}(P_{\upsilon j}\cdot j) - r(N-n+1), \upsilon - r(N-n)\} \\ &= \max\{\upsilon + E[(X-\upsilon)^+ - r(N-n+1), \upsilon - r(N-n)\} \\ &= \upsilon - r(N-n). \end{aligned}$$

Hence, $J_k(\upsilon) = \upsilon - r(N-k)$ for all $\upsilon \in S$ and all k.

If the policy stated in part (b) and (c) is used, assume at stage k, the information with the largest value υ is received at previous stage j, $j>k$. Since this information is not incorporated, we know that $E[(X-\upsilon)^+] > r(N-j+1) - r(N-j) > r(N-k+1) - r(N-k)$. Thus, part (a) follows obviously.

Proof of Theorem 3

LEMMA 1. $J_k(i,1) - r(i)$ is increasing in i.

PROOF. It is obvious that $J_0(i,1) - r(i)$ is increasing in i, so we can assume the same for $J_{n-1}(i,1) - r(i)$. Now, from Eq. (5),

$$\begin{aligned} J_n(i,1) - r(i) = \min\{&J_{n-1}(i+1,1) - r(i+1) + r(i+1) - r(i), \\ &J_n(0,0)\} \end{aligned}$$

Because of the induction hypothesis and convexity of $r(\cdot)$, $J_n(i,1)-r(i)$ is increasing in i. Thus, the result follows.

When at stage k and in state i, it is optimal to incorporate the information if $J_k(i,1) \ge r(i) + J_k(0,0)$. Let

$$s_k = \min\{s : J_k(s,1) - r(s) \ge J_k(0,0)\}.$$

By Lemma 1, we can see that for all $i \ge s_k$, $J_k(i,1) - r(i) \ge J_k(s_k,1) - r(s_k) \ge J_k(0,0)$. Hence, it is obvious that it is optimal to incorporate the information when at stage k if $i \ge s_k$.

Proof of Theorem 4

Lemma 2. $f(s)$, the average cost for any given s, is convex in s.

PROOF. Let h be the expected interval between two incorporations. Under the fixed s policy, $h(s) = s + (1-p)^s / p$. It is obvious that $h(s)$ is increasing in s . On the other hand, the long-run average cost f is equal to $(G + \alpha h^\gamma) / h$. It is immediate that f is convex in h . Thus, f is convex in s .

Given that f is convex in s , we can find the minimum of f by setting the first order derivative with respect to s equal to zero. By doing so, we get Eq. (8). Note that when $p < 1/\sqrt[\gamma]{G/\alpha(\gamma-1)}$, Eq. (8) has no solution. That implies that f is monotonically increasing. As a result, $s = 0$ minimizes f . Part (a) follows.

REFERENCES

Ash, R. and D. E. Smith-Daniels, "The Effects of Learning, Forgetting, and Relearning on Decision Rule Performance in Multiproject Scheduling," *Decision Sciences*, 30 (1999), 47-82.

Clark, K. B. and T. Fujimoto, "Lead Time in Automobile Product Development Explaining the Japanese Advangtage," *Engineering and Technology Management*, 6 (1989), 25-58.

Clark, K. B. and T. Fujimoto, *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*, Harvard University Press, Cambridge, MA, 1991.

Eisenhardt, K. M. and B. N. Tabrizi, "Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry," *Administrative Science Quarterly*, 40 (1995), 84-110

Ford, D. N. and J. D. Sterman, "Iteration Management for Reduced Cycle Time in Concurrent Development Projects," *Working Paper*, 2002

Ha, A. Y. and E. L. Porteus, "Optimal Timing of Reviews in Concurrent Design for Manufacturability," *Management Science*, 41 (1995), 1431-1447.

Hoedemaker, G. M., J. D. Blackburn, and L. N. Van Wassenhove, "Limits to Concurrency," *Decision Sciences*, 30 (1999), 1-18.

Joglekar, N. R., A. A. Yassine, S. D. Eppinger, and D. E. Whitney, "Performance of Coupled Product Development Activities with a Deadline," *Management Science*, 47 (2001), 1605-1620.

Krishnan, V., S. D. Eppinger, and D. E. Whitney, "A Model-Based Framework to Overlap Product Development Activities," *Management Science*, 43 (1997a), 437-451.

Krishnan, V., S. D. Eppinger, and D. E. Whitney, "Simplifying Iterations in Cross-Functional Design Decision Making," *Journal of Mechanical Design*, 119 (1997b), 485-493.

Krishnan, V. and K. T. Ulrich, "Product Development Decisions: a Review of the Literature," *Management Science*, 47 (2001), 1-21.

Krishnan, V. and S. Bhattacharya, "Technology Selection and Commitment in New Product Development: The Role of Uncertainty and Design Flexibility," *Management Science*, 48 (2002), 313-327.

Loch, C. H. and C. Terwiesch, "Communication and Uncertainty in Concurrent Engineering," *Management Science*, 44 (1998), 1032-1048.

Roemer, T. A., R. Ahmadi, and R. H. Wang, "Time-Cost Trade-offs in Overlapped Product Development," *Operations Research*, 48 (2000), 858-867.

Rosenthal, S. R., *Effective Product Design and Development*, Irwin, 1992.

Sheremata, W. A., "Finding and Solving Problems in Software New Product Development," *Journal of Product Innovation Management*, 19 (2002), 144-158.

Smith, P. G. and D. G. Reinertsen, *Developing Products in Half the Time*, Van Nostrand Reinhold, 1998.

Stalk, G. Jr. and T. M. Hout, *Competing Against Time: How Time Based Competition Is Reshaping Global Markets*, Free Press, New York, 1990

Takeuchi, H. and I. Nonaka, "The New Product Development Game," *Harvard Business Review*, Jan.-Feb. (1986), 137-146.

Thomke, S. and D. E. Bell, "Sequential Testing inProduct Development," *Management Science*, 47 (2001), 308-323.

Ulrich, K. T. and S. D. Eppinger, *Product Design and Development*, McGraw-Hill, New York, 1997.

Ward, A., J. K. Liker, J. J. Cristiano, and D. K. Sobek, "The Second Toyota Paradox – How Delaying Decisions Can Make Better Cars Faster," Sloan Management Review, 36 (Spring 1995), 43-61.

Wheelwright, S. C. and K. B. Clark, *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency and Quality*, Free Press, New York, 1992.

Yassine, A. A., N. R. Joglekar, D. Braha, S. D. Eppinger, and D. E. Whitney, "Information Hiding in Product Development: The Design Churn Effect," *Research in Engineering Design*, 14(3), 2003.