

# Performance of Coupled Product Development Activities with a Deadline

Nitindra R. Joglekar • Ali A. Yassine • Steven D. Eppinger • Daniel E. Whitney  
*Boston University, Boston, Massachusetts, 02215*

*Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139*

*Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139*

*Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139*

---

This paper explores the performance of coupled development activities by proposing a performance generation model (PGM). The goal of the PGM is to develop insights about optimal strategies (i.e., sequential, concurrent, or overlapped) to manage coupled design activities that share a fixed amount of engineering resources subject to performance and deadline constraints. Model analysis characterizes the solution space for the coupled development problem. The solution space is used to explore the generation of product performance and the associated dynamic forces affecting concurrent development practices. We use these forces to explain conditions under which concurrency is a desirable strategy.

*(Product Development; Design Process Modeling; Concurrent/Sequential/Overlapping Development; Component and System Performance Generation; Software Engineering)*

---

## 1. Introduction

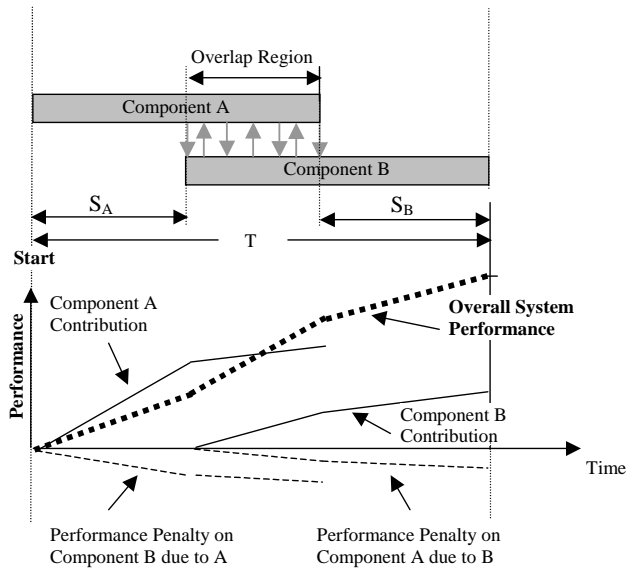
Product development (PD) is the process of transforming customer needs into an economically viable product that satisfies those needs. PD research spans many different disciplines ranging from organizational science (Brown and Eisenhardt 1995), marketing (Wind and Mahajan 1997), and engineering (Finger and Dixon 1989) to operations management (Smith and Morrow 1999, Krishnan and Ulrich 2001). Recent management science PD research has focused on approaches to reduce lead time, cut costs, and improve product quality. Concurrent engineering (CE) is one such approach (Wheelwright and Clark 1992, Griffin 1996, Terwiesch and Loch 1999). However, the risks associated with CE, such as increased communication overhead (Ha and Porteus 1995, Loch and Terwiesch 1998) or excessive iterations (Krishnan et al. 1997, Smith and Eppinger 1997a, 1997b), can result in increased development lead time and cost (AitSahlia et al. 1995). Consequently, a growing body of CE management models has been built to provide insights into the management of information, com-

munication, and dependencies among development activities.

CE can be used to either (a) reduce the development time without explicit consideration of product quality/performance issues, or (b) increase the product quality/performance for fixed development time. In this paper, we propose a model for improving the understanding of the latter—namely, concurrent product development with fixed development times. The goal of this paper is to provide insights about optimal strategies to manage coupled tasks that share a fixed amount of engineering resources subject to performance and deadline constraints. Tasks are defined as coupled when they depend on each other for input information.

The model developed in this paper is called the performance generation model (PGM) and is shown in Figure 1. It represents a hypothetical PD project consisting of two possibly overlapped design tasks. These tasks (A and B) involve upstream and downstream development in a design cycle, respectively. The model tracks the degree to which each task adds

**Figure 1** PGM Showing Component and System Performance Generation



to the overall performance in response to the effort devoted to it. Within the context of our model, performance is defined as a measure of the product's fidelity with respect to its requirements. Two examples of fidelity can be the speed of a microprocessor and the number of bugs eliminated from a new software release. We assume a simple production function for generating performance: The more time spent working on a task, the higher the level of performance that can be achieved; however, a deadline for the project has to be met. We assume that each task contributes to the overall performance at a different rate and at the same time deteriorates the performance of the other coupled task by generating rework for it, requiring effort to be expended on coordination, etc. Then, the core trade-off is to improve the overall performance by ensuring that neither task creates an unacceptable level of performance penalty for the other task. PGM is a tool for strategic assessment of concurrency early during the development process. That is, armed with an early assessment of performance generation rates, a manager can gain insights into the appropriate execution structure in terms of degree of concurrency.

PGM extends prior concurrent engineering models in general, and overlapping models in particular, in two ways. First, we address the problem of

improving PD performance subject to an imposed deadline. By explicitly accounting for deadlines, the model facilitates a better representation of many real development processes where the team is not only challenged with the task of developing a new product, but also with meeting the deadline. The deadline can be the launch of a new automobile model (2001 Ford Explorer), announced software release (MS Office 2000), or an intermediate milestone in a stage review process (Cooper and Kleinschmidt 1993). A second contribution of the PGM is explicit segregation of PD performance into component and system generation (Hoedemaker et al. 1999). Component performance refers to the contribution of individual tasks without regard to coupling effects. System performance measurements allude to the overall performance including the coupling effects. The model yields two main results:

1. We determine the optimal execution strategy in terms of recommended degree of overlap for the coupled development process that will maximize the overall product performance.
2. We characterize the solution space for coupled development projects. The coupling is manifested by differing rates of component and system performance accrual. These rates are used to explore the solution space in terms of dynamic forces affecting concurrent engineering practices and to derive conditions when concurrency is a desirable strategy.

The rest of the paper proceeds as follows. In the next section we provide a taxonomy for CE decision problems. Then we discuss major management science CE models and contrast them against our proposed model. In §§3 and 4, we introduce the PGM assumptions and formulation, and derive theorems governing the model behavior. In §5, we characterize the optimal policies for product development management in a deadline environment. In §6, we provide an example from a software development program to illustrate how the performance generation parameters are assessed. Managerial insights gained by studying this model are presented in §7. Finally, §8 presents our conclusion and sets the stage for future extensions to the base PGM.

## 2. Related Literature

In this section, we propose a taxonomy for CE product development decisions based on information dependencies and development strategies as shown in Table 1. The information dependencies between development tasks constitute the information processing view of the development process (Clark and Fujimoto 1991). The structure of development tasks is classified into three types (Eppinger et al. 1994): dependent, independent, and coupled. Two tasks are said to be *dependent* if one task depends on the other for input information. On the other hand, if both tasks depend on each other for input information, then the two tasks are *coupled*. Finally, if there is no information dependency between both tasks, then they are *independent*. The execution strategy view of the development process determines the development process schedule. Regardless of the information structure, two development tasks can be executed sequentially, overlapped, or concurrently (Yassine et al. 1999). The sequential execution of two development tasks requires the upstream task to be completely finished before the downstream task can be started. In the overlapped execution strategy, the upstream task is scheduled to start first and the downstream task starts before the completion of the upstream task. Finally, the simultaneous start and finish of both tasks characterize the concurrent execution strategy. Each box within this taxonomy can accommodate models that either aim to minimize the overall development time or maximize the performance subject to a fixed deadline. In the rest of this section, we discuss some of the models in this taxonomy that are relevant to our approach.

Smith and Eppinger (1997a, 1997b) present two analytical extensions to the design structure matrix method (Eppinger et al. 1994). In the first model, they use linear systems theory to identify controlling features of iteration in a coupled development process. In the second model, the ordering of tasks is manipulated and an expected duration for each task sequence is calculated. While both of these models are useful in characterizing the two extreme cases of product development (i.e., concurrent and sequential iteration) for any number of tasks, they do not model intermediate scenarios where overlapping might be

more appropriate. The PGM, on the other hand, considers the whole range of execution strategies for the development process (concurrent, sequential, and overlapping) and provides the optimal execution configuration.

Krishnan et al. (1997) construct a model for overlapping nominally sequential activities to reduce development lead time. In their model, the downstream activity begins with preliminary upstream information and incorporates subsequent upstream design changes in future iterations. They present a framework to determine how to disaggregate design information and overlap consecutive stages based on the evolution and sensitivity properties of the information exchanged. In contrast, our model assumes a continuous (i.e., without interruption) execution of each task, while Krishnan et al. allow for interruptions of the downstream task. In addition, while we seek to maximize product performance, Krishnan et al. utilize an objective function to minimize lead time.

Carrascosa et al. (1998) build a Markovian model that explores varying degrees of overlap between development tasks while attempting to minimize the development time. PGM differs from their formulation in two ways: It models performance maximization under a deadline constraint, and it segregates component and system performance generation.

Ha and Porteus (1995) determine the optimal number of design reviews within a coupled development process that minimizes the total lead time. Following this line of work, Loch and Terwiesch (1998) argue that the gain from overlapping activities must be weighed against the delay from downstream rework to determine the optimal overlapping magnitude and communication policy. In addition to the issue of deadline constraint, both of these models differ from ours in another important respect. They are concerned with the frequency of information transfer within a coupled development process that will minimize lead time, while our model assumes perfect communication and is concerned with the choice of the execution strategy that maximizes product performance.

Ahmadi and Wang (1999) develop a model that optimally places design reviews along the development process to minimize development risk. In addition, the model provides optimal resource allocation

**Table 1** Taxonomy of PD Execution-Related Decision Problems

		Information Structure		
		Dependent	Independent	Coupled
Executive Strategy	Sequential	Krishnan et al. (1997) Loch & Terwiesch (1998)	*	Smith & Eppinger (1997a) Loch & Terwiesch (1998) Ha & Porteus (1995)
		Cohen et al (1996) Ahmadi & Wang (1999)	*	PGM
	Overlap	Krishnan et al (1997) Loch & Terwiesch (1998)	*	Loch & Terwiesch (1998)
			*	PGM
	Concurrent	Carrascosa et al. (1998)	*	Smith & Eppinger (1997b)
			Repenning (1999)	PGM

Legend	
	Minimize Lead Time
	Maximize Performance
*	Irrelevant, unless resource constraints are considered

policies for each design stage. The PGM is similar to Ahmadi et al. in the way they set the development speed in each stage to minimize stage risk. Also, they address the question of allocating resources for each of the development stages. However, they neither allow for overlapping nor consider deadlines in their formulation, both of which are included in our model.

Cohen et al. (1996) examine the trade-off between product performance and profit as a function of a fixed sales window. More time spent on improving product development performance results in lost sales due to the fixed sales window. On the other hand, if the product is released immaturely, then profit is lost due to unsatisfied customers. The model analysis yields an optimal development time that maximizes profit. The Cohen et al. model is similar to the PGM in two respects. First, they include the concept of deadlines; however, their focus is on a deadline imposed by a marketing window and not a fixed launch date. Second, they utilize a similar production function for performance generation. However, their model ignores coupling and overlapping between development tasks.

There are very few analytical models that explore sequencing strategies for tasks that have an inde-

pendent information structure. Overlapping leads to interesting problems, if one assumes that resources are fungible and shared between independent tasks. Repenning (1999) has developed a system dynamics simulation model to address resource allocation between two separate projects while assuming a concurrent execution strategy.

### 3. Performance Generation Model (PGM) Formulation

Consider a hypothetical PD project comprising two coupled tasks as shown in Figure 1.<sup>1</sup> There are two decision variables:  $S_A$  and  $S_B$ , the amount of time spent by Task A and Task B working independently, respectively. The development project deadline is assumed to be time  $T$ . Thus, the amount of time spent while both tasks work concurrently is  $(T - S_A - S_B)$ . The goal of the model is to maximize the sum of the performance accumulated by both tasks at time  $T$ .

The performance contribution per activity is analogous to a Cobb-Douglas production function (Varian 1992). The formulation assumes that the performance

<sup>1</sup> We define the upstream task to be Task A and the downstream task to be Task B, without loss of generality.

of each task improves only by conducting work on it. The coupling or interdependency is modeled by performance deterioration in one task due to the rework generated by the other task. For instance, as Task A conducts some work to improve its own performance, it will generate a fraction ( $R_A$ ) of that work as rework for Task B, deteriorating B's performance. Similarly, work on Task B degrades Task A's performance by a fraction  $R_B$ . We assume that working on a task improves its own performance and it cannot produce more damage to other tasks (by deteriorating their performance) than the amount gained in its own performance. Then, the performance deterioration is a fraction less than unity ( $0 \leq R_A, R_B < 1$ ). This assumption is analogous to the rework fractions described in the WTM model (Smith and Eppinger 1997a).

The relationship between work and its contribution to performance, assuming linear return on labor, is described in Equations 1 and 2.<sup>2</sup>

$$\dot{X}_{Aj} = L\phi\alpha_{Aj} - R_B\dot{X}_{Bj}, \forall j \quad (1)$$

$$\dot{X}_{Bj} = L(1 - \phi)\alpha_{Bj} - R_A\dot{X}_{Aj}, \forall j, \quad (2)$$

where

$A$  and  $B$  denote the tasks;

$j$  denotes the region number/index,  $j = 1, 2, 3$ .

Regions are described as follows:

$j = 1$ : Only Task A is working;

$j = 2$ : Both Tasks A and B are working concurrently;

$j = 3$ : Only Task B is working.

$R_A(0 \leq R_A < 1)$  is a dimensionless quantity describing the penalty of Task A on Task B.

$R_B(0 \leq R_B < 1)$  is a dimensionless quantity describing the penalty of Task B on Task A.

$L$  is the maximum amount of available labor resources per unit time for the development project.<sup>3</sup> The unit for  $L$  is dollars (\$)/time.

<sup>2</sup> We have presented the formulation with  $\beta = 1$  (where  $\beta$  is the return to scale on labor), for ease of exposition. The core formulation remains the same for a generic Cobb-Douglas function except for substituting  $L$  by  $L^\beta$ ,  $\phi$  by  $\phi^\beta$ , and  $(1 - \phi)$  by  $(1 - \phi)^\beta$ . These substitutions do not affect any of the results in §4 in terms of the existence of optimal decision space. The model insights are valid for any value of  $\beta$ .

<sup>3</sup> Since  $\dot{X}_{ij}$  is strictly increasing in  $L$ , we will always use all of the available resources at any time during the development process.

$X_{Aj}(t)$  and  $X_{Bj}(t)$  are the performance achieved by Tasks A and B in region  $j$  at time  $t$ .

$\dot{X}_{Aj}$  and  $\dot{X}_{Bj}$  represent the rate of performance change for Tasks A and B, respectively. That is,

$$\dot{X}_{Aj} = \frac{dX_{Aj}}{dt} \quad \text{and} \quad \dot{X}_{Bj} = \frac{dX_{Bj}}{dt};$$

$$\dot{X}_{ij} > 0 \Rightarrow \text{Performance Improvement,}$$

$$\dot{X}_{ij} < 0 \Rightarrow \text{Performance Deterioration.}$$

$\phi$  is the fraction of  $L$  used for working on Task A.<sup>4</sup>

$$\text{for } j = 1, \quad \phi = 1$$

$$\text{for } j = 2, \quad 0 < \phi < 1$$

$$\text{for } j = 3, \quad \phi = 0.$$

$\alpha_{Aj}$  and  $\alpha_{Bj}$  are defined as performance generation rates. Unit for  $\alpha$  is performance/\$. In other words,  $\alpha$  is a measure of the productivity for labor resources devoted to the task.

Notice that because  $\phi = 1$  in Region 1 and  $\phi = 0$  in Region 3, the performance contribution from  $\alpha_{B1}$  and  $\alpha_{A3}$  is irrelevant to the solution of the problem. Hence, there are only seven input parameters; namely,  $\alpha_{A1}, \alpha_{A2}, \alpha_{B2}, \alpha_{B3}, \phi, R_A,$  and  $R_B$ . Table 2 summarizes the performance generation contribution of upstream and downstream tasks.

The overall product performance for a given region  $j$  is

$$\int_{(t_j)\text{initial}}^{(t_j)\text{final}} [\dot{X}_{Aj}(t) + \dot{X}_{Bj}(t)] dt, \forall j, \quad (3)$$

where  $(t_j)\text{initial}$  and  $(t_j)\text{final}$  are the start and finish times for region  $j$ .

Rewriting Equation (3) over the whole development process (i.e., from Time 0 to  $T$ ) results in

$$X_{Aj}(t) = L[\phi\alpha_{Aj} - (1 - \phi)R_B\alpha_{Bj}]t + c_{Aj} \quad (4)$$

and

$$X_{Bj}(t) = L[(1 - \phi)\alpha_{Bj} - \phi R_A\alpha_{Aj}]t + c_{Bj}. \quad (5)$$

<sup>4</sup> We will assume that  $\phi$  is constant. This implies that the labor resource allocation during overlapping (i.e., Region 2) is not fungible between Tasks A and B. Later, we will explore the sensitivity of the optimal solution to alternate resource allocation policies.

**Table 2** Summary of Performance Generation Functions by Region

Region ( $j$ )	1	2	3
Bounds on Time $t$	$0 \leq t < S_A$	$S_A \leq t < (T - S_B)$	$(T - S_B) \leq t \leq T$
Performance of A	$\alpha_{A1} t$	$[\phi\alpha_{A2} - (1 - \phi)\alpha_{B2}R_B]t$	$-\alpha_{B3}R_B t$
Performance of B	$-\alpha_{A1}R_A t$	$[(1 - \phi)\alpha_{B2} - \phi\alpha_{A2}R_A]t$	$\alpha_{B3}t$
Overall system performance	$\alpha_{A1}(1 - R_A)t$	$[\phi\alpha_{A2}(1 - R_A) + \alpha_2(1 - \phi)(1 - R_B)]t$	$\alpha_{B3}(1 - R_B)t$

Constants  $c_{ij}$  are determined by the boundary conditions of three different regions ( $j = 1, 2, 3$ ).<sup>5</sup> This yields expressions for  $X_{A3}(T)$  and  $X_{B3}(T)$ , the performance of Tasks A and B at time  $T$ ,

$$X_{A3}(T) = L\{\alpha_{A1}S_A + [\phi\alpha_{A2} - (1 - \phi)R_B\alpha_{B2}] \times (T - S_A - S_B) + (-R_B\alpha_{B3})S_B\}, \quad (6)$$

$$X_{B3}(T) = L\{-R_A\alpha_{A1}S_A + [(1 - \phi)\alpha_{B2} - \phi R_A\alpha_{A2}] \times (T - S_A - S_B) + \alpha_{B3}S_B\}. \quad (7)$$

The Objective Function  $\text{Max}\{X_{A3}(T) + X_{B3}(T)\}$  becomes

$$\text{Max}_{S_A, S_B} L\{\alpha_{A1}(1 - R_A)S_A + [\phi\alpha_{A2}(1 - R_A) + (1 - \phi)\alpha_{B2} \times (1 - R_B)](T - S_A - S_B) + \alpha_{B3}(1 - R_B)S_B\} \quad (8)$$

$$\text{s.t. } X_{A1}(0) = 0, \quad (8a)$$

$$X_{B1}(0) = 0, \quad (8b)$$

$$X_{A3}(T) \geq 0, \quad (8c)$$

$$X_{B3}(T) \geq 0, \quad (8d)$$

$$S_A + S_B \leq T. \quad (8e)$$

The Objective Function (8) maximizes the overall project performance. Constraints (8a) and (8b) are the initial starting conditions of the development process where no performance has been accumulated by either task. The nonnegativity constraints (8c) and (8d) guarantee that an optimal solution by the deadline ( $T$ ) will only include situations where both tasks

complete all required rework. If either constraint is binding at the optimal solution, then the task has performed just enough work to raise its performance to the minimum acceptable level of zero.<sup>6</sup> Finally, Constraint (8e) reflects the project deadline.

#### 4. Analysis of the Optimal Policies

Instead of exploring the gradients of the objective function, we choose to present a sequence of arguments that exploits the properties of the 7-tuple input parameters ( $\alpha_{A1}, \alpha_{A2}, \alpha_{B2}, \alpha_{B3}, \phi, R_A$ , and  $R_B$ ). In doing so, we derive the expressions for the optimal values of the decision variables,  $S_A$  and  $S_B$ , in terms of the 7-tuple input parameters. The proof proceeds in the following sequence. We first derive the optimal values for  $S_A$  and  $S_B$  for all possible execution strategies. Then we derive conditions under which each of these strategies is optimal. We complete our proof by showing that these conditions cover an exhaustive map of all the values that the input parameters can assume. These optimal choices map into a solution space representing the selection of sequential, overlapping, or concurrent development strategies.

In the rest of this section, we will state all the lemmas, theorems, and corollaries. All proofs are provided in the Appendix.

**LEMMA 1.** *If  $R_A, R_B < 1$ , then overall project performance is a nondecreasing function in time.*

**THEOREM 1.** *When one schedules two coupled tasks with respect to a nondecreasing performance measure, it is not necessary to consider schedules that involve idle time.*

**LEMMA 2. (THE SEE-SAW RULE).** *Given a pair of adjusted performance generation rates  $[\phi\alpha_{Aj}(1 - R_A)]$  and*

<sup>6</sup> The model will work for any threshold. Zero was selected as an arbitrary value.

<sup>5</sup>

$X_{A1}(0) = 0, X_{B1}(0) = 0,$   
 $X_{A1}(S_A) = X_{A2}(0), X_{B1}(S_A) = X_{B2}(0),$   
 $X_{A2}(T - S_B) = X_{A3}(0), X_{B2}(T - S_B) = X_{B3}(0).$

$[(1 - \phi)\alpha_{Bj}(1 - R_B)]$ , it is always optimal to perform work on the task with the largest adjusted performance generation rate, if Constraints (8c) and (8d) are not violated.<sup>7</sup>

**THEOREM 2.** *If the sequential strategy is optimal, then the corresponding solution for  $S_A^*$  is bounded by*

$$\frac{\alpha_{B3}}{\frac{\alpha_{A1}}{R_B} + \alpha_{B3}} T \leq S_A^* \leq \frac{\alpha_{B3}}{\alpha_{A1}R_A + \alpha_{B3}} T. \quad (9)$$

**COROLLARY 2.1.** *If  $\alpha_{A1} < \alpha_{B3}$ , then  $S_A^*$  takes the lower bound in (9).*

**COROLLARY 2.2.** *If  $\alpha_{A1} > \alpha_{B3}$ , then  $S_A^*$  takes the upper bound in (9).*

**COROLLARY 2.3.** *If  $\alpha_{A1} = \alpha_{B3}$ , then  $S_A^*$  takes any value between the bounds in (9).*

**THEOREM 3.** *If the overlap strategy is optimal and  $S_A^* = 0$ , then the corresponding solution for  $S_B^*$  is bounded by*

$$\begin{aligned} \frac{R_A\phi\alpha_{A2} - (1 - \phi)\alpha_{B2}}{\alpha_{B3} + R_A\phi\alpha_{A2} - (1 - \phi)\alpha_{B2}} T &\leq S_B^* \\ &\leq \frac{\phi\alpha_{A2} - (1 - \phi)R_B\alpha_{B2}}{R_B\alpha_{B3} + \phi\alpha_{A2} - (1 - \phi)R_B\alpha_{B2}} T. \end{aligned} \quad (10)$$

**THEOREM 4.** *If the overlap strategy is optimal and  $S_B^* = 0$ , then the corresponding solution for  $S_A^*$  is bounded by*

$$\begin{aligned} \frac{(1 - \phi)R_B\alpha_{B2} - \phi\alpha_{A2}}{\alpha_{A1} + (1 - \phi)R_B\alpha_{B2} - \phi\alpha_{A2}} T &\leq S_A^* \\ &\leq \frac{(1 - \phi)\alpha_{B2} - R_A\phi\alpha_{A2}}{\alpha_{A1}R_A + (1 - \phi)\alpha_{B2} - R_A\phi\alpha_{A2}} T. \end{aligned} \quad (11)$$

To explore the optimality conditions, we rewrite Equations (6) and (7) combined with Constraints (8c) and (8d) while collecting terms for the decision variables  $S_A$  and  $S_B$ ,

$$F_1S_A - F_2S_B + B_1T \geq 0, \quad (12)$$

$$F_3S_B - F_4S_A + B_2T \geq 0, \quad (13)$$

where

$$F_1 = \alpha_{A1} - \phi\alpha_{A2} + (1 - \phi)R_B\alpha_{B2},$$

<sup>7</sup> Lemma 2 is equivalent to taking the partial derivative of Equation (8) with respect to  $S_A$  and  $S_B$ .

$$F_2 = \phi\alpha_{A2} - (1 - \phi)R_B\alpha_{B2} + R_B\alpha_{B3},$$

$$F_3 = \alpha_{B3} - (1 - \phi)\alpha_{B2} + \phi R_A\alpha_{A2},$$

$$F_4 = R_A\alpha_{A1} + (1 - \phi)\alpha_{B2} - \phi R_A\alpha_{A2},$$

$$B_1 = \phi\alpha_{A2} - (1 - \phi)R_B\alpha_{B2},$$

$$B_2 = (1 - \phi)\alpha_{B2} - \phi R_A\alpha_{A2}.$$

In effect, we transform the 7-tuple input parameters into another set of four composite parameters ( $F_1, F_2, F_3$ , and  $F_4$ ) that account for interactions between the original input parameters.<sup>8</sup> In the rest of the paper we will deal exclusively with these transformed parameters to characterize the optimal solution space. The transformed parameters are called *performance generation coefficients* and are interpreted in §5.4.

**LEMMA 3A.** *While maximizing the performance of Task A, with  $F_1 > 0$  and  $F_3 \leq 0$ , a comparison of the generation coefficients  $F_1$  and  $F_4$  determines whether Task A should overlap with Task B.*

**LEMMA 3B.** *While maximizing the performance of Task B, with  $F_1 > 0$  and  $F_3 \leq 0$ , a comparison of the generation coefficients  $F_2$  and  $F_3$  determines whether Task A should overlap with Task B.*

**THEOREM 5.** *The tree shown in Figure 2 provides an exhaustive mapping of conditions for an optimal solution based on all possible values of the transformed input parameters,  $F_1, F_2, F_3$ , and  $F_4$ .*

**COROLLARY 5.1.** *Policies outside the region ABCODA (in Figure 3) are either infeasible or result in suboptimal performance.*

## 5. Discussion

The discussion is structured in four parts. First, we provide a characterization of the optimal decision space as it appears in Figure 3. Next, we discuss the sensitivity of the optimal solution to resource allocation policy. Then, we discuss the dynamics of concur-

<sup>8</sup> T is taken out of the discussion by expressing the final results in  $S_A/T$  and  $S_B/T$ , respectively. Consequently,  $B_1$  and  $B_2$  are irrelevant because they are interactions associated with  $T$ .



on Line BC'. To increase the duration of Task A further beyond Point B, we need to compensate for that by simultaneously increasing the duration of Task B. Thus, the solution moves from Point B to Point C along Line BC. Point C represents the "Late Overlapping" strategy. The solutions along Line CO are characterized by working on Task A all the time (i.e., from Time 0 until the deadline). However, the duration of Task B increases as the solution moves from Point C toward Point O. When Point O is reached, both tasks work concurrently. The reverse explanation holds when the solution moves from Point O to Point A along Edge ODA.

### 5.2. Sensitivity of the Optimal Solution Space to Resource Allocation Policy

Assuming that labor resources are fungible in Region 2, we can look at the impact of changing the resource allocation ( $\phi$ ) during overlap of Tasks A and B by further exploring the delta-wing-shaped polygon (ABCODA), as shown in Figure 3. Note that  $\phi$  is irrelevant on Line AB. Furthermore, Edge AD of the delta wing represents the locus of all optimal solutions when  $X_{A3}(T) = 0$ . Similarly, Edge BC represents the locus of all optimal solutions when  $X_{B3}(T) = 0$ . Reducing  $\phi$  collapses the delta wing by shifting the location of Lines BC and AD toward the Origin O while changing their original slopes as they move inward.<sup>9</sup> Finally, the leading edge of the delta wing (DOC) represents the line of constant product performance derived by changing  $\phi$  such that the optimal solution moves from  $X_{A3}(T) = 0$  (at Point D) to  $X_{B3}(T) = 0$  (at Point C).

### 5.3. Dynamics of Concurrency

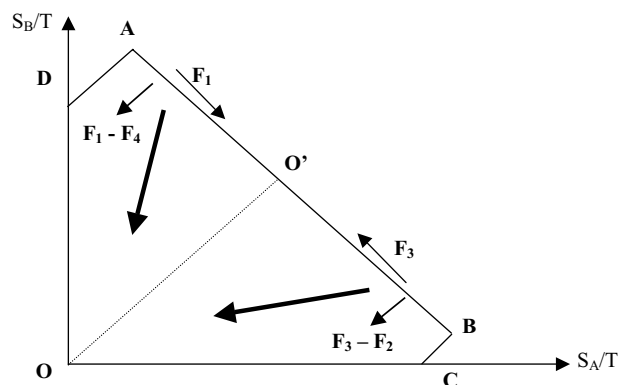
The generation coefficients,  $F_i$ , may also be used to describe composite forces that drive the solution

<sup>9</sup> It is worth noting that a similar shift would also happen if Constraints (8c) and (8d) require a nonzero threshold value. In a limiting case, Line BC will eventually coincide with Line OO', the bisector of the orthogonal axes, and Line AD will coincide with OO'. In this limiting scenario, it is obvious that  $S_A = S_B$ . However, the performance contribution of each task is not necessarily equal unless the problem parameters for Task A and Task B are symmetrical.

within the optimal decision space as shown in Figure 4.<sup>10</sup> Notice that a positive force in the direction A to B moves the optimal solution toward a larger  $S_A$  and smaller  $S_B$ . Similarly, for a fixed ratio ( $S_A/S_B$ ), a positive force in the direction O to O' moves the optimal point toward sequential strategy, and a negative force moves the optimal solution toward concurrency.

Examining the tree logic given in Figure 2, we observe that the interaction between the four composite forces determines the location of the optimal solution. These forces are  $F_1, F_3, (F_1 - F_4)$ , and  $(F_3 - F_2)$ . For instance, the bold arrows in Figure 4 depict the resultant force leading toward a concurrent scenario when  $F_1 > 0, F_3 > 0, (F_1 - F_4) < 0$ , and  $(F_3 - F_2) < 0$ . It is sufficient to know the direction in which the resultant points, because based on the discussion in the previous section we have established that the optimal solution will lie on the corner point of the Polygon ABCODA. These forces can be used to visualize the dynamics of concurrency in the following sense: If one or more of the 7-tuple input parameters are changed from a base setting, then one can compute the new generation coefficients,  $F_i$ , and use the new composite forces to establish the direction in which the optimal solution will shift.

Figure 4 The Dynamics of Concurrent Engineering



<sup>10</sup> The evolution coefficients can be thought of as forces for studying the dynamics of concurrency. Caveat: These coefficients are not analogous to any physical force.

#### 5.4. Component and System Performance Generation

In this section, we discuss performance generation in terms of the generation coefficients. Recall that we have defined performance generation as the rate at which performance is improved by conducting work. Our performance generation construct is analogous to the definition of PD evolution in the literature (Krishnan et al. 1997). However, we distinguish between two different types of performance generations: component and system. Recall that component performance refers to the contribution of individual tasks without regard to the coupling effects. The system performance is a result of combining component performance and associated rework while accounting for the degree of overlap between upstream and downstream tasks. Thus, managers need to distinguish between upstream and downstream components and system performance generation.

**Component Performance Generation.** For the upstream task, component performance generation is labeled as *fast* if a higher rate of performance accumulation is attained in Region 1 as compared to Region 2. Alternatively, if the upstream task has a higher rate in Region 2, then we refer to it as *slow*. According to this definition,  $F_1 > 0$  represents fast upstream component performance generation.<sup>11</sup> Similarly,  $F_1 < 0$  represents slow upstream component performance generation.

For the downstream task, slow generation is characterized by a slower performance accumulation rate in Region 2 (compared to Region 3), and fast generation occurs when the performance accumulation rate in Region 2 is bigger than that of Region 3. The performance generation of the downstream task is also determined by inspecting  $F_3$ . If  $F_3 > 0$ , then the downstream component has a slow rate of performance generation.<sup>12</sup> Conversely, a downstream component has a fast rate of performance generation when  $F_3 < 0$ .

**System Performance Generation.** At the system level, the upstream performance generation is labeled

as slow if the overall (i.e., project or system) performance accumulation rate in Region 1 is smaller than that of Region 2. The overall performance is measured not only by how much the upstream task gains through conducting work (i.e., component performance generation), but also by how much the downstream task loses in this process. Slow system performance generation for the upstream task is characterized by  $(F_1 - F_4) < 0$ . Alternatively, the upstream task exhibits fast system performance generation when  $(F_1 - F_4) > 0$ . Downstream system performance generation is labeled as slow when the overall performance gain in Region 2 is smaller than that of Region 3. It is labeled as fast when the reverse is true. The condition for slow downstream system performance generation is  $(F_3 - F_2) > 0$  and for fast system performance generation is  $(F_3 - F_2) < 0$ .

## 6. Assessment of Component and System Performance Generation

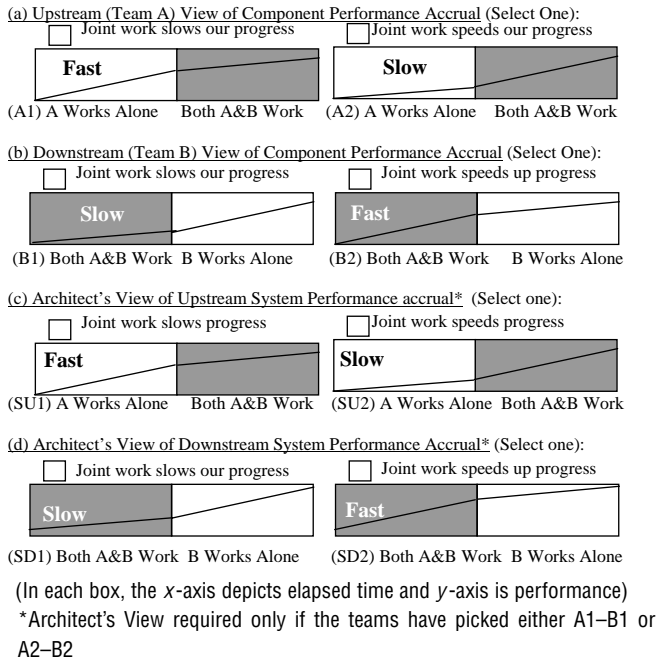
The following section describes a subjective assessment of the component and system performance generation. Component generation coefficients are assessed by interviewing domain experts. Generally, component expertise resides locally within an engineering team. System generation coefficients are assessed by interviewing system experts. Generally, system expertise resides with system architects, managers overlooking the overall development, and/or system integration and test teams. Recent literature on subjective interviews of product development projects suggests that different generation rates have been ascribed to tasks by different experts, based on their vantage point within the development process (Ford and Sterman 1998).

We illustrate the assessment process using an example from a software development project. Relevant data are gathered by circulating a survey instrument (see Figure 5) to individuals responsible for coding, system architecture, and program management. Our subject, Softex (a fictitious company name), is in the business of developing e-commerce solutions that integrate legacy systems and processes across multiple companies into a unified digital marketplace. Their development process involves the integration

<sup>11</sup>  $F_1$  is always positive when  $\alpha_{A1} > \alpha_{A2}$ , for all  $R_A$  and  $R_B < 1$ . The reverse statement is also true.

<sup>12</sup>  $F_3$  is always positive when  $\alpha_{B3} > \alpha_{B2}$ , for all  $R_A$  and  $R_B < 1$ . The reverse statement is also true.

**Figure 5 Assessment of Performance Generation**



of an off-the-shelf e-commerce system with custom-developed software components. The example does not reflect the specific details of the project, but represents a “typical and plausible” representation of the actual performance generation rates.

The product specification involves about 10,000 function points, representing a moderate to high degree of development complexity, and requires task coordination among more than 25 developers. Developers are quite sophisticated in process management. They hold a level V rating based on the Systems Engineering Capability Maturity Model<sup>®</sup>. It is a standard practice at Softex to build a system-level behavioral model of the product at the very beginning of the project using the Unified Modeling Language. This modeling exercise yields preferred data models, use models, and interaction diagrams. These artifacts provide a clear sense of the level of coupling and performance penalties among various local tasks and their contribution to system performance. These data are the basis for the subjects’ responses to the assessment questionnaire.

In a segment of the project that deals with the integration of the web front-end with the legacy back

end, teams respond to the questionnaire by establishing relative shapes for the generation functions as depicted by Figure 5(a) and 5(b), respectively. Code developers within individual teams, in this case Team A (Web Front-End) and Team B (Legacy Back-End), are asked to assess performance generation. For instance, slow or fast shapes for component performance generation are captured by asking Team A to pick either Shape A1 or Shape A2 as shown in Figure 5(a). Team B is asked to select either B1 or B2 in Figure 5(b). The context in which teams assess performance generation relates to their own performance when they are working by themselves or in conjunction with the other team. Individual teams know if the joint work actually speeds up or slows down their own progress. However, teams do not know how much rework they create for each other. Interviews with system managers reveal the impact of rework on the relative progress. They respond to the questions in Figure 5(c) and 5(d).

If these teams pick the combinations A1-B1 or A2-B2, then there is no ambiguity in terms of the preferred execution strategy, i.e., they both wish to either work together or work separately. However, if the component teams select combinations A1-B2 or A2-B1, then there exists a conflict between Teams A and B’s desired development sequence. In these situations, the system architect answers the question: What is the rate of system performance accrual for the upstream and the downstream component, respectively, while accounting for the coupling effects? Figure 5(c) and 5(d) capture the choices available to the system architect: SU1 (Upstream is Fast) or SU2 (Upstream is Slow) and SD1 (Downstream is Fast) or SD2 (Downstream is Slow). In the Softex case, data indicated that the A1-B1 scenario governed the development process.

Field observations at Softex confirm that measurement of performance generation coefficients, especially early in the development process, are imprecise. However, a qualitative assessment of the performance generation rates is possible for the purpose of applying the PGM. These observations are consistent with the fuzzy nature of the front-end during product development (Khurana and Rosenthal 1997). Given the nature of these data, a strategic model to provide early insights about structure of

the process is managerially more relevant and feasible than a tactical/operational model. A framework for informing managerial decisions based on the rates of performance generation is presented in the next section. We will utilize the assessments described here to illustrate the use of the framework.

## 7. Managerial Implications

To facilitate managerial utility of the PGM, we have transformed the optimal decision trees into a conceptual framework as shown in Figure 6. This framework is built around the concept of performance generation, both at the component and at the system level. Managers can utilize the generation coefficients within the framework to structure the development process (i.e., choice of a sequential, concurrent, or overlapped process).

The subjective assessment of the generation coefficients described in the previous section is needed for utilizing this framework. Comparison of component and system performance generations yields the following four cases:

*Case 1.* When the component performance generation for Task A is slow ( $F_1 \leq 0$ ) and component performance generation for Task B is fast ( $F_3 \leq 0$ ), a concurrent strategy is optimal. The rationale underlying this strategy can be explained as follows: The upstream task contributes less in Region 1 than it does in Region 2, and the downstream task contributes more in Region 2 than in Region 3; therefore, it is optimal to conduct all the work in Region 2.

*Case 2.* When the component performance generation for Task A is fast ( $F_1 > 0$ ) and component performance generation for Task B is slow ( $F_3 > 0$ ), a sequential strategy is optimal. The rationale underlying this strategy is exactly the reverse of Case 1. Both activities accumulate more performance independently than when they are concurrent; therefore, it is optimal not to conduct any work in Region 2.

*Case 3.* When the component performance generation for Task A is fast ( $F_1 > 0$ ) and component performance generation for Task B is also fast ( $F_3 < 0$ ), then we need to check the system performance generations for these tasks. If the system performance generation for the upstream task (A) is slow (i.e.,

$F_1 - F_4 < 0$ ) and the system performance generation for the downstream task (B) is fast (i.e.,  $F_3 - F_2 < 0$ ), then the concurrent strategy is optimal. In this case, the feedback during overlapping increases the performance of both tasks relative to the situation where they work independently. If the situation is reversed (i.e., upstream system performance generation is fast and downstream system performance generation is slow), then conducting work concurrently decreases the rate of performance generation for both tasks, compared to working independently, and the sequential strategy becomes optimal. Finally, if only one task benefits from the feedback during overlapping (i.e., does better by conducting work concurrently with the other task as compared to when working independently), then either early or late overlapping results in the optimal strategy. Late overlapping is optimal when both system performance generations are fast, while early overlapping is optimal when both system performance generations are slow. Case 4 follows similar rationale.

Drawing on the stylistic assessments from the previous section to illustrate the use of the framework, recall that the two teams from Softex picked Scenarios A1 and B1. This implies that the upstream component generation is fast, and the downstream component generation is slow. Thus, it is best that the teams work in a sequential manner. This development structure is possible because the product architecture (i.e., work flow and data structure) for the web interface and the legacy system has minimized the impact of the coupling effect.

We asked the project architect at Softex about the use of the framework in more general settings. In the architect's view, the process of *ex ante* performance generation assessments allows the development team to compare and contrast assumptions about the relative rates of performance accrual and coupling penalties. In some instances when the developers pick either A1-B2 or B2-A1 as their scenario, there is a conflict between the upstream and downstream team preferences. The architect is then called on to review the interaction diagrams and decide on the overall sequence based on the system performance requirements.

Figure 6 Optimal Strategies Based on the Generation Coefficients

		A		Upstream Component Generation				
				Slow ( $F_1 \leq 0$ )		Fast ( $F_1 > 0$ )		
Downstream Component Generation	Fast ( $F_3 \leq 0$ )	C		System Perform. Generation		Upstream ( $F_1 - F_4$ )		
						Slow ( $<0$ )	Fast ( $\geq 0$ )	
	Downstream ( $F_3 - F_2$ )					C	LO	
	Slow ( $>0$ )			Fast ( $\leq 0$ )	EO	S		
Downstream Component Generation	Slow ( $F_3 > 0$ )	System Perform. Generation		Upstream ( $F_1 - F_4$ )		S		
				Slow ( $<0$ )	Fast ( $\geq 0$ )			
	Downstream ( $F_3 - F_2$ )				C			LO
	Slow ( $>0$ )	Fast ( $\leq 0$ )	EO	S				

**Legend**

C = Concurrent

S = Sequential

EO = Early Overlap

LO = Late Overlap

## 8. Conclusion

The trade-off captured in this model allows for the optimization of development resources (as represented by the choice of  $S_A$  and  $S_B$ ), with the goal of maximizing project performance. The PGM enriches PD literature by a new model that does not limit itself with time minimization concerns. It models resource constraints more realistically than the literature that postulates that more concurrency is better without considering resources. Moreover, our observations in an industrial setting show that the model provides process management insights and helps managers structure the PD process even with imprecise inputs, especially early in the development process.

We have kept the model sparse to gain clear managerial insights using a small number of parameters and assumptions. Our core assumptions, namely fixed deadlines, interactions through a two-way information exchange, rework, and minimal performance thresholds for individual components, are valid in a vast majority of product development projects. It is also instructive to point out that some of the managerial insights from PGM (e.g., the need for concurrency under certain settings) duplicate results generated by fundamentally different models aimed at minimizing development time (Loch and Terwiesch 1998, Car-

rascosa et al. 1998, Ahmadi and Wang 1999, and Hoedemaker et al. 1999).

Discussions with the architect at Softex exposed some limitations of the PGM model. Product development teams have multiple tasks within a single project and have to run multiple projects simultaneously. While the generic results shown by the PGM framework are viewed to be logical, management is concerned with applying these insights in settings where resource levels might not be fixed. Management at Softex has expressed interest in further exploration of the resource allocation issue in such settings.

The PGM can be extended in several ways. One might view Task A as the amount of product performance that is being provided by a supplier, and Task B as the amount of work being done by a principal. Thus, the PGM model provides a platform for optimal information exchange between the principal and the supplier, such that the product performance is maximized. It is also possible to extend the results through the lens of game theory with respect to the actions of two divisions of a firm responsible for Components A and B (Lewis and Mistree 1998). In another extension, A and B can be viewed as two consecutive product development processes whose completions are subject to a periodic deadline (Repenning 1999), as shown in the taxonomy of Table 1. In this scenario,  $\phi$ , the resource allocation fraction, will be an explicit deci-

sion variable. Then, resource constraints in the PGM can be reinterpreted while associating different costs to tasks, and the problem can be examined as a margin maximization exercise.

In summary, the key managerial insight from this model is that concurrent engineering need not be the optimal work strategy in many settings. Managers must consider the information exchanges, rework issues, performance thresholds, and resource restrictions while structuring their development projects. This result is contrary to the conventional wisdom that recommends use of task concurrency (Lawson and Karandikar 1994). The genesis of this counterintuitive result lies in the trade-off between the gain in project performance due to working on a task weighed against performance deterioration caused by the other coupled task. Further, we have developed a decision space for executing two coupled development tasks and established the dynamics of the sequential/concurrent/overlapping strategies. On one hand, our decision space allows an explanation of the forces that play a leading role in driving the optimal strategy toward full concurrent engineering. On the other hand, we show what forces prevent the system from drifting toward that point of full concurrency. In doing so, we provide managers with a tool to control the degree of concurrency in the process by examining the rates of performance generation.

## Acknowledgments

The authors thank Thomas Roemer, Durward Sobek, Jen Jooter, and two anonymous referees for helpful comments on an earlier version of this paper. We are grateful to the development team at Softex for access and for answering questions about their PD process.

## Appendix—Proofs

LEMMA 1. If  $R_A, R_B < 1$ , then overall project performance is a non-decreasing function in time.

PROOF. The overall performance of the project in the three regions is given in Table 2. Looking at the overall performance, it is evident that if  $R_A$  and  $R_B < 1$ , then all the terms are positive. Therefore, the overall performance of the project is nondecreasing.  $\square$

THEOREM 1. When one schedules two coupled tasks with respect to a nondecreasing performance measure, it is not necessary to consider schedules that involve idle time.

PROOF. Consider a schedule  $S$  without idle time. Assume that we have inserted idle time within the interval  $[0, T]$  of  $S$ ; namely, from Time  $t_1$  to  $t_2$  ( $0 \leq t_1 < t_2 \leq T$ ). Call this schedule  $S'$ .  $S'$  can take three different forms based on the values of  $t_1$  and  $t_2$ .

Case 1 ( $t_1 = 0$ ): If we transform the time axis from 0 to  $t_2'$ , then by Lemma 1 the performance level achieved at Time  $t$  (in  $S'$ ) is less than the performance level achieved at Time  $t < T$  (in  $S$ ).

Case 2 ( $t_2 = T$ ): Assume we have an optimal schedule for  $0 \leq t \leq t_1$ . This schedule can be stretched by multiplying each segment of  $S'$  by factor  $(T - t_1)/t_1$ . This new schedule will result in a greater performance level using Lemma 1.

Case 3 ( $0 < t_1 < t_2 < T$ ): Under any of the three execution strategies, it is clear that the performance of  $S$  is more than or equal to the performance of  $S'$  due to Lemma 1. Therefore, it is sufficient to only consider schedules similar to  $S$  in that they do not contain any period of inserted idle time.  $\square$

LEMMA 2. (THE SEE-SAW RULE). Given a pair of adjusted performance generation rate,  $\phi\alpha_{A_j}(1 - R_A)$  and  $(1 - \phi)\alpha_{B_j}(1 - R_B)$ , it is always optimal to perform work on the task with the largest adjusted performance generation rate, if Constraints (8c) and (8d) are not violated.

PROOF: Because  $\phi\alpha_{A_j}(1 - R_A) > 0$  and  $(1 - \phi)\alpha_{B_j}(1 - R_B) > 0$ , and the objective function (Equation (8)) is a linear combination of  $\alpha_{ij}(1 - R_i)$ , it is always better to work on the task associated with the largest adjusted performance generation rates for every  $j$  ( $j = 1, 2, 3$ ).  $\square$

THEOREM 2. If the sequential strategy is optimal, then the corresponding solution for  $S_A^*$  is bounded by

$$\frac{\alpha_{B3}}{\frac{\alpha_{A1}}{R_B} + \alpha_{B3}} T \leq S_A^* \leq \frac{\alpha_{B3}}{\alpha_{A1}R_A + \alpha_{B3}} T. \quad (9)$$

PROOF: Lower bound:  $X_{A3}(T) \geq 0 \Rightarrow \alpha_{A1}S_A \geq R_B\alpha_{B3}(T - S_A)$ ;  
Upper bound:  $X_{B3}(T) \geq 0 \Rightarrow R_A\alpha_{A1}S_A \leq \alpha_{B3}(T - S_A)$   $\square$

COROLLARY 2.1. If  $\alpha_{A1} < \alpha_{B3}$ , then  $S_A^*$  takes the lower bound in (9).

COROLLARY 2.2. If  $\alpha_{A1} > \alpha_{B3}$ , then  $S_A^*$  takes the upper bound in (9).

COROLLARY 2.3. If  $\alpha_{A1} = \alpha_{B3}$ , then  $S_A^*$  takes any value between the bounds in (9).

PROOF. All above corollaries are true using Lemma 2.  $\square$

THEOREM 3. If overlap strategy is optimal and  $S_A^* = 0$ , then the corresponding solution for  $S_B^*$  is bounded by

$$\begin{aligned} \frac{R_A\phi\alpha_{A2} - (1 - \phi)\alpha_{B2}}{\alpha_{B3} + R_A\phi\alpha_{A2} - (1 - \phi)\alpha_{B2}} T &\leq S_B^* \\ &\leq \frac{\phi\alpha_{A2} - (1 - \phi)R_B\alpha_{B2}}{R_B\alpha_{B3} + \phi\alpha_{A2} - (1 - \phi)R_B\alpha_{B2}} T. \end{aligned} \quad (10)$$

PROOF. Lower bound:  $X_{A3}(T) \geq 0 \Rightarrow \{\phi\alpha_{A2} - (1 - \phi)R_B\alpha_{B2}\}(T - S_B) \geq R_B\alpha_{B3}S_B$ ;

Upper bound:  $X_{B3}(T) \geq 0 \Rightarrow \{(1 - \phi)\alpha_{B2} - \phi R_A\alpha_{A2}\}(T - S_B) \geq \alpha_{B3}S_B$ .  $\square$

**Table 3** Performance Contribution of Tasks A and B in Regions 1 and 2

Region ( <i>j</i> )	1	2
Perf. (Task A)	$\alpha_{A1}$	$\phi\alpha_{A2} - (1 - \phi)\alpha_{B2}R_B$
Perf. (Task B)	$-\alpha_{A1}R_A$	$(1 - \phi)\alpha_{B2} - \phi\alpha_{A2}R_A$

**THEOREM 4.** *If the overlap strategy is optimal and  $S_B^* = 0$ , then the corresponding solution for  $S_A^*$  is bounded by*

$$\begin{aligned} & \frac{(1 - \phi)R_B\alpha_{B2} - \phi\alpha_{A2}}{\alpha_{A1} + (1 - \phi)R_B\alpha_{B2} - \phi\alpha_{A2}} T \leq S_A^* \\ & \leq \frac{(1 - \phi)\alpha_{B2} - R_A\phi\alpha_{A2}}{\alpha_{A1}R_A + (1 - \phi)\alpha_{B2} - R_A\phi\alpha_{A2}} T. \end{aligned} \quad (11)$$

**PROOF.** Similar to the proof for Theorem 3.  $\square$

**LEMMA 3A.** *While maximizing the performance of Task A, with  $F_1 > 0$  and  $F_2 \leq 0$ , a comparison of the generation coefficients  $F_1$  and  $F_4$  determines whether Task A should overlap with Task B.*

**PROOF.** Assume that  $F_1 > 0$  and  $F_3 \leq 0$ . This means that Task A prefers to work independently and Task B prefers to work concurrently. Therefore, to decide the choice of region for accumulating the performance of A, a comparison between the performance gain while working independently and the performance gain while working concurrently is necessary. Table 3 describes the performance contribution of Tasks A&B in Regions 1 and 2.

$F_1 = [\alpha_{A1} - \phi\alpha_{A2} + (1 - \phi)R_B\alpha_{B2}]$  and  $F_4 = [R_A\alpha_{A1} + (1 - \phi)\alpha_{B2} - \phi R_A\alpha_{A2}]$  describe the performance gain of Task A (working independently) and Task B (working concurrently), respectively.

Therefore, if  $F_1 - F_4 > 0$ , then Task A in the independent mode contributes more to the overall project performance than Task B in the concurrent mode.  $\square$

**LEMMA 3B.** *While maximizing the performance of Task B, with  $F_1 > 0$  and  $F_3 \leq 0$ , a comparison of the generation coefficients  $F_2$  and  $F_3$  determines whether Task A should overlap with Task B.*

**PROOF.** Follows an argument similar to Lemma 3a.  $\square$

**THEOREM 5.** *The tree shown in Figure 2 provides an exhaustive mapping of conditions for optimal solution based on all possible values of the transformed input parameters:  $F_1, F_2, F_3,$  and  $F_4$ .*

**PROOF.** If  $F_1 > 0$ , then Task A creates more performance when working independently than when it is working concurrently with Task B. In addition, if  $F_3 > 0$ , then Task B also produces more performance when working independently than when it is working concurrently with Task A. Thus, when  $F_1 > 0$  and  $F_3 > 0$ , a “sequential” strategy depicted by the extreme left branches of the tree in Figure 2 is optimal.

The scenario  $F_1 > 0$  and  $F_3 \leq 0$ , refers to instances when Task A produces more performance when working independently, and Task B produces more performance when working concurrently

with A. We invoke Lemma 3 to point out that:

(1) When  $F_2 \leq F_3$ , Regions 1, 2, and 3 are all required to maximize performance. Hence, an overlapping strategy is optimal.

(2) When  $F_1 > F_4$  and  $F_2 > F_3$ , Region 2 will generate more performance than either Region 1 or Region 2, and hence a concurrent strategy is optimal.

(3) When  $F_1 \leq F_4$  and  $F_2 > F_3$ , Region 2 will generate less performance than either Region 1 or Region 2, a sequential strategy is optimal.

Existence of optimal choices in Figure 2, when  $F_3 > 0$ , follows similar arguments.  $\square$

**COROLLARY 5.1.** *Policies outside the Region ABCODA (in Figure 3) are either infeasible or suboptimal performance.*

**PROOF.** The tree shown in Figure 2 provides an exhaustive mapping of the solutions bounded by the Space ABCODA in Figure 3. According to Theorem 1, the regions above Line AB will lead to suboptimal performance. The triangular regions CBC' and DAD' lead to infeasible solutions because the Nonnegativity Constraints (8c) and (8d) are violated in those regions.  $\square$

## References

- Ahmadi, R., R. H. Wang. 1999. Managing development risk in product design processes. *Oper. Res.* **47**(2) 235–246.
- AitSahlia, F., E. Johnson, P. Will. 1995. Is concurrent engineering always a sensible proposition? *IEEE Trans. on Eng. Management* **42**(2) 166–170.
- Brown, S., K. Eisenhardt. 1995. Product development: Past research, present findings, and future directions. *Acad. Management Rev.* **20**(2) 343–378.
- Carrascosa, M., S. D. Eppinger, D. E. Whitney. 1998. Using the design structure matrix to estimate a product development time. ASME Design Automation Conference, Atlanta, GA.
- Clark, K., T. Fujimoto. 1991. *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Harvard Business School Press, Boston, MA.
- Cohen, M. A., J. Eliashberg, T. Ho. 1996. New product development: The performance and time-to-market tradeoff. *Management Sci.* **42**(2) 173–186.
- Cooper, R. G., E. J. Kleinschmidt. 1993. Stage gate systems: A game plan for new product success. *Marketing Management* **1**(4) 20–29.
- Eppinger, S. D., D. E. Whitney, R. P. Smith, D. Gebala. 1994. A model-based method for organizing tasks in product development. *Res. Engrg. Design* **6**(1) 1–13.
- Finger, S., J. Dixon. 1989. A review of research in mechanical engineering design: Part I: Descriptive, prescriptive, and computer-based models of design processes. *Res. Engrg. Design* **1**(1) 51–67.
- Ford, D. N., J. D. Sterman. 1998. Expert knowledge elicitation to improve formal and mental models. *System Dynam. Rev.* **14**(4) 309–340.

- Griffin, A. 1996. The impact of engineering design tools on new product development efficiency and effectiveness. *Proc. Third EIASM Internat. Product Development Conf.* Fontainebleau, France, 363–380.
- Ha, A. Y., E. L. Porteus. 1995. Optimal timing of reviews in concurrent design for manufacturability. *Management Sci.* **41**(9) 1431–1447.
- Hoedemaker, G., J. Blackburn, L. Van Wassenhove. 1999. Limits to concurrency, *Decision Sci.* **30**(1) 1–18.
- Khurana, A., S. R. Rosenthal. 1997. Integrating the fuzzy front end of new product development. *Sloan Management Rev.* **38**(2) 103–120.
- Krishnan, V., S. D. Eppinger, D. E. Whitney. 1997. Model-based framework to overlap product development activities. *Management Sci.* **43**(4) 437–451.
- , V., K. Ulrich. 2001. Product development decisions: A review of the literature. *Management Sci.* **47**(1) 1–21.
- Lawson, M., H. Karandikar. 1994. A survey of concurrent engineering. *Concurrent Engrg. Res. Appl.* **2** 1–6.
- Lewis, K., F. Mistree, 1998. Collaborative, sequential, and isolated decisions in design. *ASME J. Mechanical Design* **120** (Dec) 643–652.
- Loch, C., C. Terwiesch. 1998. Communication and uncertainty in concurrent engineering. *Management Sci.* **44**(8) 1032–1048.
- Repenning, N. P. 1999. A dynamic model of resource allocation in multi-project research and development systems. Working paper, MIT Sloan School, Cambridge, MA.
- Smith, R. P., S. D. Eppinger. 1997a. Identifying controlling features of engineering design iteration. *Management Sci.* **43**(3) 276–293.
- , ———. 1997b. A predictive model of sequential iteration in engineering design. *Management Sci.* **43**(8) 1104–1120.
- , J. Morrow. 1999. Product development process modeling, *Design Studies* **20** 237–261.
- Terwiesch, C., C. H. Loch. 1999. Measuring the effectiveness of overlapping development activities. *Management Sci.* **45**(4) 455–465.
- Varian, H. R. 1992. *Microeconomic Analysis*. W. W. Norton & Company, New York.
- Wheelwright, S. C., K. B. Clark. 1992. *Revolutionizing Product Development*. Free Press, New York.
- Wind, J., V. Mahajan. 1997. Issues and opportunities in new product development: An introduction to the special issue. *J. Marketing Res.* **34**(Feb) 1–12.
- Yassine, A., K. Chelst, D. Falkenburg. 1999. A decision analytic framework for evaluating concurrent engineering. *IEEE Trans. Engrg. Management* **46**(2) 144–157.

*Accepted by Wallace J. Hopp; received June 21, 2000. This paper was with the authors 2 months for 1 revision.*