Performance of Coupled Product Development Activities with a Deadline

Nitindra R. Joglekar, Ali A. Yassine, Steven D. Eppinger and Daniel E. Whitney
Boston University, Boston, Massachusetts, 02215
Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139

Performance of Coupled Product Development Activities with a Deadline

Nitindra R. Joglekar λ Ali A. Yassine λ Steven D. Eppinger λ Daniel E. Whitney
Boston University, Boston, Massachusetts, 02215
Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139

Abstract

This paper explores the performance of coupled development tasks subject to a deadline constraint 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 tasks that share 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, Performance Generation, Design Process Modeling, Concurrent/ Sequential/ Overlapping Development, Component / System Performance, Software Engineering)

1. Introduction

Product development (PD) is the process of transforming customer needs into an economically viable product that satisfies these needs. PD research spans many different disciplines ranging from organizational science (Brown and Eisenhardt 1995), marketing (Wind and Mahajan 1997), 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,b)

can result in increased development lead time and cost (AitSahlia et al. 1995). Consequently, a growing body of CE management models is built to provide insights into the management of information, communication, 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 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 clockspeed 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. Then, the core tradeoff 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.

The 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 date of a new automobile model (2001 Ford Explorer), the announced software release (MS Office 2000) or an intermediate milestone in a stage-gate review process (Cooper 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 the 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 for the coupled development tasks 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 taxonomy for CE decision problems. Then, we discuss major management science CE models and contrast them against our proposed model. In sections 3 and 4, we introduce the PGM assumptions and formulation, and derive theorems governing the model behavior. In section 5, we characterize the optimal policies for product development management in a deadline environment. In section 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 section 7. Finally, section 8 presents our conclusion and sets the stage for future extensions to the base PGM.

2. Related Literature

In this section, we propose 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 Fujimoto1991). Development activities are classified into three types (Eppinger et al. 1994): dependent, interdependent, 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 dead line. 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. parallel 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 (parallel, sequential, and overlapping) and provides the optimal execution configuration.

Krishnan et al. (1997) construct a model for overlapping nominally sequential activities in order 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 which maximizes product performance.

Ahmadi and Wang (1999) develop a model that optimally places design reviews along the development process in order to minimize development risk. In addition, the model provides optimal resource allocation policies for each design stage. The PGM is similar to Ahmadi et al. in

the way they set the development speed in each stage in order 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 tradeoff 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 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 independent 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[1] as shown in Figure 1. 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 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 ($0 < R_A, R_B < 1$). 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.[2]

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

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

Where:

A and B denote the tasks

j denotes the region number/index, j = 1,2,3. Regions are described as follows:

- j = 1 means that task A is working only
- j = 2 means that both tasks A and B are working concurrently
- j = 3 means that task B is working only

R_A ($0 \le R_A < 1$) is the penalty of task A on task B.

R_B ($0 \le R_B < 1$) is the penalty of task B on task A.

L is the maximum amount of available labor resources for the development project[3]. The unit for L is dollars ($)/time.

$X_{Aj}(t)$ 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 improvement for tasks A and B respectively. That is, $\dot{X}_{Aj} = \frac{dX_{Aj}}{dt}$ and $\dot{X}_{Bj} = \frac{dX_{Bj}}{dt}$.

ϕ is the fraction of L used for working on task A[4].

for j =1, $\phi = 1$

for j = 2, $0 < \phi < 1$

for j =3, $\phi = 0$

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

Notice that since $\phi = 1$ in region 1 and $\phi = 0$ in region 3, then the performance contribution from α_{B1} and α_{A3} is irrelevant to the solution of the problem. Hence, there are only 7 input parameters; namely, α_{A1}, α_{A2}, α_{B2}, α_{B3}, ϕ, R_A, and R_B. The following table summarizes the performance generation contribution of upstream and downstream tasks at system level.

The overall product performance for a given region j is:

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

Where $(t_j)_{initial}$ and $(t_j)_{final}$ 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:

$$X_{Aj}(t) = L[f a_{Aj} - (1-f) R_B a_{Bj}]t + c_{Aj} \tag{4}$$

$$X_{Bj}(t) = L[(1-f) a_{Bj} - f R_A a_{Aj}]t + c_{Bj} \tag{5}$$

The constants c_{ij} are determined by the boundary conditions of the three different regions (j=1,2,3)[5]. This yields the following expressions for $X_{A3}(T)$ and $X_{B3}(T)$, the performance of tasks A and B by time T:

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

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

The objective function Max.$\{X_{A3}(T) + X_{B3}(T)\}$ becomes:

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

s.t.

$$X_{A1}(0) = 0 \tag{8a}$$

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

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

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

$$S_A+S_B \leq T \tag{8e}$$

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 non-negativity 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[6]. 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 exploit the properties of the 7-tuple input parameters (α_{A1}, α_{A2}, α_{B2}, α_{B3}, ϕ, 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, as shown in the legend of Figure 3.

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 non-decreasing function in time.

Theorem 1: When one schedules two coupled tasks with respect to a non-decreasing performance measure, it is not necessary to consider schedules which involve idle time.

Lemma 2[7]: (the See-Saw rule): Given a pair of adjusted performance generation rates $[\phi\alpha_{Aj}(1-R_A)]$ and $[(1-\phi)\alpha_{Bj}(1-R_B)]$, it is always optimal to perform work on the task with the largest adjusted performance generation rate, if constraint (8c and 8d) are not violated.

Theorem 2: If the sequential strategy is optimal, the corresponding solution for S^*_A is bounded by:

$$\frac{a_{B3}}{\frac{a_{A1}}{R_B} + a_{B3}} T \le S^*_A \le \frac{a_{B3}}{a_{A1}R_A + a_{B3}} T \qquad (9)$$

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: $$\frac{R_A f a_{A2} - (1-f) a_{B2}}{a_{B3} + R_A f a_{A2} - (1-f) a_{B2}} T \le S^*_B \le \frac{f a_{A2} - (1-f) R_B a_{B2}}{R_B a_{B3} + f a_{A2} - (1-f) R_B a_{B2}} T \quad (10)$$

Theorem 4: If the overlap strategy is optimal and $S^*_B = 0$, then the corresponding solution for S^*_A is bounded by: $$\frac{(1-f) R_B a_{B2} - f a_{A2}}{a_{A1} + (1-f) R_B a_{B2} - f a_{A2}} T \le S^*_A \le \frac{(1-f) a_{B2} - R_A f a_{A2}}{a_{A1} R_A + (1-f) a_{B2} - R_A f a_{A2}} T \quad (11)$$

In order 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:

$$F_1 S_A - F_2 S_B + B_1 T \ge 0 \quad (12)$$
$$F_3 S_B - F_4 S_A + B_2 T \ge 0 \quad (13)$$

Where: $F_1 = \alpha_{A1} - \phi\alpha_{A2} + (1-\phi)R_B\alpha_{B2}$
$F_2 = \phi\alpha_{A2} - (1-\phi)R_B\alpha_{B2} + R_B\alpha_{B3}$
$F_3 = \alpha_{B3} - (1-\phi)\alpha_{B2} + \phi R_A\alpha_{A2}$
$F_4 = R_A\alpha_{A1} + (1-\phi)\alpha_{B2} - \phi R_A\alpha_{A2}$
$B_1 = \phi\alpha_{A2} - (1-\phi)R_B\alpha_{B2}$
$B_2 = (1-\phi)\alpha_{B2} - \phi R_A\alpha_{A2}$

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) which account for interactions between the original input parameters[8]. In the rest of the paper we will exclusively deal with these transformed parameters in order to characterize the optimal solution space. The transformed parameters are called performance generation coefficients and are interpreted in Section 5.4.

Lemma 3a: While maximizing the performance of task A, with $F_1 > 0$ and $F_3 \le 0$, 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 $F_3 \le 0$, a comparison of the generation coefficients F_2 and F_3 determines whether task A should overlap with task B.

Theorem 5: Tree shown in Figures 2 provide 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.

Corollary 5.1: Policies outside the region ABCODA (in Figure 4) are either infeasible or result in sub-optimal 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 concurrency in the context of our model. Finally, we provide an intuitive interpretation for the generation coefficients and the role they play in determining the optimal solution.

5.1 Characterization of the Optimal Decision Space

We start by mapping the solutions for every possible value of the 7-tuple inputs into a decision space for optimal S_A and S_B as shown in Figure 3. The perimeter of polygon ABCODA provides a graphical representation for the set of all optimal solutions as a function of the decision variables S_A and S_B. The area above line AB represents a space where the schedules involve idle time. Theorem 1 shows that the optimal solution need not consider schedules with idle time. Consequently, optimal solutions lie on or below line AB. Furthermore, line AB represents the locus of points where the choice of S_A and S_B result in a sequential strategy according to Theorem 5. Coordinates for points A and B are derived in Theorem 2 and its corollaries. Theorem 3 and its corollaries provide the coordinates for Point C. Existence of a concurrent strategy, as represented by point O, is provided by Theorem 5. The condition for existence of optimal overlapping policy, as shown along line CO is given by Theorem 5. Similarly, Theorem 5 together with Theorem 4 shows the existence of point D as another optimal overlapping strategy. Finally, the tree analysis given in Figure 2 confirms that no optimal solution exists in the interior of the region ABCODA. Using the See-Saw Lemma, the optimal solution will lie at one of the corner points of the polygon ABCODA

as long as the adjusted performance generation coefficients [$\phi\alpha_{Aj}(1-R_A)$ and $(1-\phi)\,\alpha_{Bj}(1-R_B)$] are not equal. If these coefficients are equal, and the sequential strategy is optimal, then any policy that lies along the edge AB will lead to maximum performance. Similarly, if they are equal and partial overlapping is optimal, then any policy that lies along the edge COD will lead to maximum performance.

Starting at point A and moving around ABCODA clockwise, the reader may follow the duration of task A (represented by the shaded rectangle in the icon next to each point). Notice that task A's duration increases as the solution moves towards point B along line AB, and consequently the duration of task B is reduced by an equal amount. If the solution moves beyond point B (i.e. towards point C') to increase task A's duration further, then this violate the non-negativity constraint (Equation 9d). Therefore, no feasible solutions fall on line BC'. In order 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 towards 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 (ϕ) during overlap of tasks A and B by further exploring the delta wing shaped polygon (ABCODA), as shown in Figure 3. Note that ϕ 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 ϕ collapses the delta wing by shifting the location of lines BC and AD towards the origin O while changing their original slopes as they move inward[9]. Finally, the leading edge of the delta wing

(DOC) represents the line of constant product performance derived by changing ϕ, 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[10] that drive the solution within the optimal decision space as shown in Figure 4. Notice that a positive force in the direction A to B moves the optimal solution towards 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 towards sequential strategy, and a negative force moves the optimal solution towards 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 towards 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-tupple 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.

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 down stream component 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[11]. 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[12]. 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 a 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 of 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®. 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 team tasks and their contribution to system performance. These data are the basis for the subject's response to the assessment questionnaire.

For instance, in a segment of the project that deals with the integration of the web front-end with the legacy back end, teams establish relative shapes for the generation functions as depicted by Figure 5. 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. 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. 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).

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 team 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 d) captures the choices available to system architect: SU1 (Upstream is fast) or SU2 (Upstream is slow) and SD1 (Downstream is fast) or SD2 (Downstream is slow). In the case, data indicated that the A1-B1 scenario governed the development process.

The example shows that an assessment of the performance generation rates is possible for the purpose of applying the PGM. Field observations at Softex confirm that measurement of performance generation coefficients, especially early in the development process, are imprecise. 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 the structure of the process is managerially more relevant 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

In order 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 in parallel 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 in parallel 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 precisely the same rationale described in case 3.

Drawing on the stylistic assessments from the previous section to illustrate the use of the framework, recall that the two teams from Softex picked scenario 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 upon to review the interaction diagrams and decide on the overall sequence based on the system performance requirements.

8. Conclusion

The tradeoff 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 dead lines, 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; Carrascosa 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 two divisions of a firms that are 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 ϕ, the resource allocation fraction, will be an explicit decision variable. Then, resource constraints in the PGM can be re-interpreted 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 counter-intuitive result lies in the tradeoff 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 towards full concurrent engineering. On the other hand, we show what forces prevent the system to drift towards 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.[13]

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 3 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 non-decreasing.

Theorem 1: When one schedules two coupled tasks with respect to a non-decreasing performance measure, it is not necessary to consider schedules which 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 ($0 \le t_1 < t_2 \le T$). Call this schedule S'. S' can take 3 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 $0 \le t \le t_1$. This schedule can be stretched by multiplying each segment of S' by factor $\mathbf{(T-t_1)/t_1}$. This new schedule will result in 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.

Lemma 2: (the See-Saw rule): Given a pair of adjusted performance generation rate $[\phi\alpha_{Aj}(1-R_A)]$ and $[(1-\phi)\alpha_{Bj}(1-R_B)]$, it is always optimal to perform work on the task with the largest adjusted performance generation rate, if constraint (8c and 8d) are not violated.

Proof: Since $\phi\alpha_{Aj}(1-R_A) > 0$ and $(1-\phi)\alpha_{Bj}(1-R_B) > 0$ 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 efficient for every j (j = 1, 2, 3).

Theorem 2: If the sequential strategy is optimal, the corresponding solution for S^*_A is bounded by:

$$\frac{a_{B3}}{\frac{a_{A1}}{R_B} + a_{B3}} T \le S^*_A \le \frac{a_{B3}}{a_{A1}R_A + a_{B3}} T \qquad (9)$$

Proof: Lower bound: $X_{A3}(T) \ge 0 \Rightarrow \alpha_{A1} S_A \ge R_B \alpha_{B3} (T - S_A)$

Upper bound: $X_{B3}(T) \ge 0 \Rightarrow R_A \alpha_{A1} S_A \le \alpha_{B3} (T - S_A)$

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.

Theorem 3: If the overlap strategy is optimal and $S^*_A = 0$, then the corresponding solution for S^*_B is

bounded by: $$\frac{R_A f a_{A2} - (1-f) a_{B2}}{a_{B3} + R_A f a_{A2} - (1-f) a_{B2}} T \le S^*_B \le \frac{f a_{A2} - (1-f) R_B a_{B2}}{R_B a_{B3} + f a_{A2} - (1-f) R_B a_{B2}} T \quad (10)$$

Proof: Lower bound: $X_{A3}(T) \ge 0 \Rightarrow \{\phi\alpha_{A2} - (1-\phi) R_B \alpha_{B2}\}(T-S_B) \ge R_B \alpha_{B3} S_B$

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

Theorem 4: If the overlap strategy is optimal and $S^*_B = 0$, then the corresponding solution for S^*_A is

bounded by: $$\frac{(1-f) R_B a_{B2} - f a_{A2}}{a_{A1} + (1-f) R_B a_{B2} - f a_{A2}} T \le S^*_A \le \frac{(1-f) a_{B2} - R_A f a_{A2}}{a_{A1} R_A + (1-f) a_{B2} - R_A f a_{A2}} T \quad (11)$$

Proof: Similar to the proof for Theorem 3.

Lemma 3a: While maximizing the performance of task A, with $F_1 > 0$ and $F_3 \le 0$, 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 ($F_3 \le 0$). This means that task A prefers to work independently and task B prefers to work concurrently. Therefore, in order 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.

Region (j)	**1**	**2**
Perf. (task A)	α_{A1}	$[\phi\alpha_{A2}-(1-\phi)\alpha_{B2}R_B]$
Perf. (task B)	$-\alpha_{A1} R_A$	$[(1-\phi)\alpha_{B2}-\phi\alpha_{A2}R_A]$

Table 3: Performance Contribution of Tasks A and 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.

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.

Theorem 5: Tree shown in Figures 2 provide 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 when working independently creates more performance 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 as to 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 where 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:

(i) When F2 ≤ F3, regions 1, 2, and 3 are all required to maximize performance. Hence an overlapping strategy is optimal.

(ii) When F1 > F4 and F2 > F3, region 2 will generate more performance than either region 1 or region 2, and hence a concurrent strategy is optimal.

(iii) When F1 ≤ F4 and F2 > F3, 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 F3 > 0, follows similar arguments.

Corollary 5.1: Policies outside the region ABCODA (in Figure 3) are either infeasible or sub-optimal 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 sub-optimal performance. The Triangular regions CBC' and DAD' lead to infeasible solutions, because the non-negativity constraints (8c) and (8d) are violated in those regions.

References

Ahmadi, R., R. H. Wang. 1999. Managing Development Risk in Product Design Processes. *Operations Research* **47**(2) 235-246.

AitSahlia, F., E. Johnson, P. Will. 1995. Is Concurrent Engineering Always a Sensible Proposition? *IEEE Trans. on Engineering Management* **42**(2) 166-170.

Brown, S., K. Eisenhardt. 1995. Product Development: Past Research, Present Findings, and Future Directions. *Academy of Management Review* **20**(2) 343-378.

Carrascosa, M., S.D. Eppinger, D.E. Whitney. 1998. Using the Design Structure Matrix to Estimate Time to Market in a Product Development Process. *ASME Design Automation Conference, Atlanta* 98-6013.

Clark, K., T. Fujimoto. 1991. *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Harvard Business School Press, Boston.

Cohen, M. A., J. Eliashberg, T. Ho. 1996. New Product Development: The Performance and Time-to-Market Tradeoff. Management Science **42**(2), 173-186.

Eppinger, S. D., D.E. Whitney, R.P. Smith, D. Gebala. 1994. A Model-Based Method for Organizing Tasks in Product Development. *Res. Eng. Design* **6**(1), 1-13.

Ford, D. N., J. D. Sterman. 1998. Expert Knowledge Elicitation to Improve Formal and Mental Models. *System Dynamics Review* **14** (4) 309-340.

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. Eng. Design* **1**(1) 51-67.

Griffin, A. 1996. The Impact of Engineering Design tools on new Product Development Efficiency and Effectiveness. *Proceedings of the Third EIASM International Product Development Conference* Fontainebleau, France, 363-380.

Ha, A.Y., E. L. Porteus. 1995. Optimal Timing of Reviews in Concurrent Design for Manufacturability. *Management Science* **41**(9) 1431-1447.

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

Khurana, A., S. R. Rosenthal. 1997. Integrating the Fuzzy Front End of New Product Development. *Sloan Management Review* **38**(2).

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

Krishnan, V., S.D. Eppinger, D.E. Whitney. 1997. Model-Based Framework to Overlap Product Development Activities, *Management Science* **43**(4), 437-451.

Lawson, M., H. Karandikar. 1994. A Survey of Concurrent Engineering. Concurrent Engineering: Research and Applications **2** 1-6.

Lewis, K., F. Mistree, 1998. Collaborative, Sequential, and Isolated Decisions in Design. ASME Journal of Mechanical Design, 120 (Dec).

Loch, C., C. Terwiesch. 1998. Communication and Uncertainty in Concurrent Engineering. Management Science **44**(8) 1032-1048.

Repenning, N.P. 1999. A Dynamic Model of Resource Allocation in Multi-Project Research and Development Systems. MIT Sloan School working paper.

Smith, R.P., S.D. Eppinger. 1997a. Identifying Controlling Features of Engineering Design Iteration. *Management Science* **43**(3) 276-293.

Smith, R.P., S.D. Eppinger. 1997b. A Predictive Model of Sequential Iteration in Engineering Design. *Management Science* **43**(8) 1104-1120.

Smith, R.P., J. Morrow. 1999. Product Development Process Modeling," *Des. Studies* **20** 237-261.

Terwiesch, C., C. H. Loch. 1999. Measuring the Effectiveness of Overlapping Development Activities. *Management Science* **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. *Journal of Marketing Research* **34** (Feb) 1-12.

Yassine, A., K. Chelst, D. Falkenburg. 1999. A Decision Analytic Framework for Evaluating Concurrent Engineering," IEEE Transactions on Engineering Management **46** (2) 144-157.

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

Legend

Minimize Lead Time

Maximize Performance

* Irrelevant, unless resource constraints are considered

Table 1: Taxonomy of PD execution related decision problems

Region (j)	1	2	3
Bounds on time t	$0 \le t < S_A$	$S_A \le t < (T-S_B)$	$(T-S_B) \le t \le T$
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	$\alpha_{A1}(1-R_A)\, t$	$[\phi\alpha_{A2}(1-R_A)+ \alpha_{B2}(1-\phi\ )(1-R_B)]t$	$\alpha_{B3}\,(1-R_B)\, t$

Table 2: Summary of performance generation functions by region

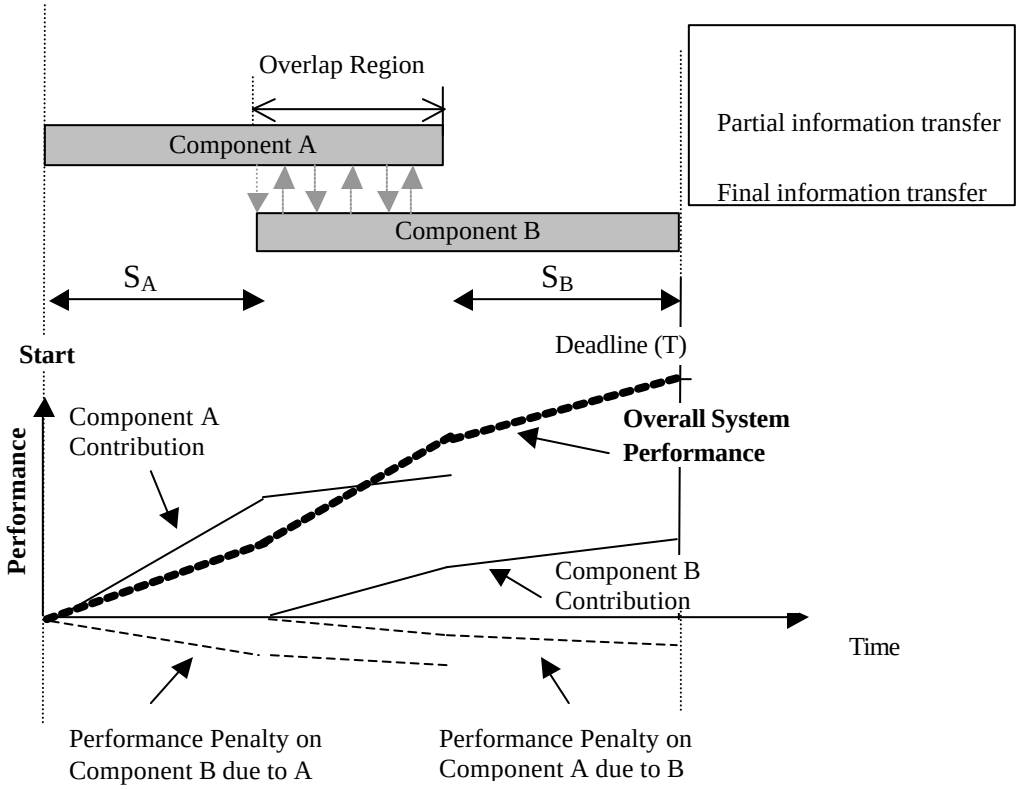


Figure 1: PGM showing component and system performance generation

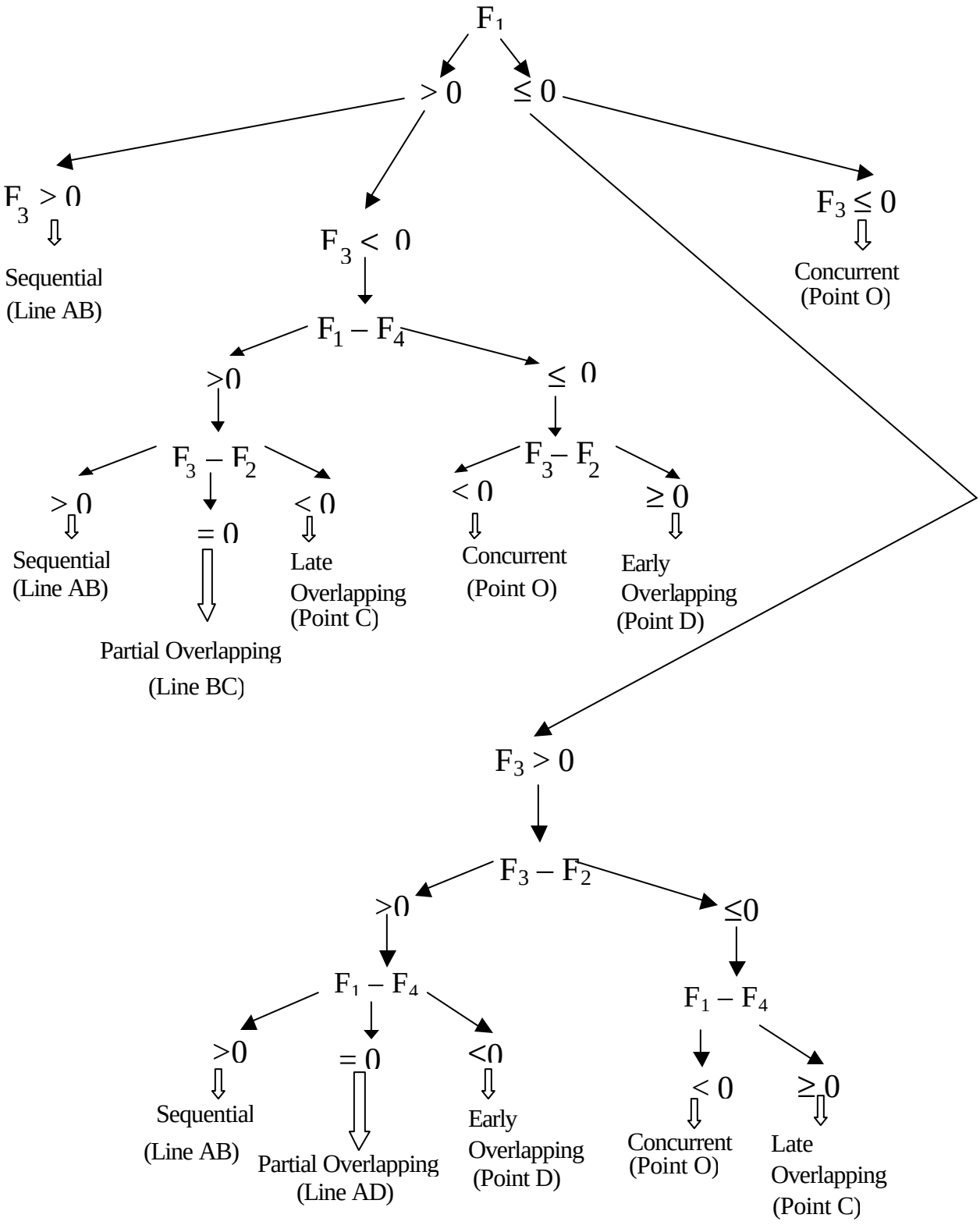


Figure 2: Tree characterizing full solution space

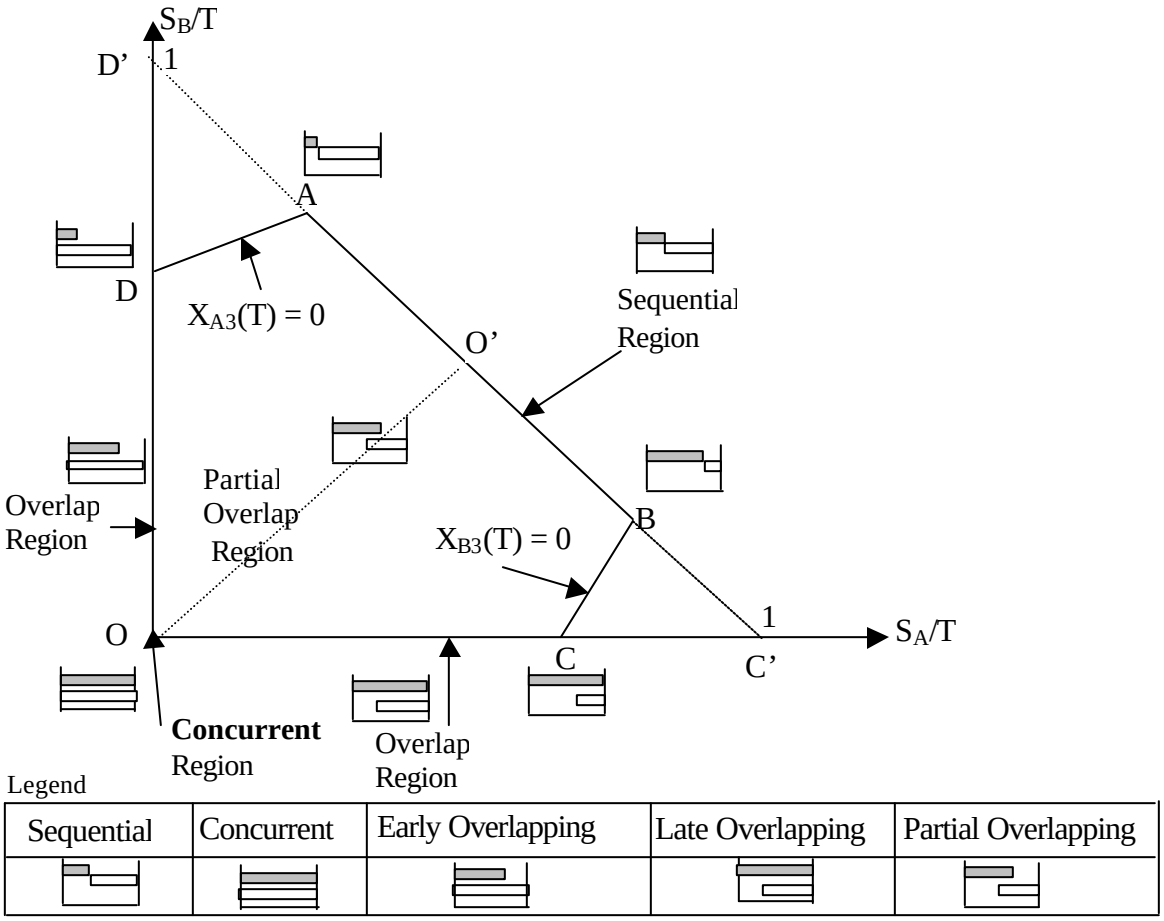


Figure 3: Optimal policy interpretations

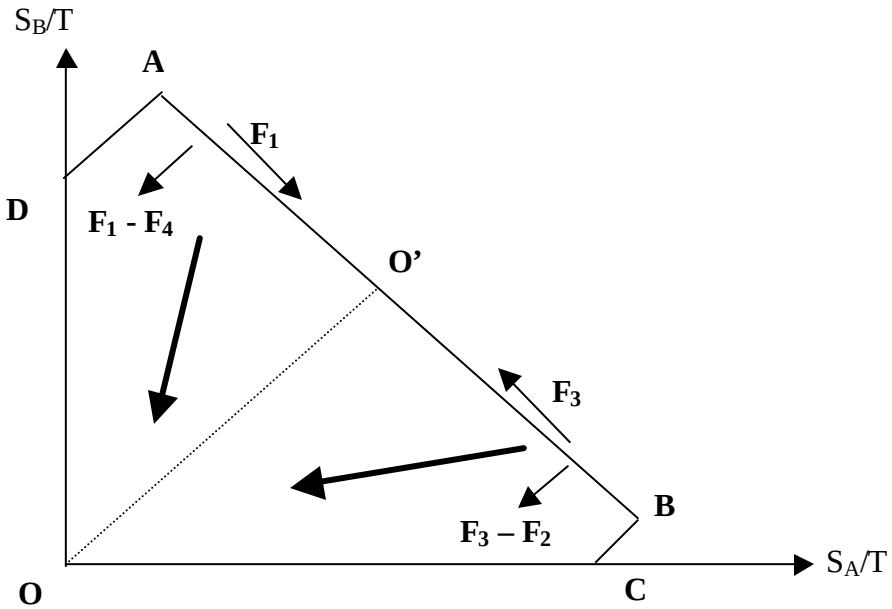
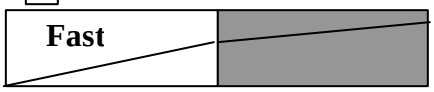


Figure 4: The dynamics of concurrent engineering
(Bold arrows show the summation of these forces and thus the direction of the solution movement)

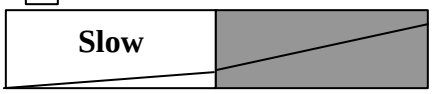
(a) Upstream (Team A) View of Component Performance Accrual (Select One):

☐ Joint work slows our progress



(A1) A Works Alone Both A&B Work

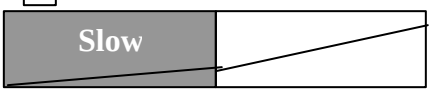
☐ Joint work speeds our progress



(A2) A Works Alone Both A&B Work

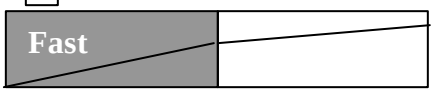
(b) Downstream (Team B) View of Component Performance Accrual (Select One):

☐ Joint work slows our progress



(B1) Both A&B Work B Works Alone

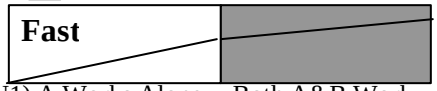
☐ Joint work speeds up progress



(B2) Both A&B Work B Works Alone

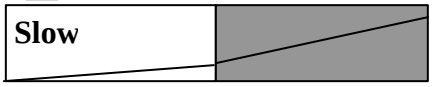
(c) Architect's View of Upstream System Performance accrual* (Select one):

☐ Joint work slows progress



(SU1) A Works Alone Both A&B Work

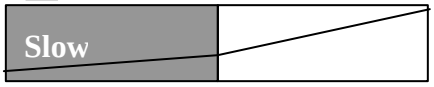
☐ Joint work speeds progress



(SU2) A Works Alone Both A&B Work

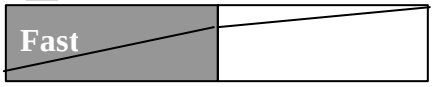
(d) Architect's View of Downstream System Performance Accrual* (Select one):

☐ Joint work slows our progress



(SD1) Both A&B Work B Works Alone

☐ Joint work speeds up progress



(SD2) Both A&B Work B Works Alone

Figure 5: Assessment of performance generation

(In each box, the x-axis depicts elapsed time and y-axis is performance)

*Architect's View required only if the teams have picked either A1–B1 or A2-B2

A / B		**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 (≥0)
				Downstream ($F_3 - F_2$) Fast (≤ 0)	**C** / **LO**
				Downstream ($F_3 - F_2$) Slow (>0)	**EO** / **S**
	Slow ($F_3 > 0$)	**System Perform. Generation**	**Upstream ($F_1 - F_4$)**	**S**	
			Slow (<0) / Fast (≥0)		
		Downstream ($F_3 - F_2$) Fast (≤ 0)	**C** / **LO**		
		Downstream ($F_3 - F_2$) Slow (>0)	**EO** / **S**		

Legend

C = Concurrent

S = Sequential

EO = Early Overlap

LO = Late Overlap

Figure 6: Optimal strategies based on the generation coefficients

End Notes

[1] We define the upstream task to be task A and the downstream task to be task B, without loss of generality.

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

[3] 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.

[4] We will assume that ϕ 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.

[5] $X_{A1}(0) = 0$, $X_{B1}(0) = 0$

$X_{A1}(S_A) = X_{A2}(0)$, $X_{B1}(S_A) = X_{B2}(0)$

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

[6] The model will work for any threshold. Zero was selected as an arbitrary value.

[7] Lemma 2 is equivalent to taking the partial derivative of Equation (8) with respect to S_A and S_B.

[8] 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.

[9] It is worth noting that a similar shift would also happen if constraints (8c) and (8d) require a non-zero 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.

[10] The evolution coefficients can be thought as forces for studying the dynamics of concurrency. Caveat: these coefficients are not analogous to any physical force.

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

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

[13] The authors thank Thomas Roemer, Durward Sobek 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 question about their PD process.