A Simulation-Based Process Model for Managing Complex Design Projects

Soo-Haeng Cho and Steven D. Eppinger, *Member, IEEE*

***Abstract*—This paper presents a process modeling and analysis technique for managing complex design projects using advanced simulation. The model computes the probability distribution of lead time in a stochastic, resource-constrained project network where iterations take place among sequential, parallel, and overlapped tasks. The model uses the design structure matrix representation to capture the information flows between tasks. We use a simulation-based analysis to account for many realistic aspects of design process behavior which were not possible in previous analytical models. We propose a heuristic for the stochastic, resource-constrained project scheduling problem in an iterative project network. The model can be used for better project planning and control by identifying leverage points for process improvements, and for evaluating alternative planning and execution strategies. An industrial example is provided to illustrate the utility of the model.**

***Index Terms*—Design iteration, design structure matrix, process modeling, project management, project simulation.**

I. Introduction

TODAY'S competitive industrial market has created a highly challenging environment for product development. Companies are under increasing pressure to create and sustain competitive advantage by reducing product development time and cost, while maintaining a high level of quality. These needs drive companies to focus more than ever before on streamlining their product development process [17], [53], [56], [60].

A complex design project usually involves a large number of tasks executed by a network of professionals from various disciplines. As complexity increases, it becomes more difficult to manage the interactions among tasks and people; it may be impossible to even predict the impact of a single design decision throughout the development process [25], [55]. Due to the tremendous complexity of many engineering projects, project management techniques have played a vital role in the success of such projects.

Since the introduction of network-based project scheduling techniques such as the critical path method (CPM) [30] and program evaluation and review technique (PERT) [37], many researchers have developed extensions to add new power to these classical methods. Some of the pioneering work includes the use of Monte Carlo sampling [58] to account for stochastic task duration, the graphical evaluation and review technique (GERT) [40] that allows probabilistic routing and feedback loops, the generalized precedence relations (GPRs) [23], and various exact and heuristic techniques for the resource-constrained project scheduling problem (RCPSP). The classical RCPSP is the problem of scheduling tasks such that precedence and resource constraints are obeyed and project lead time is minimized. Given the NP-hardness of the problem [9], various exact and heuristic approaches have been used. More recently, researchers have developed techniques for the RCPSP with random task durations, so-called the stochastic RCPSP. Extensive literature review for both the classical and stochastic RCPSPs can be found in [20] and [39].

Iteration is a fundamental characteristic of complex design projects [5], [25], [31], [61]. However, the network-based project scheduling techniques discussed above have very limited capabilities in modeling iterations. Furthermore, as Smith and Morrow [50] pointed out, it is not clear that the primary behaviors that those scheduling techniques are able to capture (precedence task relationships, resource constraints, stochastic task durations) are the behaviors that are most critical to engineering design management. For these reasons some researchers have turned to modeling frameworks other than CPM/PERT to develop models of design iteration and to better address other behaviors.

Steward [52] developed the design structure matrix (DSM) method for such purposes. The DSM provides a compact representation of a complex system by showing information dependencies in a square matrix. The DSM method is based on the earlier work in large-scale system decomposition [34], [44], [51], [59]. Eppinger *et al.* [25] extended Steward's work by explicitly modeling information coupling among tasks and investigating different strategies for managing task procedures. Some researchers have used the DSM framework to design iteration modeling to extend its information-based structuring analysis to schedule analysis. A recent survey by Browning [14] shows its increasing use in various application areas including product development, project planning, project management, systems engineering, and organization design.

Several analytical models have been developed to characterize timing of iterative design processes. These include sequential iteration models, parallel iteration models, and overlapping models. Sequential iteration models, wherein tasks are repeated one after the other by a probabilistic rule, have been implemented in several approaches. Smith and Eppinger [48] developed a model based on a reward Markov chain using the

Manuscript received February 11, 2003. Review of this manuscript was arranged by Department Editor A. Marucheck. This work was supported by the Center for Innovation in Product Development, Massachusetts Institute of Technology, Cambridge, MA, and by the Singapore-MIT Alliance.

S.-H. Cho is with the UCLA Anderson School of Management, Los Angeles, CA 90095-1481 USA (e-mail: scho@anderson.ucla.edu).

S. D. Eppinger is with the Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02142 USA (e-mail: eppinger@mit.edu).

Digital Object Identifier 10.1109/TEM.2005.850722

DSM representation for repetition probabilities and task durations. Ahmadi and Wang [3] extended the sequential iteration model by taking into account dynamic iteration probabilities and learning effects. Belhe and Kusiak [8] included probabilistic OR and exclusive OR relationships between tasks. Eppinger *et al.* [24] first used signal flow graphs to compute the probability distribution of lead time. Andersson *et al.* [7] extended the signal flow graph model to include learning effects and other nonlinearities. In parallel iteration models, multiple interdependent tasks work concurrently with frequent information exchanges. AitSahlia *et al.* [4] compared sequential and parallel iteration strategies in terms of a time-cost tradeoff. Hoedemaker *et al.* [28] discussed the limitations of parallel iteration due to increasing communication needs. Smith and Eppinger [49] developed a model of fully parallel iteration processes, built upon the DSM framework to predict slow and rapid convergence of parallel iteration. In overlapping models, tasks are overlapped to reduce total lead time. Ha and Porteus [27] analyzed a model of concurrent product and process design tasks and explored the optimal review timing to minimize total expected completion time. Krishnan *et al.* [33] developed a framework for overlapped sequential tasks. They explained appropriate overlapping strategies based on upstream information evolution and downstream iteration sensitivity. Loch and Terwiesch [36] presented an analytical model for optimal communication policy between two sequentially overlapped tasks. Roemer *et al.* [43] discussed time-cost tradeoffs in multiple overlapped tasks.

While each of the above iteration models captures some unique aspects of engineering design projects, they do not consider all of the effects handled by generalized project network scheduling techniques. The sequential iteration models do not handle resource constraints, repetition of multiple parallel tasks, and nonzero lag. The parallel iteration models analyze important aspects of concurrent engineering but use highly simplified assumptions. The two-task overlapping models provide the optimal way to reduce the time of two sequential tasks having the interface of unidirectional information transfer. However, the concept does not easily apply to multiple tasks, in particular, having multiple paths with iteration. In addition, there has not been any significant work resolving resource over-allocation issues in the overlapped and coupled project networks where tasks repeat by a probabilistic rule.

Since many of limitations of the above models are due to the limitation of analytical approaches, some researchers have resorted to simulation. Taylor and Moore [54] used the simulation package Q-GERT [41] for R&D project planning assuming no resource constraints. The GERT and Q-GERT methods assume that the sequence of tasks is fixed, whereas the sequence of tasks may sometimes be variable, as in Smith and Eppinger [48]. Adler *et al.* [2] developed a simulation-based framework using queuing principles for a multiproject, shared-resources setting. The model incorporates simple iterative effects on task duration by grouping iterative tasks together. The duration of the grouped task is computed by assuming an average number of iterations of tasks within the group, with constant iteration probabilities and no learning effects. However, this approach applies only to a simple network where repetition of multiple parallel tasks, nonzero lag, and variable task sequence are not permitted. Levitt *et al.* [35] proposed the virtual design team (VDT) framework, which builds organizational simulation models of projects. The model in this paper differs from the VDT model in that the objective of the VDT simulation is to predict organizational breakdowns in performing activities, while the goal of ours is to predict dynamic behavior of iterative processes. Browning and Eppinger [15] developed the first DSM-based simulation model that analyzes design iteration in a more generalized project network than the previous analytical models. However, their model does not account for resource constraints and is based on rather restrictive assumptions regarding task concurrency and rework, which will be further discussed here. Yassine and Browning [64] used the preference function of various attributes of task, project, and processor to determine resource priorities in the first DSM-based simulation model in limited resource settings. However, the heuristic does not provide a solution toward minimizing expected project lead time.

We present in this paper a second-generation, DSM-based simulation model that extends the previous model-based design process analysis to a project management tool applicable to a generalized project network. There are three key contributions of this paper.

1) The proposed model allows the streamlined interface between information-based DSM structuring analysis and network-based project scheduling analysis. The methods are complementary to each other in that the DSM method explores the information structure of a project while a network-based project scheduling technique provides a thorough time-based analysis.
2) We have improved the previous iteration models by incorporating many more general characteristics of complex design processes. The simulation approach accounts for the normal variance of processing time and permits the relaxation of a "Markovian" transition, meaning that iteration probabilities as well as the amount of rework are no longer constant in each iteration as in the earlier models. We also study a risk-based rework control policy that increases task concurrency during iterations. The proposed model is one of the first iteration process models applicable to a generalized project network having sequential, parallel, and overlapped tasks with limited resources.
3) We propose a heuristic for the stochastic RCPSP in an iterative project network where the techniques developed for the classical and stochastic RCPSPs do not work very well, as many of the heuristic measures are not defined for an iterative network.

In order to build such a rich process model, we employ numerical simulation methods. Simulation techniques are effective for the two analytical purposes: sampling of a task duration from the known distribution function and modeling of the dynamic progress of a project. For duration sampling, we use the simple Latin hypercube sampling (LHS) method [38] for its ease of implementation. The conditional Monte Carlo sampling techniques (e.g., [1], [16], and [46]) do not help reduce computational effort in an iterative project since tasks iterate probabilistically. We employ the parallel discrete-event simulation method for modeling the progress of a project as a dynamic system,

where system variables evolve over time. Note that modeling such non-Markovian transitions is impossible to represent as a Markov chain.

The remainder of this paper is organized as follows. Section II gives an overview of model constructs. Section III presents the discrete-event simulation scheme as a modeling framework and how rework of tasks is simulated. We also show why the heuristic techniques applied to the RCPSP are not valid in an iterative project network. Section IV is devoted to an application of the model to an industrial project example. Section V discusses applications of the model to project management and how the model can complement an existing project management tool. Section VI discusses limitations and possible extensions. The paper ends with a conclusion in Section VII.

II. Model Constructs

We follow the information-based view [5] of design projects in which a task is the information-processing unit that receives information from other tasks and transforms it into new information to be passed on to subsequent tasks. The information exchanged between tasks includes both tangible and intangible types such as parts, part dimensions, bills of material, etc. Model inputs characterize behaviors of individual tasks and interactions among the tasks from a schedule perspective. The duration of a task is used to model uncertainty and complexity within the domain of the task. Precedence and resource constraints determine the start times of tasks. Iterations are modeled to depict the patterns of workflows caused by dynamic information exchanges among the tasks.

A. Task Durations

A variety of distributions have been used to represent stochasticity of task duration. Our model chooses the triangular probability distribution to represent task durations since this distribution is simple and familiar to many project managers [62]. For each task, the model receives three estimated values for the expected duration of one-time execution—optimistic, most likely and pessimistic (as in some PERT-based analyses). These values represent the duration of a task from the start to the end of its continuous work, even though the task may later be repeated after its initial completion. Remaining duration decreases over time as the model simulates the project's progress. It has been found that estimating the 10th and 90th percentiles of the expected duration (as the optimistic and pessimistic values) is more reliable than the extremes of the PDF which are typically outside the realm of experience [29], [62].

The model uses the LHS method [38] to incorporate the uncertainty of the expected duration of each task based on the three estimated durations. After calculating the extreme values of the PDF, the LHS method divides the range between them into N strata of equal marginal probability $1/N$, where N is the number of random values for the expected duration representing the triangular PDF. Then, it randomly samples once from each stratum and sequences the sampled values randomly.

B. Precedence Constraints

From a schedule perspective, we consider two types of *information flow in a task*: 1) information flow at the beginning or at the end of the task and 2) information flow in the middle of the task. Accordingly, we define two types of *information flow between two tasks*. The first type represents the case that the task requires final output information from the upstream task to begin its work. The second type represents the case that the task uses final output information from the upstream task in the middle of its process or begins with preliminary information but also receives a final update from the upstream task.

The first type of information flow is translated to a "finish-to-start" precedence constraint between two tasks, while the second type is translated to a "finish-to-start-plus-lead" constraint.[1] With lead time, two tasks are overlapped so that a successor task starts before a predecessor task is finished. During information mapping between tasks, this constraint indicates the possibility of overlapping. The decision of overlapping must be made after preliminary scheduling analysis so that overlapping is effectively used for tasks on a critical path. The DSM is used to document these information flows and precedence constraints. The notation $DSM(i,j)$ for $i,j = 1,\ldots,n$ represents this two-type scheme in the DSM having n tasks where:

- $DSM(i,j) = 0$ when there is no information flow from task j to task i;
- $DSM(i,j) = 1$ when there is a finish-to-start type of information flow from task j to task i;
- $DSM(i,j) = 2$ when there is a finish-to-start-plus-lead type of information flow from task j to task i.

Various sequencing analyses such as partitioning and tearing [25], [52] can be performed by constructing the DSM using these inputs.

C. Resource Constraints

The model assumes that there exists a fixed, renewable resource pool throughout the entire project duration. It consists of specialized resources and/or resource groups of which constituents exhibit the same functional performance. Each task has its own resource requirement which is assumed to be constant over the entire period the task is processed. When two or more tasks are competing for limited resources in a certain period of time, i.e., resources are over-allocated, the model determines priorities by the heuristic rules explained later in the paper.

D. Iteration

Eppinger *et al.* [24] defined iteration as the repetition of tasks to improve an evolving development process. It is generally accepted that iteration improves the quality of a product in a design project while increasing development time. Managers must control the project to address this time-quality tradeoff [25]. In this paper, iteration is the rework of a task caused by the execution of other tasks. This definition excludes any repetitive work within a single task's execution (that being modeled as variance in the task's duration). We include all *planned* and *unplanned* iterations that can be modeled probabilistically. Some unplanned iterations cannot be considered because they result in structural

[1]The FS (+lead/lag) is the most conventional type of relationship used in practice, as well as in the project management tools such as MS Project and Primavera. Note that other relationships in the GPRs can be converted into this relationship in the case of fixed task durations [23].

changes to the project. For example, a major project failure or redirection would involve replanning the entire process, not simply reworking the established tasks.

The model assumes that rework of a task is generated due to the following causes (similar to [15], [24], [33], and [48]):

- receiving new information from overlapped tasks after starting to work with preliminary inputs;
- probabilistic failure to meet the established criteria;
- probabilistic change of inputs when other tasks are reworked.

In the proposed model, the first cause gives rise to overlapping iteration, and the second and the third causes give rise to sequential iteration. Parallel iteration of a limited number of tasks is simulated in this model by combining overlapping and sequential iteration.

1) Overlapping Iteration: Overlapping has been described as a "core technique for saving development time" [47]. It is generally acknowledged that overlapping tasks may save time, but is more costly than the conservative sequential approach. Suppose two nominally sequential, dependent tasks are partially overlapped. Then, the downstream task will start to work with the preliminary information from the upstream task. As the upstream task proceeds, its output information evolves to its final form and is released to the downstream task according to its communication policy. The downstream task may repeat part of its work to accommodate this new information. Such rework would be unnecessary if the downstream task started to work only with the final information from the upstream task (the nonoverlapped case).

In our model, we incorporate the results of previous two-task overlapping models by Krishnan and Eppinger [33] and by Loch and Terwiesch [36]. Thus, it is assumed that the desired overlap amount and the corresponding downstream rework for the overlapped tasks can be estimated in the planning stage. For the tasks i and j of which $DSM(i,j) = 2$, the notation $OA(i,j)$ is used for maximum overlap amount and $OI(i,j)$ for overlap impact for $i,j = 1,\ldots,n$. The former represents the planned overlap amount between tasks i and j, and it is a fraction of the expected duration of task i, d_i. This carries the assumption that the downstream task cannot be completed before the upstream task finishes. The latter represents the expected amount of rework in task i when task i is overlapped with task j by the amount $OA(i,j) \times d_i$ and it is a fraction of that amount. $OI(i,j) = 1$ implies no benefit from overlapping. To implement an overlapping strategy, $OI(i,j)$ should be somewhat less than 1, considering the additional cost due to the rework, as well as the risk due to the evolution of volatile preliminary information.

2) Sequential Iteration: The model takes an approach similar to Browning and Eppinger [15] by explaining sequential iteration using rework probability, rework impact, and the learning curve. Rework probability is a source of uncertainty in sequential iteration. $RP(i,j,r)$ represents the probability that task i does rework affected by task j in rth iteration for $i,j = 1,\ldots,n$ and $r = 1,2,\ldots$. In the case of $i < j$, it represents the feedback rework caused by the change of information from downstream task j or by the failure of downstream task j to meet the established criteria. In the case of $i > j$, it represents the feed-forward rework that downstream task i needs to do since upstream task j has generated new information after it has done its own rework. As a design process converges to a solution with iterative rework, there is less new information generated and fewer errors are discovered. Therefore, rework probability tends to decrease in each iteration.

Rework impact is a measure of the level of dependency between tasks in sequential iteration. $RI(i,j)$ represents the percentage of task i to be reworked when rework is caused by task j for $i,j = 1,\ldots,n$. Rework impact is assumed to be constant in each iteration. The learning curve measures a characteristic of a task when it repeats. $(L_{\mathrm{ori}})_i$ for $i = 1,\ldots,n$ represents the fraction of original duration when task i does the same work for a second time. The model assumes that the learning curve improves by $(L_{\mathrm{ori}})_i$ in each repetition until it reaches $(L_{\max})_i$, which is the minimum fraction of the original duration when task i does the same work repeatedly. Thus, rework amount is calculated as the original duration multiplied by the rework impact and learning curve.

III. Model Description

The model employs a parallel discrete event simulation (e.g., [42], [45], and [66]) to compute the distribution of lead time. Analytical features are included so that the model can describe the complex behavior of engineering design processes having overlapped tasks and sequential iterations. This section explains the underlying structure of the simulation-based model.

A. Modeling Framework

In the discrete event simulation, events trigger state transitions and time advances in discrete steps by the time elapsed between events. The distinctive feature of the discrete event simulation is that no components within a system need to be scanned at times between events. A parallel simulation allows multiple model components to be active and to send their outputs to other components.

The model uses different expected durations of tasks in each simulation run which are initially sampled using the LHS method. With those task durations, it simulates a series of sequential state transitions incorporating iterations in multiple paths. States are determined dynamically based on all the inputs of tasks and task interfaces explained earlier. In each state, it scans all tasks and determines a set of active tasks satisfying both precedence and resource constraints. If the amount of resources required by the tasks satisfying precedence constraints exceeds the resource capacity of the project, resource assignments are made by the heuristic priority rules. It assumes that a task begins to work as early as possible when it has all the necessary inputs from upstream tasks and all the required resources.

An event is defined as the completion of an active task instead of any information transfer. Thus, when any active task in the current state q is completed, the model makes a transition to the next state $q+1$. The duration of state q $(q = 0,1,2,\ldots)$ is defined as the minimum remaining duration of active tasks in the state. Before making a transition to the next state, the model subtracts the duration of the current state from the remaining durations of all active tasks. If all the remaining durations of

For each simulation run,

STEP1. Initialize model variables from the inputs at state 0.

STEP2. Initialize model variables in the current state q.

STEP3. Identify a set of concurrent active tasks in the current state satisfying precedence and resource constraints based on the priority rules.

STEP4. Account for overlapping iteration.

STEP5. Adjust the durations of the active tasks and the lead time.

STEP6. Generate sequential iteration rework.

STEP7. Make a transition to the next state q+1 and go to STEP2, or complete one simulation run if satisfying the termination condition.

Fig. 1. Algorithm summary.

tasks are zero (the termination condition), one simulation run is complete and the lead time is calculated as the sum of all the state durations. After N simulation runs, the probability distribution of lead time can be constructed. Fig. 1 summarizes the algorithm to compute lead time in one simulation run. A simulation run starts with initializing model variables from the model inputs at STEP1. It simulates time advance of tasks by following STEP2–STEP7 in each state until it satisfies the termination condition. Note that the state transitions of the proposed algorithm are similar to the so-called parallel scheduling scheme [13] that many rule-based heuristics for the RCPSP have employed. Our model is distinguished from the parallel scheme by modeling iterations, as explained next.

B. Overlapping Iteration Modeling

In each state, the model identifies a set of active tasks which have started to work in the current state. For each task in this set, the model simulates that its overlapped work has been performed in prior state(s) and the expected impact due to iterations has been added to the projection in the current state. The overlap amount of a task is dynamically determined by both precedence and resource constraints with other tasks in multiple paths. The model assumes that the overlapped portion of work has the lowest priority for limited resources and does not start unless resources are secured during its processing time. When the amount of overlap is different from the planned amount with any information-providing task, it computes the overlap impact by assuming this to be linear to the overlap amount. If a task is overlapped with multiple tasks, the overlap impact is between the maximum of single impacts and the sum of them depending on the amount of duplicate rework caused by those tasks. In this case, the model takes the latter as a default. Finally, the overlap amount is subtracted from the remaining duration of the active task and the overlap impact is added to it.

C. Sequential Iteration Modeling

In each state, the model identifies a set of active tasks which are supposed to be completed in the current state. It is those tasks that cause state-transition events. For each task in the set, the model determines whether it causes feedback and/or feedforward rework to other tasks. Rework decisions are simulated by comparing each rework probability with a randomly chosen number from the uniform distribution between 0 and 1. When rework occurs, the amount of rework is computed by the original duration of a task doing rework multiplied by a rework impact adjusted for learning curve effect. Then rework amounts are added to the remaining durations of those tasks that are determined to rework. Finally, rework parameters are changed for the next iteration. The model also simulates that a rework decision can be made before final output information is produced through overlapping.

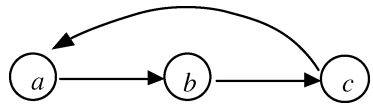


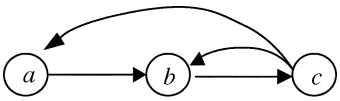
Fig. 2. Network example for sequential iteration.

Note that feedback rework in an upstream task can cause successive feedforward rework in subsequent states to the downstream tasks that have been in process or completed before. For illustration, consider the example iterative network with no overlapping shown in Fig. 2. Assume that $RP(i,j,k{+}1) = m \times RP(i,j,k)$, where $0 < m < 1$ for $k = 1,2,\ldots$ During states 1, 2, and 3, tasks a, b, and c are executed, respectively. In state 4, task a may be reworked with the probability of $RP(1,3,1)$ or the project may end with the probability of $1 - RP(1,3,1)$. In the former case, rework of task a in state 4 may cause successive feedforward rework to task b in state 5, as well as to task c in state 6. In state 7, after rework of task c, task a may be reworked with the probability of $RP(1,3,2)$ for its second iteration, and so forth. A key observation in this example is that a task in each state is dynamically determined by a probabilistic rule so that there can be an infinite number of states with variable task sequences unless feedback rework probabilities reach zero. However, the first DSM-simulation model [15] did not explain such successive feedforward iterations, ignoring the size of a feedback loop. Thus, there could be only three scenarios—abc, $abca$, and $abcab$ in the order of task execution since it does not model the possible scenario that rework of task b creates feedforward rework of task c. If this successive feedforward rework is not taken into account, the model may significantly underestimate the lead time.

D. Risk-Based Rework Control Policy

When tasks iterate sequentially, a choice of rework control policy may allow additional concurrency to accelerate rework time. For illustration, consider the simple example given in Fig. 3. In the example, there is a 20% chance that task c causes rework of both tasks a and b after its completion. In this case, do we wait for task a to be completed before starting task b even though there is only 10% chance that task a will cause feedforward rework to task b? The surveyed literature for sequential iteration models tells us to wait because task b cannot start before task a is finished due to the finish-to-start constraint. This is based on the underlying assumption that the precedence constraint between tasks a and b should also be respected when tasks iterate. However, in practice, a project manager may prefer a different policy when there is a small chance that task a will produce new information also causing task b to rework. By performing the rework of both tasks a and b concurrently, the lead time can be reduced with small additional risk.

We introduce the *rework concurrency* (RC) parameter to model this strategic decision upon task concurrency during



(a)

Task Name		1	2	3
a	1			0.5
b	2	0.1		0.4
c	3		0.5	

(b)

Fig. 3. Network example for task concurrency. (a) Task(or activity)-on-node network. (b) $RP(i, j, 1)$ $(i, j = 1, 2, 3)$.

	1	2	3	4	5	6
1						
2	0.5		0.4			
3		0.5				
4		0.5			0.1	
5			0.1	0.4		
6					0.4	

(a)

	1	2	3	4	5	6
1						
2	0.50					
3	0.25	0.50				
4	0.25	0.50	0.00			
5	0.13	0.25	0.10	0.40		
6	0.05	0.10	0.04	0.16	0.40	

(b)

Fig. 4. Example of computing RC. (a) $RP(i, j, 1)$ $(i, j = 1, \ldots, 6)$. (b) $RC(i, j)$ $(i, j = 1, \ldots, 6)$.

iteration. It represents total *direct* and *indirect* feedforward rework probabilities which control the level of concurrency in sequential iteration. RC is a lower-triangular matrix which takes direct rework probabilities from $RP(i, j, k)$ $(i = 2, \ldots, n;$ $j = 1, \ldots, i-1;$ $k = 1, 2, \ldots)$ and adds them with indirect rework probabilities. The indirect probability (i, j) represents the probability of task i doing rework caused by task j through the intermediary of other tasks between i and j. For example, $RC(5, 2)$ in Fig. 4 is computed as the sum of indirect rework probabilities between tasks 2 and 5 through tasks 3 and 4 as intermediaries $(0.5 \times 0.1 + 0.5 \times 0.4 = 0.25)$. $RC(i, j)$ $(i > j)$ is computed at STEP3 of the simulation algorithm when determining the concurrency of tasks i and j when task j is reworked. RC is a dynamic variable during state transitions, so that it is updated whenever there is a feedback rework in the same loop. The algorithm to compute RC is explained in Appendix A.

During a simulation run, the model assumes that a task can be performed even though there exists an upstream dependent task being reworked if the total rework probability between the two tasks in the RC is less than the probability $P_{\text{tolerance}}$, a prespecified *rework risk tolerance*. In the example of Fig. 3, if $RC(2, 1) < P_{\text{tolerance}}$, the model simulates that both tasks a and b are reworked concurrently when both must be reworked. Otherwise, task b waits until the rework of task a is completed, at which time, new information from task a becomes available. In the latter case, if the rework of task a does create additional rework for task b, the total amount of rework of task b is between the maximum and the sum of reworks generated by tasks a and c. The model uses the latter as the default amount of rework required for task b and assumes that this quantity cannot exceed the task's original duration, diminished by the learning curve effect.

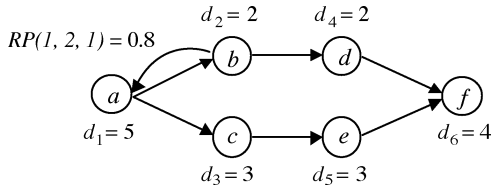


Fig. 5. Network example for resource priorities: task(or activity)-on-node network.

E. Measures and Rules for Resource Priorities

An iterative project network we study in this paper is distinguished from a project network without iteration in that the network is dynamic, meaning that a task in each state is probabilistically determined and there can be an infinite number of states with variable task sequences. Thus, the various exact approaches and metaheuristic approaches such as genetic algorithms, simulated annealing, and tabu search designed for a *static* resource-constrained project network without iteration (surveyed in [20] and [39]) are not suitable for such a *dynamic* network. Also, the best classical priority rules such as the latest finish time, the minimum slack, the minimum worst case slack, and the greatest rank positional weight (tested by [6], [10], [19], [32], and [57]) are not defined in an iterative project network since a task can be probabilistically repeated after its first execution. To see this, consider the hypothetical example in Fig. 5, where it is assumed that a resource conflict exists only between tasks b and c and there is no learning benefit during iterations. Without taking into account the possible iterations represented by an arc from task b to task a, all the classical heuristic rules mentioned above will give a higher priority for the limited resource to task c than task b since task c is on the critical path while task b is not. However, with 80% chance, task b is more critical than task c due to the possible rework of task a caused by task b. Thus, a heuristic measure in an iterative project network must be invented to reflect such iteration effect.

In this paper, a *rework-adjusted rank positional weight* (RARPWP) is proposed as a good measure that can help a project manager to determine task priorities in an iterative, resource-constrained project. The rank positional weight is one of the measures that Cooper [18] originally proposed and other researchers [6], [32], [57] found among the best for minimizing lead time in project networks without iteration. (See Appendix B for the definition.) In order to account for iteration, we define the rework-adjusted duration (RAD) of a task as the expected value of the sum of its duration and the delay of a project due to successive rework it creates to its preceding tasks, assuming no resource constraints in the project network. The RARPW is computed by replacing the deterministic duration of a task in Cooper's definition with the RAD. The model initially computes the RADs and RARPWs of tasks assuming no resource constraints. During a simulation run with resource constraints, the RARPWs of active tasks in each state are reduced by the state duration during iteration as the expected delay of a project due to iteration decreases. Thus, the RARPW is a dynamic priority measure having different values during state transitions.

The model determines priorities by the heuristic rules, whereby a task has a higher priority if:

ID	Task Name	Exp. Duration			Learning
		Opt.	Likely	Pess.	
1	Prepare UAV Preliminary DR&O	1.9	2.0	3.0	0.35
2	Create UAV Preliminary Design Configuration	4.8	5.0	8.8	0.20
3	Prepare Surfaced Models & Internal Drawings	2.7	2.8	4.2	0.60
4	Perform Aerodynamics Analyses & Evaluation	9.0	10.0	12.5	0.33
5	Create Initial Structural Geometry	14.3	15.0	26.3	0.40
6	Prepare Structural Geometry & Notes for FEM	9.0	10.0	11.0	1.00
7	Develop Structural Design Conditions	7.2	8.0	10.0	0.35
8	Perform Weights & Intertias Analyses	4.8	5.0	8.8	1.00
9	Perform S&C Analyses & Evaluation	18.0	20.0	22.0	0.25
10	Develop Freebody Diagrams & Applied Loads	9.5	10.0	17.5	0.50
11	Establish Internal Load Distributions	14.3	15.0	26.3	0.75
12	Evaluate Structural Strength, Stiffness, & Life	13.5	15.0	18.8	0.30
13	Preliminary Manufacturing Planning & Analyses	30.0	32.5	36.0	0.28
14	Prepare UAV Proposal	4.5	5.0	6.3	0.70

(a)

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1														
2	1								1					
3		1		1										
4	1		1											
5	1		1			1		1				2	1	
6	1				2									
7	1					2								
8						1						1		
9	1		1					1						
10				1		1	1	1			1			
11						1	1	1		2				
12	1					1	1			2	2			
13	1				1							1		
14	1	1	1	1	1	1	1	1	1	1	1	1	1	

(b)

	2	3	4	5	6	7	8	9	10	11	12	13
2							0.2					
3	0.5		0.4									
4		0.5										
5		0.5			0.1		0.1				0.3	0.1
6				0.4								
7					0.4							
8					0.5						0.5	
9		0.5	0.5			0.5						
10			0.1		0.5	0.2	0.1			0.4		
11					0.5	0.5	0.5		0.5			
12					0.4	0.5			0.5	0.4		
13				0.5							0.4	

(c)

Fig. 6. Model inputs for UAV project. (a) Project table. (b) $DSM(i,j)$ $(i,j = 1,\ldots,14)$. (c) $RP(i,j,1)$ and $RI(i,j)$ $(i,j = 2,\ldots,13)$.

1) it has been in process;
2) it has a higher user-specified priority;
3) it has a greater RARPW;
4) it is sequenced more upstream [from 1) to 4) in order of significance].

Rule 1) implies that a task cannot be interrupted once it has started (nonpreemption). The user-specified priority rule 2) can be used when resource priorities should be determined considering project objectives other than minimum lead time. Rule 3) stipulates that a higher priority is given to the task that exposes the project to higher schedule risk. Rule 4) is applied as a tie breaker when tasks have the same RARPW.

IV. Industrial Application Example

Extensive numerical experimentation was undertaken to test the simulation method. The computer program is written in Visual Basic and added in to Microsoft Excel which is used to receive model inputs and to display analysis results. In this paper, the uninhabited aerial vehicle (UAV) preliminary design process at an aerospace company is presented as a sample application. The data are from the first DSM-based simulation model by Browning and Eppinger [15] and extended to perform further analyses using additional modeling parameters. Fig. 6 shows the basic model inputs.

A. Results Using Basic Modeling Parameters

We begin with the same conditions of the first DSM-based simulation model [15] as follows:

1) 0 and 100th percentiles for optimistic and pessimistic duration estimates;
2) constant rework probabilities in all iterations;
3) learning curve benefit only in the first iteration;
4) zero rework risk tolerance, $P_{\text{tolerance}}$;
5) no overlapping;
6) no resource conflicts.

Simulation of the model yields an average lead time of 146.8 days and a standard deviation (s.d.) of 17.0 days after 2000 simulation runs. The number of simulation runs must be chosen such that both the sampled task durations and random numbers for rework probabilities closely follow the selected triangular and uniform distributions, respectively. The probability distribution of the lead time shown in Fig. 7 is skewed to the right because the lead time becomes longer as more iterations take place. Both the average and standard deviation are higher than those obtained by Browning and Eppinger (average 138, s.d. 14). This is mainly because the new model accounts for all the successive feedforward rework, while the earlier model does not.

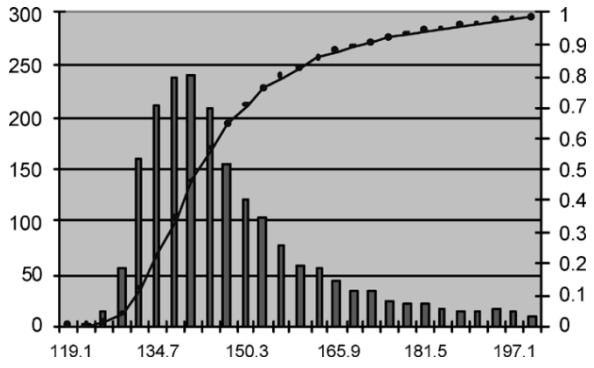


Fig. 7. Probability distribution of lead time with basic inputs.

Basic	(1)	(1), (2)	(1) - (3)	(1) - (4)
(146.8, 17.0)	(152.2, 20.1)	(149.3, 15.8)	(148.5, 14.2)	(144.2, 13.1)

<Note>
(i) (#, #): (average, standard deviation)
(ii) "(1)-(3)" denotes the results of the model with assumptions (1) through (3), and soon.

Fig. 8. Results using additional modeling parameters.

B. Results Using Additional Modeling Parameters

The new simulation model allows great flexibility to account for more general features of dynamic development processes. Fig. 8 summarizes the results using additional modeling parameters as follows.

1) Optimistic and pessimistic duration estimates are 10th and 90th percentiles, respectively.
2) Rework probabilities decrease in each iteration by 50%.
3) Maximum learning curve is 50% of that in the first iteration.
4) Rework risk tolerance $P_{\text{tolerance}} = 0.3$, i.e., more concurrency is allowed when tasks iterate.

Under the new assumption about task duration estimates in 1), both the average and standard deviation are greater than those with 0 and 100th percentile estimates. This is mainly due to the right-skewness of most of the task duration distributions. Also, the existence of multiple paths in the project contributes to the increase of the lead time. Incorporating dynamic characteristics of sequential iteration in 2) and 3), and increased task concurrency in 4), the model predicts smaller averages and standard deviations of the lead time (average 144.2, s.d. 13.1). The cumulative effect of the additional modeling parameters from 2) to 4) is a 5.3% decrease from the lead time under the assumption 1). Note that this difference will be more significant with more tasks and iterations.

Ignoring feedback marks in the DSM, i.e., assuming no sequential iterations, the path along the tasks 1-2-3-5-6-7-10-11-12-13-14 is a critical path when tasks have most likely durations. This implies that the lead time can be reduced by overlapping the tasks along this path such as the development of surfaced models (task 3) and structural design (tasks 5–7). Other important leverage points for reducing the lead time are the feedback marks. By transferring preliminary review decisions or testing results to upstream tasks, feedback rework can start earlier. This allows for the accelerations of iterative rework. Under the following overlapping scenario, the average lead time is reduced to 137.7 days.

5) For the six information flows marked by "2" in the DSM in Fig. 6(b), tasks are planned to complete 25% of work with preliminary inputs before receiving final updates, and expected to redo 50% of work completed without final updates.

The above scenario includes the overlapping between tasks 12 and 5 in feedback rework. The following scenario, for example, would cause such overlapping.

> As a result of structural evaluation in task 12, it may become necessary to redesign the structural geometry of specific subsystems before having evaluation results for the entire system. The need to redesign some subsystems (task 5) can be detected early in a series of tests in task 12, whereas the need for additional weight and inertial analyses (task 8) can be determined only at the end of the tests.

All the above analyses are performed under the assumption that there is no resource conflict among the tasks in the UAV project. This assumption is reasonable because tasks that can be performed in parallel are related to different disciplines, therefore, no resource sharing is necessary between those tasks. In multiple project environments, however, the resources belonging to the same functional group may need to get involved in tasks in different projects during a certain period of time. In this case, optimal resource allocation toward project objectives becomes an important issue. For the purpose of illustration, an example situation is tested as follows.

6) Tasks 7 and 8 compete for limited resources and one of them should be delayed.

Under assumptions 1)–5), the RARPWs of tasks 7 and 8 are computed as 104.5 and 122.9, respectively. Following the resource priority rules, the model assigns resources to task 8 and delays task 7. Then, both tasks 7 and 8 become critical and the project is delayed by the duration of task 8. The average computation time of 2000 simulation runs with additional modeling parameters in 1)–6) is 8 s in the system of Pentium 4 1.60 GHz CPU with 256 MB RAM.

Fig. 9 shows an example of a simulated Gantt chart. This is only one of many simulation runs under assumptions 1)–6). The numbers inside the bars indicate the amounts of overlapped work performed in prior state(s). Even though the scenario shows that tasks rework during states 5 and 14–19, it delays the entire project only in states 5 and 14 since no rework is added to task 13 after starting earlier. This is due to the predetermined work policy for increased task concurrency during iterations. Since total rework probabilities are less than 0.3 except for task 5, preliminary manufacturing planning and analyses (task 13) are simulated to work concurrently with functional performance analyses (tasks 8–11) after redesigning structural geometry (task 5).

V. Applications to Managerial Decision Making

Previous design process models have provided practical insights for process understanding and improvements. The dis-

ID \ State	D (w/o rework)	D (w. rework)	1	2	3	4	6	7	8	9	10	11	12	13	20	21
1	2.0	2.0														
2	5.9	5.9														
3	2.9	3.8														
4	8.9	8.9														
5	17.0	19.2												05		
6	9.0	10.1						2.2								
7	10.3	10.3														
8	7.7	11.5														
9	20.2	21.7														
10	16.3	18.7														
11	15.5	23.8										3.9			06	
12	14.8	16.7												3.7		
13	34.7	34.7														
14	5.0	5.0														

Fig. 9. Example of a simulated Gantt chart.

tinctive feature of this model is that it has greater flexibility and generality to describe real engineering design processes. Engineering management can benefit from this richer model through better project planning and control in several ways discussed in this section.

A. Finding an Effective Task Ordering

The process model presented in this paper is a predictive model, not an optimization model. However, the model can be easily utilized to find an effective task ordering by comparing the predicted outcomes of different task sequences. A relevant objective function might be, for instance, a function of the mean and variance of lead time, and the project due date. Note that finding an optimal task ordering is an $\mathrm{O}(n!)$ operation (where n is the number of tasks) if we allow tasks to be positioned anywhere in the sequence [15]. By fixing some logical precedence relationships between tasks, computation time can be reduced.

B. Setting Appropriate Due Date and Buffer

By analyzing the probability distribution of lead time, an appropriate due date can be chosen with predictable confidence [15]. When the schedule deadline is given, the risk that the project fails to complete before this date can be assessed via the simulation. When such schedule risk is high, the model can be used to evaluate different improvement efforts such as adding resources, overlapping tasks, executing fewer or faster iterations, etc. This perspective may facilitate decision making and communications between senior management and the project team. When a working project plan is constructed (usually with deterministic task durations), a project buffer can be added at the end representing the necessary aggregate safety time based on the simulation result. Such analysis adds rigor to Goldratt's critical chain method [26].

C. Finding Areas for Process Improvement

The model can be used to evaluate various process improvements through simulation by adjusting model parameters. Below are examples of model applications for this purpose.

- **Task criticality**: The sensitivity of project lead time to variation in task duration provides a measure of task criticality (see Elmaghraby [22] for a review of various sensitivity measures). The classical measures such as path criticality index and activity criticality index are not defined in an iterative project network where tasks repeat probabilistically and their sequences are not fixed. The cruciality index Williams [63] proposed can be computed in our model but it has significant drawbacks [22]. Thus, we suggest to use the sensitivity of the mean and variance of the project lead time, in particular, derivatives of the project lead time with respect to the mean and variance of task duration owing to its ease of computation [12]. These measures can be effectively used in a resource-constrained iterative environment, where conventional slack and resource-constrained slack [11] cannot be defined.
- **Strategic work policy for concurrency**: The rework risk tolerance defined earlier can be used as guidance for work policy during iterations. Lead time can be reduced by increasing concurrency level strategically, although this may result in increased development costs.
- **Faster and/or fewer iterations**: Smith and Eppinger [49] proposed two general strategies for accelerating iterative processes. Faster iterations can be achieved by increasing learning curve and/or decreasing rework impact, i.e., reducing the amount of rework. For instance, the efficient use of information technology such as computer-aided tools could help enable this effect. Fewer iterations can be achieved by decreasing rework probabilities. Well-defined interfaces between tasks, and well-established coordination and communication routes between project members could reduce the number of iterations. The proposed model can be used to identify the most efficient points for such improvements.

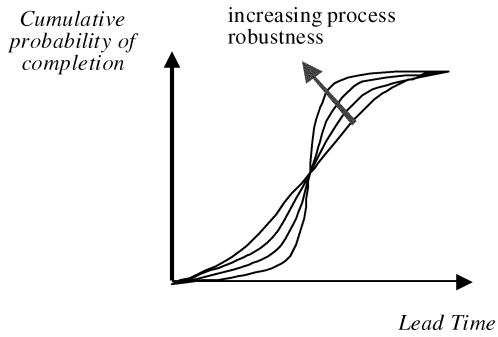


Fig. 10. Process robustness.

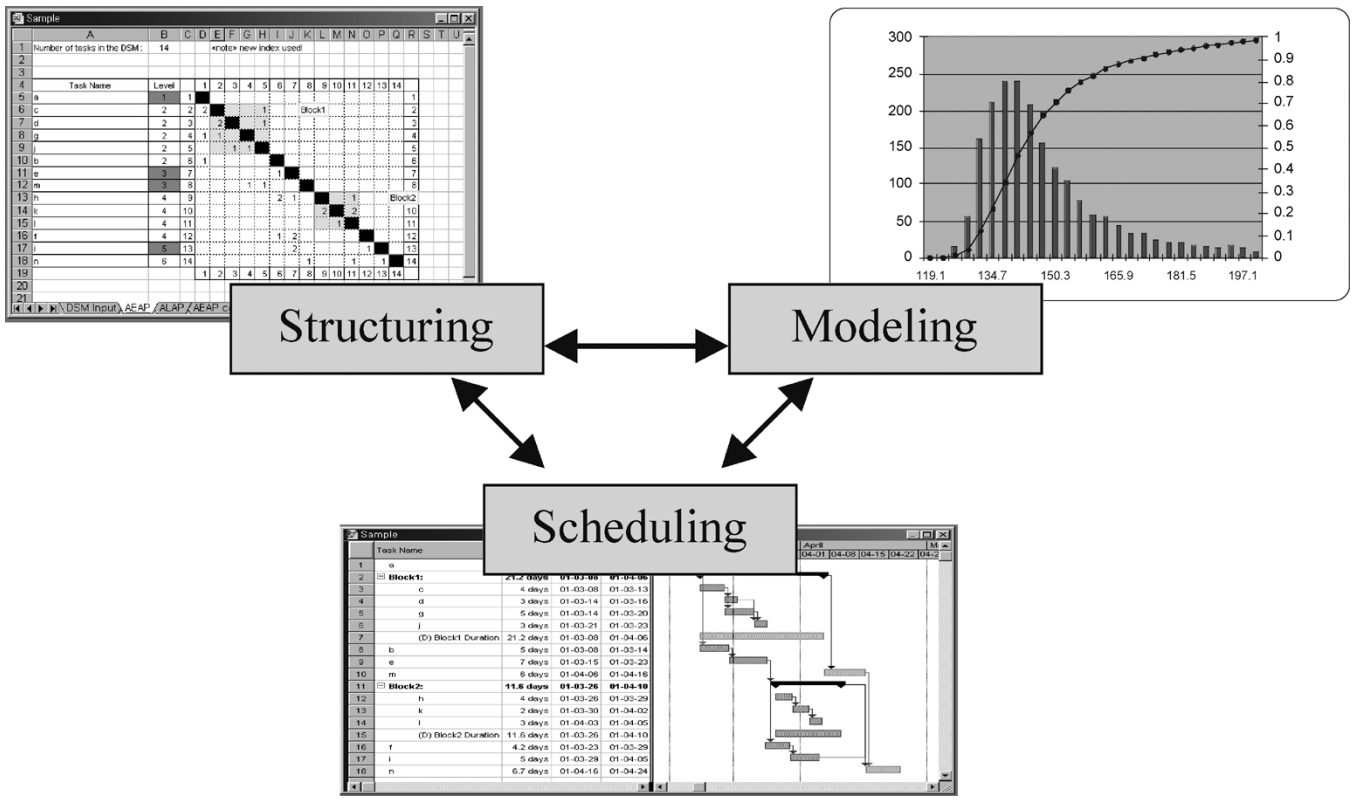


Fig. 11. Integrated project management framework.

- **Process robustness**: Yassine *et al.* [65] discussed process robustness in a product development project as the ability of a process to absorb design changes. Fig. 10 shows that more robust processes are less sensitive to variance of project parameters, leading to less variance of project lead time. The model in this paper is useful to assess the robustness of alternative process configurations.

D. Ongoing Risk Assessment

The inputs of our model can be easily updated to incorporate current information as the project progresses. For instance, initial estimates for task durations, overlap amounts, and impacts can be updated or replaced with actual values as they become available. Rework parameters may be updated to represent foreseeable iterations. As uncertainties diminish while the project advances, the variance of lead time becomes smaller. The schedule risk can be identified quantitatively and its assessment can facilitate proactive risk management efforts.

E. Evaluating Multiple Projects

The model can also be applied in a multiproject environment by representing each project on a different path. In this manner, it is possible to consider resource allocation across the projects and the effect of such constraints on completion of all of the projects.

VI. Discussion

A. Limitations and Extensions of the Model

The model assumes that the processing time of each task is independent of those of other tasks. However, when uncertainties affect multiple tasks, independent duration distributions cannot be assumed [62]. Various methods have been developed to model interdependencies, from estimation of correlation coefficients between tasks to use of joint distributions. Sampling of task duration using such methods are possible in commercial software packages such as @RISK. However, difficulties remain in assessing correlation among task durations.

The model does not guarantee optimal resource allocation to minimize lead time. This is a limitation of a richer model incorporating iteration and overlapping in multiple paths. A heuristic resource priority rule can no longer exploit slack (float) under the existing definition in the iterative project network where a probabilistic rule is applied. Thus, the model resorts to the heuristic rule using the RARPW accounting for probabilistic rework, and uses the sensitivity of project lead time to variation of a task duration as a measure of criticality. However, a comprehensive test of the RARPW in numerical examples is left for future work.

In this model, we assume a fixed resource pool for the project and constant resource requirements for tasks. The model can be extended by allowing variable resource capacity and requirements.

The model uses a simple assumption that cumulative overlap impact is the sum of single impacts when a task is overlapped with more than one task. This is not useful to predict the progress

of concurrent execution of tightly coupled tasks. More accuracy can be achieved if we could model frequent bidirectional information exchange and consequent impacts between parallel tasks.

Finally, the model can be further extended by incorporating development cost as in Browning and Eppinger [15]. An interesting modeling issue is how tightly to correlate the cost and time of each development task in the model.

B. Integrated Project Management Framework

The process model in this paper can be incorporated into an integrated project management framework as illustrated in Fig. 11. The framework streamlines project planning and control through structuring, modeling, and scheduling. In the structuring module, the DSM method is used to structure the information flows among tasks and capture the iteration loops. In the modeling module, the generalized process model in this paper predicts complex behaviors of iterative processes. Using the results of analyses from the structuring and modeling modules, a network-based schedule in the form of the Gantt chart is developed in the scheduling module. The schedule can be used as the basis for monitoring and control of the project. Under this integrated framework, the positive aspects of the methods used in each module can be utilized, while overcoming the limitations of standalone applications.

C. Validity of the Model

The work presented in this paper is a methodological work rather than a theoretical or empirical work. The proposed predictive model can improve important managerial decision making in various ways, as discussed. It has a high "face validity" according to the criteria of Smith and Morrow [50] since it addresses an important managerial issue, possesses reasonable computational tractability, and uses modeling parameters and assumptions based on the existing literature surveyed in Section I. Smith and Morrow also observed that limitations in the applicability of process models arise from the modeler's need to reduce the complex situation to a more structured form in order to have it fit in the modeling framework and from the lack of quantitative modeling approach. Our model has improved the applicability of earlier process models in this context because it addresses various limitations they faced. However, although the model was tested using real industrial data under some hypothetical scenarios to show its utility, more extensive applications in various project settings need to be undertaken to further demonstrate the applicability and validity of the model. Currently, these extended efforts are undergoing by several (volunteer) project managers through the industry collaboration program at our schools and the website www.dsmweb.org. We, therefore, expect follow-up empirical work using the integrated framework proposed in this paper. One of them is found already in Duniam [21].

VII. Conclusion

This paper presents a DSM-based process model using advanced simulation. The model accounts for important characteristics of engineering design processes, including information transfer patterns, uncertain task durations, resource conflicts, overlapping and sequential iterations, and task concurrency. The model addresses several limitations of previous analytical and simulation-based approaches. It can be applied to a wide range of processes, where iteration takes place among sequential, parallel, and overlapped tasks in a resource-constrained project. Increased understanding of realistic behavior of engineering design processes can be achieved through modeling information flows and predicting distributions of project lead time. The model is also useful for evaluating different project plans and for identifying strategies for process improvements. Proactive risk management can be achieved by assessing the status of the project as it progresses.

Appendix A
Algorithm to Compute Rework Concurrency (RC)

The RC between tasks a and b is computed as follows.

For $i = a+1, \ldots, b$,
1) Set $\mathrm{RC}(i,j) = \mathrm{RP}(i,j,1)$ for $j = a, \ldots, b-1$.
2) For $i > a+1$, execute the following loop:
for $j = i-2$ to 1 decreasing by 1
for $k = j+1$ to $i-1$

$$\mathrm{RC}(i,j) = \mathrm{RC}(i,j) + \mathrm{RP}(k,j,1) \times RC(i,k)$$

next k
next j

Appendix B
Definition of the Rank Positional Weight

Cooper [18] defined a rank positional weight of task i as follows:

$$rpw_i = d_i + \sum_j d_j$$

where

d_i expected duration of task i;
$\sum_j d_j$ sum of all expected durations over all successors of task i.

(Note: a set of successors includes all downstream tasks that receive outputs from the task.)

By adding the summation part in the above definition, the RPW reflects the global importance of a task.

Acknowledgment

The authors would like to thank Prof. T. Browning and Prof. A. Yassine for helpful comments on this work. They also appreciate the comments of Prof. A. Marucheck and anonymous reviewers.

References

[1] V. Adlakha, "An improved conditional Monte Carlo technique for the stochastic shortest path problem," *Manage. Sci.*, vol. 32, no. 10, pp. 1360–1367, 1986.

[2] P. Adler, A. Mandelbaum, V. Nguyen, and E. Schwerer, "From project to process management: An empirically-based framework for analyzing product development time," *Manage. Sci.*, vol. 41, no. 3, pp. 458–484, 1995.

[3] R. Ahmadi and H. Wang, "Managing development risk in product design processes," *Oper. Res.*, vol. 47, no. 2, pp. 235–246, 1999.

[4] F. AitSahlia, E. Johnson, and P. Will, "Is concurrent engineering always a sensible proposition?," *IEEE Trans. Eng. Manage.*, vol. 42, no. 2, pp. 166–170, 1995.

[5] C. Alexander, *Note on the Synthesis of Form*. Cambridge, MA: Harvard Univ. Press, 1964.

[6] R. Alvarez-Valdes and J. Tamarit, "Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis," in *Advances in Project Scheduling*, R. Slowinski and J. Weglarz, Eds. Amsterdam, The Netherlands: Elsevier, 1989, pp. 113–134.

[7] J. Andersson, J. Pohl, and S. Eppinger, "A design process modeling approach incorporating nonlinear elements," in *Proc. ASME Design Eng. Tech. Conf.*, 1998, no. DETC98–5663.

[8] U. Belhe and A. Kusiak, "Modeling relationships among design activities," *J. Mech. Design*, vol. 118, no. 4, pp. 454–460, 1996.

[9] J. Blazewicz, J. Lenstra, and A. Rinnooy Kan, "Scheduling subject to resource constraints: Classification and complexity," *Discrete Appl. Math.*, vol. 5, pp. 11–24, 1983.

[10] F. Boctor, "Some efficient multiheuristic procedures for resource-constrained project scheduling," *Euro. J. Oper. Res.*, vol. 49, pp. 3–13, 1990.

[11] J. Bowers, "Criticality in resource constrained networks," *J. Oper. Res. Soc.*, vol. 46, pp. 80–91, 1995.

[12] R. Bowman, "Stochastic gradient-based time-cost tradeoffs in PERT networks using simulation," *Annal. Oper. Res.*, vol. 53, pp. 533–551, 1994.

[13] G. Brooks and C. White, "An algorithm for finding optimal or near optimal solutions to the production scheduling problem," *J. Ind. Eng.*, pp. 34–40, Jan.–Feb. 1965.

[14] T. Browning, "Applying the design structure matrix to system decomposition and integration problems: A review and new directions," *IEEE Trans. Eng. Manage.*, vol. 48, no. 3, pp. 292–306, 2001.

[15] T. Browning and S. Eppinger, "Modeling impact of process architecture on cost and schedule risk in product development," *IEEE Trans. Eng. Manage.*, vol. 49, no. 4, pp. 443–458, Aug. 2002.

[16] J. Burt and M. Garman, "Conditional Monte Carlo: A simulation technique for stochastic network analysis," *Manage. Sci.*, vol. 18, no. 3, pp. 207–217, 1971.

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

[18] D. Cooper, "Heuristics for scheduling resource-constrained projects: An experimental investigation," *Manage. Sci.*, vol. 22, no. 11, pp. 1186–1194, 1976.

[19] E. Davis and J. Patterson, "A comparison of heuristic and optimum solutions in resource-constrained project scheduling," *Manage. Sci.*, vol. 21, no. 8, pp. 944–955, 1975.

[20] E. Demeulemeester and W. Herroelen, *Project Scheduling: A Research Handbook*, ser. International Series in Operations Research and Management Science. Norwell, MA: Kluwer, 2002.

[21] P. Duniam, "Using DSM techniques to compare integrated product and process developments at two major chemicals organizations," Queensland Univ., Brisbane, Australia, Tech. Rep., 2003.

[22] S. Elmaghraby, "On criticality and sensitivity in activity networks," *Euro. J. Oper. Res.*, vol. 127, pp. 220–238, 2000.

[23] S. Elmaghraby and J. Kamburowski, "The analysis of activity networks under generalized precedence relations (GPRs)," *Manage. Sci.*, vol. 38, no. 9, pp. 1245–1263, 1992.

[24] S. Eppinger, M. Nukala, and D. Whitney, "Generalized models of design iteration using signal flow graphs," *Res. Eng. Design*, vol. 9, no. 2, pp. 112–123, 1997.

[25] S. Eppinger, D. Whitney, R. Smith, and D. Gebala, "A model-based method for organizing tasks in product development," *Res. Eng. Design*, vol. 6, no. 1, pp. 1–13, 1994.

[26] E. Goldratt, *Critical Chain*. Great Barrington, MA: The North River Press, 1997.

[27] A. Ha and E. Porteus, "Optimal timing of reviews in the concurrent design for manufacturability," *Manage. Sci.*, vol. 41, no. 9, pp. 1431–1447, 1995.

[28] G. Hoedemaker, J. Blackburn, and L. Wassenhove, "Limits to concurrency," *Decision Sci.*, vol. 30, no. 1, pp. 1–18, 1999.

[29] D. Keefer and W. Verdini, "Better estimation of PERT activity time parameters," *Manage. Sci.*, vol. 39, no. 9, pp. 1086–1091, 1993.

[30] J. Kelley and M. Walker, "Critical-path planning and scheduling," in *Proc. Easter Joint Comput. Conf.*, 1959, pp. 160–173.

[31] S. Kline, "Innovation is not a linear process," *Res. Manage.*, pp. 36–45, Jul.–Aug. 1985.

[32] R. Kolish, "Efficient priority rules for the resource-constrained project scheduling problem," *J. Oper. Manage.*, vol. 14, pp. 179–192, 1996.

[33] V. Krishnan, S. Eppinger, and D. Whitney, "A model-based framework to overlap product development activities," *Manage. Sci.*, vol. 43, no. 4, pp. 437–451, 1997.

[34] W. Ledet and D. Himmelblau, "Decomposition procedures for the solving of large scale systems," *Adv. Chem. Eng.*, vol. 8, pp. 185–224, 1970.

[35] R. Levitt, G. Cohen, J. Kunz, C. Nass, T. Christiansen, and Y. Jin, "The virtual design team: simulating how organization structures and information processing tools affect team performance," in *Computational Organization Theory*, K. M. Carley and M. J. Prietula, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1994.

[36] C. Loch and C. Terwiesch, "Communication and uncertainty in concurrent engineering," *Manage. Sci.*, vol. 44, no. 8, pp. 1032–1048, 1998.

[37] D. Malcolm, J. Roseboom, C. Clark, and W. Fazar, "Application of a technique for research and development program evaluation," *Oper. Res.*, vol. 7, no. 5, pp. 646–669, 1959.

[38] M. McKay, R. Beckman, and W. Canover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[39] K. Neumann, C. Schwindt, and J. Zimmermann, *Project Scheduling With Time Windows and Scare Resources*, ser. Lecture Notes in Economics and Mathematical Systems. New York: Springer-Verlag, 2002.

[40] K. Neumann and U. Steinhardt, "GERT Network and the Time-Oriented Valuation of Projects," *Lecture Notes in Economics and Mathematical Systems*, vol. 172, 1979.

[41] A. Pritsker, *Modeling and Analysis Using Q-GERT Networks*, 2nd ed. New York: Wiley, 1979.

[42] A. Pritsker and J. O'Reilly, *Simulation With Visual SLAM and AweSim*, 2nd ed. New York: Wiley, 1999.

[43] T. Roemer, R. Ahmadi, and R. Wang, "Time-cost tradeoffs in overlapped product development," *Oper. Res.*, vol. 48, no. 6, pp. 858–865, 2000.

[44] W. Sargent and A. Westerberg, "Speed-up in chemical engineering design," *Trans. Inst. Chem. Eng.*, vol. 42, pp. 190–197, 1964.

[45] A. Seila, V. Ceric, and P. Tandikamalla, *Applied Simulation Modeling*, 1st ed. Belmont, CA: Thomson, 2003.

[46] C. Sigal, A. Pritsker, and J. Solberg, "The use of cutsets in Monte Carlo analysis of stochastic networks," *Math. Comput. Simulation*, vol. 21, pp. 376–384, 1979.

[47] P. Smith and D. Reinertsen, *Developing Products in Half the Time*, 2nd ed. New York: Van Nostrand, 1995.

[48] R. Smith and S. Eppinger, "A predictive model of sequential iteration in engineering design," *Manage. Sci.*, vol. 43, no. 8, pp. 1104–1120, 1997.

[49] ——, "Identifying controlling features of engineering design iteration," *Manage. Sci.*, vol. 43, no. 3, pp. 276–293, 1997.

[50] R. Smith and J. Morrow, "Product development process modeling," *Design Studies*, vol. 20, pp. 237–261, 1999.

[51] D. Steward, "Partitioning and tearing systems of equations," *SIAM Numer. Anal.*, ser. B, vol. 2, no. 2, pp. 345–365, 1965.

[52] ——, "The design structure system: a method for managing the design of complex systems," *IEEE Trans. Eng. Manage.*, vol. 28, no. 3, 1981.

[53] H. Takeuchi and I. Nonaka, "The new product development game," *Harvard Bus. Rev.*, pp. 137–146, Jan.–Feb. 1986.

[54] B. Taylor and L. Moore, "R&D project planning with Q-GERT network modeling," *Manage. Sci.*, vol. 26, no. 1, pp. 44–59, 1980.

[55] C. Terwiesch and C. Loch, "Managing the process of engineering change orders: The case of the climate control system in automobile development," *J. Prod. Innov. Manage.*, vol. 16, pp. 160–172, 1999.

[56] K. Ulrich and S. Eppinger, *Product Design and Development*, 2nd ed. New York: McGraw-Hill, 2000.

[57] V. Valls, M. Perez, and M. Quintanilla, "Heuristic performance in large resource-constrained projects," Department D'Estadistica I Invecigacio Operativa, Universitat de Valencia, Tech. Rep. 92-2, 1992.

[58] R. Van Slyke, "Monte Carlo methods and the PERT problem," *Oper. Res.*, vol. 11, no. 5, pp. 839–860, 1963.

[59] J. Warfield, "Binary matrices in system modeling," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 5, pp. 441–449, 1973.

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

[61] D. Whitney, "Designing the design process," *Res. Eng. Design*, vol. 2, pp. 3–13, 1990.
[62] T. Williams, "Practical use of distributions in network analysis," *J. Oper. Res. Soc.*, vol. 43, pp. 265–270, 1992.
[63] ——, "Criticality in stochastic networks," *J. Oper. Res. Soc.*, vol. 43, pp. 353–357, 1992.
[64] A. Yassine and T. Browning, "Analyzing multiple product development projects based on information and resource constraints," *UIUC*, 2002. Working Paper.
[65] A. Yassine, D. Whitney, and T. Zambito, "Assessment of rework probabilities for simulating product development processes using the design structure matrix (DSM)," in *Proc. ASME Design Eng. Tech. Conf.*, 2001, no. DETC2001/DTM-21693.
[66] B. Zeigler, H. Praehofer, and T. Kim, *Theory of Modeling and Simulation—Integrating Discrete Event and Continuous Complex Dynamic Systems*, 2nd ed. New York: Academic, 2000.

Soo-Haeng Cho received the B.S. degree in mechanical and aerospace engineering from Seoul National University, Seoul, Korea, and the M.S. degree in engineering from Massachusetts Institute of Technology, Cambridge. He is currently working towards the Ph.D. degree in decisions, operations and technology management at UCLA Anderson School of Management, Los Angeles, CA.

He previously worked with Arthur D. Little as a Consultant, conducting various corporate and process restructuring projects across industries. His current research interests include new product strategy, technology innovation, and management of complex engineering processes.

Mr. Cho is a member of INFORMS. He received the Xerox/DTM Best Paper Award from the ASME International Design Theory and Methodology Conference in 2001.

Steven D. Eppinger (S'86–M'88) received the S.B., S.M., and Sc.D. degrees in mechanical engineering from the Massachusetts Institute of Technology (MIT), Cambridge.

He is the General Motors Professor of Management Science and Engineering Systems at the Sloan School of Management, MIT, and currently serves as the Deputy Dean. His research deals with the management of complex engineering processes. He is coauthor of the textbook *Product Design and Development* (McGraw-Hill, New York, 2004). He has published papers in the IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, *Management Science*, *ASME Journal of Mechanical Design*, *Research in Engineering Design*, *Journal of Engineering Design*, *Harvard Business Review*, and other publications.

Prof. Eppinger is a member of INFORMS.