Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development

Tyson R. Browning and Steven D. Eppinger, *Member, IEEE*

***Abstract*—To gain competitive leverage, firms that design and develop complex products seek to increase the efficiency and predictability of their development processes. Process improvement is facilitated by the development and use of models that account for and illuminate important characteristics of the process. Iteration is a fundamental but often unaddressed feature of product development (PD) processes. Its impact is mediated by the architecture of a process, i.e., its constituent activities and their interactions. This paper integrates several important characteristics of PD processes into a single model, highlighting the effects of varying process architecture. The PD process is modeled as a network of activities that exchange deliverables. Each activity has an uncertain duration and cost, an improvement curve, and risks of rework based on changes in its inputs. A work policy governs the timing of activity execution and deliverable exchange (and thus the amount of activity concurrency). The model is analyzed via simulation, which outputs sample cost and schedule outcome distributions. Varying the process architecture input varies the output distributions. Each distribution is used with a target and an impact function to determine a risk factor. Alternative process architectures are compared, revealing opportunities to trade cost and schedule risk. Example results and applications are shown for an industrial process, the preliminary design of an uninhabited combat aerial vehicle. The model yields and reinforces several managerial insights, including: how rework cascades through a PD process, trading off cost and schedule risk, interface criticality, and occasions for iterative overlapping.**

***Index Terms*—Activity network, budgeting, cycle time, design iteration, design structure matrix, engineering design management, process architecture, process modeling, process structure, product development, rework, risk management.**

I. Introduction

To INCREASE their competitiveness, firms that develop products have realized the importance of improving the efficiency and predictability of their design processes. Since process improvement requires process understanding [81], researchers and practitioners put effort into observing product design and development processes—looking for their important characteristics—and developing models that account for those features. Most of the advances in this area assume that the design process has an underlying structure [5], [7], [72]. An important characteristic of *product development* (PD) processes is that, unlike most business and production processes, they are described by terms like "creative," "innovative," and "iterative." At an interesting level of detail, PD processes do not proceed in a purely sequential fashion [28], [47], [58]. The activities in a PD process interact by exchanging information (e.g., [19] and [25]). The data that activities need to do their work effectively must be available in the right place, at the right time, and in the right format. The structure of this information flow has a bearing on process efficiency and predictability [35], [74]. In particular, the structure of the PD process impacts project cost [58] and cost and schedule risk [13]. Thus, PD can be described as a complex web of interactions, some of which precipitate a cascade of rework among activities. Models that highlight the characteristics of this network are helpful in improving our understanding of PD processes and ultimately their efficiency and predictability.

According to Hammer, a process is an organized group of related activities that work together to create a result of value. *Process architecture*—the elements of a process (activities) and their pattern of interaction—is an important process variable [80] [1] Just as different product architectures can deliver varied product capabilities and levels of effectiveness, alternative process architectures have different cost, duration, and risk characteristics. Much like a product can be improved through architectural innovation [41], process improvement includes architecting an efficient and predictable process. Processes are *systems* and benefit from the application of systems thinking and the tenets of systems engineering. Modeling and comparing alternative process architectures can provide insights to help navigate cost, schedule, and risk tradeoffs.

In trying to improve PD processes, planners and managers become interested in how activities should be arranged within the process, how rework cascades through the process, cost and schedule tradeoffs, outcome predictability, and the interplay between these issues. For instance, regarding cost and schedule tradeoffs, consider the range of possible cost and duration outcomes depicted in Fig. 1 [3], [56]. Five outcomes lie along an efficient frontier. Each of these outcomes indicates the shortest possible duration that can be achieved for the cost and *vice-versa* (given a firm's process capabilities). Many other outcomes lie in the shaded region: it is always possible to spend more time and money than necessary. A scatter plot of many simulated cost and schedule outcomes for a particular process will cover a certain area and will have its own feasible region. What can be done to improve the process by moving the frontier to the lower left? What can be done to increase the predictability of

Manuscript received April 24, 2000; revised October 1, 2001, March 1, 2002, and May 1, 2002. This work was supported in part by the Lean Aerospace Initiative at the Massachusetts Institute of Technology, Lockheed Martin Aeronautics Company, The Boeing Company, and a National Science Foundation graduate fellowship.

T. R. Browning is with Lockheed Martin Aeronautics Company, Fort Worth, TX 76101 USA (e-mail: tyson@alum.mit.edu).

S. D. Eppinger is with the Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: eppinger@mit.edu).

Digital Object Identifier 10.1109/TEM.2002.806709

[1]Process architecture describes the process activities and their relationships (ordering, interfaces, interdependencies, etc.), including relationships with external processes.

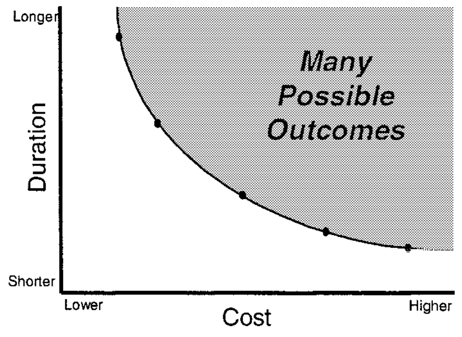


Fig. 1. Cost and duration tradeoffs.

the process—i.e., to ensure a desired outcome at a prespecified point along the frontier?

To address the above issues, we contribute a richer model of the PD process architecture. The model accounts for important attributes of each activity and deliverable. Activities may have uncertain cost and duration, rework caused by changes in particular inputs, and reduced cost and duration in successive iterations. Deliverables may vary in their volatility (propensity to change). A work policy governs the timing of activity execution and interaction—including whether activities can work concurrently—as a function of process architecture. The model incorporates aspects of several other models into a single integrated framework, and it introduces the concept of rework risk in terms of separate probability and impact components.

While the model enables a number of interesting analyses with multiple applications, we use it primarily to explore the impacts of alternative process architectures on cost and schedule risk in PD. For most of the analysis, we use multiple runs of a Monte Carlo simulation to produce cost and duration distributions. The distributions are evaluated against given budgets and deadlines to determine the probability of an overrun and the level of risk. This information is valuable for project planning and process improvement. The model is demonstrated with example data from an industrial process, an *uninhabited combat aerial vehicle* (UCAV) preliminary design process at an aerospace company. The paper also contributes and validates insights for both researchers and practitioners.

II. Model Constructs and Literature Review

A. PD as a Complex Web of Interactions

A powerful way to increase understanding of a process is to look at its parts and their relationships. Decomposition is the standard approach to addressing system complexity—desirable because it is generally possible to make more accurate estimates about simpler elements. However, it is generally more difficult to make accurate estimates of the effects on the overall system of relationships between simpler elements. The relationships among elements are an important characteristic that differentiates a system from a mere grouping of elements. As a kind of system, a process is defined not only by its decomposition into activities but also by how they work together [15], [16].

In practice, most process definitions and models account for a minimal amount of information about the element relationships or interfaces. A single input and output for each activity is often considered sufficient. However, especially in the early stages of PD, people and the activities they execute tend to provide and require a great deal of information to and from each other (e.g., [19] and [25]). [2] A large number of activity interfaces are necessary to document the full range of deliverable flow. Most process models do not attempt to elicit and represent the actual information flow, even though it is a major driver of process efficiency and predictability [16], [26], [35].

A process is often modeled as an activity network. Activity-on-arc [project evaluation and review technique (PERT)], activity-on-node, and other flowchart-type representations are widely used to represent activities and their precedence relationships. Despite the enormous amount of research on activity networks (see [32] for an excellent review) using these formats, they are not convenient for representing a large number of interfaces or for comparing alternative process architectures. The visual representation is too busy, making it difficult to discern architectural differences. And since many process flowcharts capture only a single input and output for any given activity, the full range of information flow is seldom represented. Using a flowchart for this purpose would simply be too complicated and cumbersome.

A *design structure matrix* (DSM) can also be used to represent a process [14], [35], [73]. The DSM shows activities and interfaces in a concise format. A DSM is a square matrix in which a cell on the diagonal represents each activity. Activity names are usually given to the left of the matrix. A mark in an off-diagonal cell indicates an activity interface. For each activity, its row shows its inputs and its column shows its outputs.[3] When activities are listed in temporal order, subdiagonal marks denote a feeding of deliverables forward in the process, from upstream activities to downstream activities, while superdiagonal marks indicate feedback. The DSM provides a simple way to visualize the structure of an activity network and to compare alternative process architectures.[4]

The DSM in Fig. 2 represents the network of PD activities for the UCAV preliminary design process. This DSM was built by asking an expert on each activity to list the inputs (mainly information) it requires and the outputs it produces. To distinguish alternative process architectures, we define a sequencing vector, $\boldsymbol{V}$, which is given by numbering the rows (and columns) in the DSM. By resequencing the activities in the DSM, and in conjunction with a work policy (discussed below), we alter the pattern of interactions among the activities, thus creating varied process architectures.

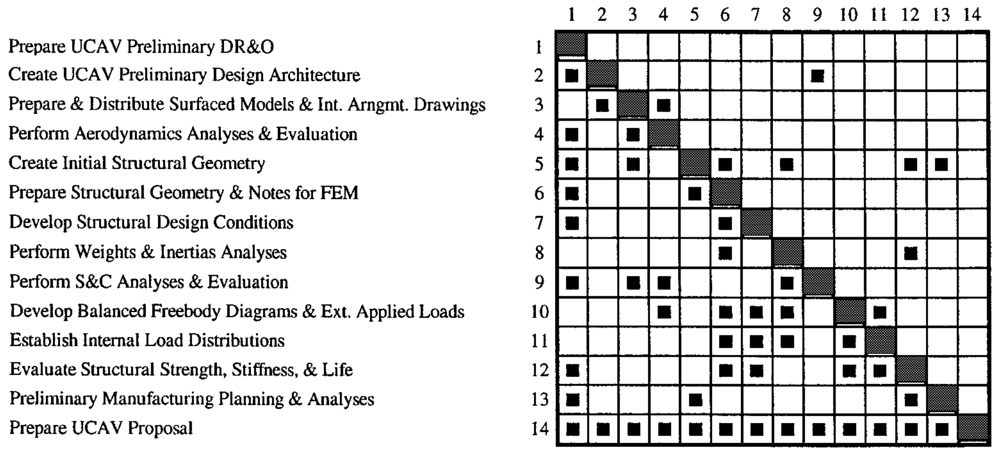
B. Activity Iteration

The iteration of design activities is a fundamental characteristic of PD [5], [12], [35], [47]. In many ways, PD is a creative,

[2]Burns and Stalker characterize PD as having many work elements, where "each is performed in response to information received; each involves altering, rearranging, or recomposing information or things; each ends with the transmission of the altered information or thing to somebody else" [19, p. 78].

[3]Some DSMs use the opposite convention—rows for outputs and columns for inputs. The two formats convey equivalent information.

[4]Another process modeling and representation approach we considered is the structured analysis and design technique (SADT), particularly its well-known subset, IDEF0. However, IDEF0 diagrams were more cumbersome to represent and manipulate than the DSM.



		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Prepare UCAV Preliminary DR&O	1	▨													
Create UCAV Preliminary Design Architecture	2	■	▨							■					
Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings	3		■	▨	■										
Perform Aerodynamics Analyses & Evaluation	4	■		■	▨										
Create Initial Structural Geometry	5	■		■		▨	■		■				■	■	
Prepare Structural Geometry & Notes for FEM	6	■				■	▨								
Develop Structural Design Conditions	7	■					■	▨							
Perform Weights & Inertias Analyses	8						■		▨				■		
Perform S&C Analyses & Evaluation	9	■		■	■				■	▨					
Develop Balanced Freebody Diagrams & Ext. Applied Loads	10				■		■	■	■		▨	■			
Establish Internal Load Distributions	11						■	■	■		■	▨			
Evaluate Structural Strength, Stiffness, & Life	12	■					■	■			■	■	▨		
Preliminary Manufacturing Planning & Analyses	13	■				■							■	▨	
Prepare UCAV Proposal	14	■	■	■	■	■	■	■	■	■	■	■	■	■	▨

Fig. 2. DSM representation of UCAV preliminary design process. DR&O = Design Requirements and Objectives. FEM = Finite Element Model. S&C = Stability & Control.

discovery process [60]. While product design quality improves with successive iterations, in a general sense [66], [67], [71], [81], iteration is a key driver of cost and schedule risk in PD projects [1], [13], [59], [77]. Understanding iteration becomes even more important when attempting concurrent engineering: activities that were once distinct and sequential are now intermingled or overlapped, resulting in more interactions and a greater need for coordination [79].

Most process modeling literature and software is oriented toward production or business processes, where the goal is to repeat a chain-like process without interwoven iterative loops. Thus, the shortcomings of standard flowchart representations in clearly representing many feedbacks are seldom exposed. However, much of the waste and inefficiency in iterative processes stems from these interactions and feedbacks—i.e., having to repeat activities because of changes in the information and/or assumptions upon which they were initially executed. Whereas some amount of iteration may be planned in order for designers to converge to a satisfactory design solution, much *unplanned* iteration stems from a poor process architecture [12], [59].

In recognition of the importance of iteration in PD processes, several models have been constructed to analyze it. An extension to PERT called generalized evaluation and review technique (GERT) [e.g., [57] and [63]] enables simulation-based analyses of activity networks with feedbacks. Several models have been built using signal flow graphs [6], [34], [45] and system dynamics [e.g., [36]]. However, none of these models is convenient for exploring alternative process architectures where the activities have many distinct deliverables.

Other recent efforts to model iteration utilize the DSM. Smith and Eppinger produced three DSM-based process models: one that assumes all interdependent activities are worked concurrently [53], [68], another that assumes such activities are attempted sequentially [69], and a hybrid of the first two [22], [70]. These models identify critical activities and find analytic solutions for process duration with a limited number of activities. Others have explored methods for improving processes by reducing feedback information [2], [50], [65], [73], [74]. Recent work by Yassine *et al.* [84] more explicitly defines dependencies in the DSM based on sensitivity to and variability of information. However, existing DSM-based models do not account for stochastic activity durations and costs, do not treat rework probabilities and impacts distinctively, and are quite limited in accounting for concurrency in large activity sets.

The model developed in this paper characterizes the PD process as a network of activities that exchange deliverables. If an activity does work and produces an output based on inputs or assumptions, then a change in either may imply rework for the activity. The accomplishment of that rework then changes the activity's outputs, thereby potentially affecting other activities in the same way (second-order rework [69]). Rework is not always a certainty; the risk of rework is a function of its probability and its consequences. Each input to each activity has a probability of changing (volatility) and a probability of a typical change causing rework for the activity (sensitivity).[5] These probabilities are multiplied to get the probability of rework

$$\begin{aligned} &P(\text{rework for an activity caused by}\\ &\quad \text{change in one of its inputs})\\ &= P(\text{change in the input})\\ &\cdot P(\text{the change affecting the activity}). \end{aligned} \tag{1}$$

Rework probabilities for the UCAV example are shown in $\boldsymbol{DSM}_{\text{xy1}}$ (Fig. 3), where

$$P(\text{rework for activity } i \text{ caused by a typical change in its input form activity } j) = \boldsymbol{DSM}_{ij1} \tag{2}$$

and the subscript 1 (number one) indicates the first DSM plane (a second is added below). Rework in an upstream activity i, caused by downstream activity j, can also cause *second-order rework*. That is, when an iteration occurs, and the process must backtrack from activity j to rework some activity i $(i < j)$, this provides the potential for the change in output from activity i to affect interim activities $(i+1, i+2, \ldots, j-1)$ and any completed, downstream activities $(j, j+1, \ldots, n)$ dependent on activity i. Thus, in $\boldsymbol{DSM}_{\text{xy1}}$, superdiagonal numbers $(x < y)$

[5]The concept of activity sensitivity to input changes as a cause for rework has been explored by several authors [e.g., [1], [48], and [52]].

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Prepare UCAV Preliminary DR&O	1	▨													
Create UCAV Preliminary Design Architecture	2	.4	▨							.2					
Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings	3		.5	▨	.4										
Perform Aerodynamics Analyses & Evaluation	4	.3		.5	▨										
Create Initial Structural Geometry	5	.4		.5		▨	.1		.1				.3	.1	
Prepare Structural Geometry & Notes for FEM	6	.1				.4	▨								
Develop Structural Design Conditions	7	.4					.4	▨							
Perform Weights & Inertias Analyses	8						.5		▨				.5		
Perform S&C Analyses & Evaluation	9	.4		.5	.5				.5	▨					
Develop Balanced Freebody Diagrams & Ext. Applied Loads	10				.1		.5	.2	.1		▨	.4			
Establish Internal Load Distributions	11						.5	.5	.5		.5	▨			
Evaluate Structural Strength, Stiffness, & Life	12	.4					.4	.5			.5	.4	▨		
Preliminary Manufacturing Planning & Analyses	13	.5				.5							.4	▨	
Prepare UCAV Proposal	14	.3	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	▨

Fig. 3. $\boldsymbol{DSM}_{\mathrm{xy1}}$ showing rework probabilities for UCAV process.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Prepare UCAV Preliminary DR&O	1	▨													
Create UCAV Preliminary Design Architecture	2	.5	▨							.1					
Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings	3		.3	▨	.5										
Perform Aerodynamics Analyses & Evaluation	4	.4		.8	▨										
Create Initial Structural Geometry	5	.1		.1		▨	.1						.3	.1	
Prepare Structural Geometry & Notes for FEM	6	.1				.3	▨								
Develop Structural Design Conditions	7	.5					.8	▨							
Perform Weights & Inertias Analyses	8						.5		▨				.5		
Perform S&C Analyses & Evaluation	9	.3		.3	.3				.3	▨					
Develop Balanced Freebody Diagrams & Ext. Applied Loads	10				.1		.5	.4	.3		▨	.3			
Establish Internal Load Distributions	11						.5	.5	.3		.3	▨			
Evaluate Structural Strength, Stiffness, & Life	12	.5					.3	.5			.5	.5	▨		
Preliminary Manufacturing Planning & Analyses	13	.9				.9							.3	▨	
Prepare UCAV Proposal	14	.5	.8	.8	.8	.8	.8	.8	.8	.8	.8	.8	.8	.8	▨

Fig. 4. $\boldsymbol{DSM}_{\mathrm{xy2}}$ showing rework impacts for UCAV process.

represent the probability of iteration (returning to previous activities), while the subdiagonal numbers $(x > y)$ note the probability of second-order rework (following a first-order iteration). In our basic model, these probabilities are held constant through successive iterations.[6]

Rework can also have a variable *impact* on an activity. While very likely, some changes in inputs can be absorbed by a robust activity with little impact [78], [79]. The consequences of changing other inputs may be more severe. The model uses an impact measure—the percentage of the activity that must be reworked—for each input to each activity.[7] Rework impacts for the UCAV example are given in a second DSM plane, $\boldsymbol{DSM}_{\mathrm{xy2}}$ (Fig. 4), where

$$\%(\text{rework for activity } i \text{ caused by a typical change in input from activity } j) = \boldsymbol{DSM}_{ij2}. \quad (3)$$

In both the probability and impact DSMs, in a few cases where more than one distinct deliverable passes though a single off-diagonal cell, the DSMs record only the most influential deliverable. Overall, the DSMs describe a kind of state transition matrix for the activities in a process.

For purposes of modeling, we assume that the process of interest has been chosen to be as modular as possible, thereby allowing us to ignore external influences. However, this assumption does not hold completely in practice, since every process is enmeshed in the context of a larger process, and external inputs impact the process's activities and therefore its outcome [16].

C. Activity Overlapping

One of the most intuitive approaches to decreasing cycle time in PD processes is to do more activities concurrently. However, overlapping dependent activities is problematic, since doing work based on assumptions or preliminary data is riskier than working with final data. Several have proposed models to explore aspects of this issue (e.g., [4], [31], [40], [42]). Carrascosa *et al.* [21] use work by Krishnan *et al.* [48], and Terwiesch and Loch [52], [76] to build a multistage task model that accounts for iteration probability and impact for a few overlapped activities. Roemer *et al.* [64] explore how overlapping affects time–cost tradeoffs. Most of these frameworks focus on overlapping just two activities (although some of the models use these couplets as building blocks for multi-staged processes).

[6]An extension to the model [23] allows for variable higher order rework parameters.

[7]Loch and Terwiesch [52] define the impact as a function of time, while this model relates the impact to the amount of work to be done, thereby influencing both time and cost.

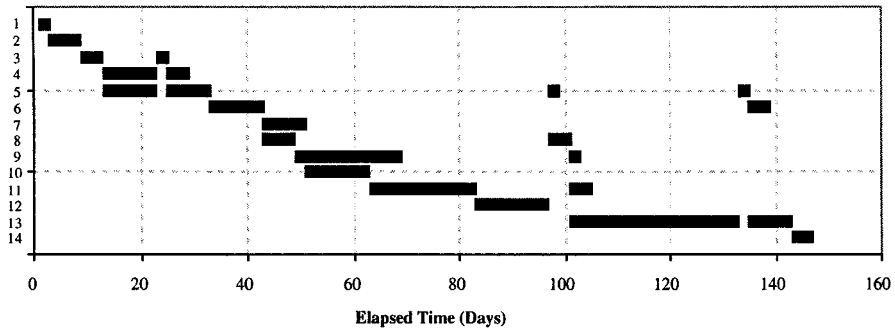


Fig. 5. Example Gantt chart of simulated process execution.

Our basic model allows limited activity overlapping. For now, we use a simple *work policy* that: 1) an activity may not begin until it has received all of its inputs from foregoing or "upstream" activities (those listed prior in the sequencing vector, $\boldsymbol{V}$) and 2) an activity can make assumptions about its inputs from subsequent or "downstream" activities. Thus, activities may work concurrently only if the downstream one does not depend on the upstream one for input. The work policy furthermore specifies that 3) only adjacent activities in $\boldsymbol{V}$ may work concurrently.[8] Thus, the sequence of activities in the DSM and the work policy governing their interaction and overlap prescribe a process architecture.

This work policy plays out as follows. The most upstream activity requiring work is identified and made *active*. Then, successive downstream activities are checked for dependence on the active activity. If the subsequent activity: 1) has work to do and 2) does not depend on the active activity, then this subsequent activity is also activated. Each subsequent activity requiring work is similarly checked for dependence on active activities until a dependent activity is found, at which point the current set of active activities has been determined. Hence, only *consecutive* activities (of those requiring work) may be active, making the set depend on the process architecture. Finished activities and assumptions about subsequent activities (superdiagonal marks in the DSM) are both ignored when determining the active set.

In the case of coupled activities, the basic work policy implies that one of the activities will go first and the other will wait. It also implies that downstream activities will stop and wait when upstream activities on which they depend are reactivated because of rework. The Gantt chart in Fig. 5 shows this effect. When rework is generated by activity four for activity three, activities four and five wait on the new results of activity three before proceeding. (Their work could be invalidated if they did not wait). McDaniel [53] found that strategic waiting conserves resources. However, causing certain individuals or groups to stop work temporarily requires clear management policies not present in many engineering environments. Our model can also be adjusted to accommodate alternative work policies.

D. Activity Cost and Duration

A variety of distributions or *probability density functions* (PDFs), notably the beta distribution [46], have been used to represent uncertainty in activity cost and duration [e.g., [62] and [82]]. These distributions usually exhibit positive skewness due to the tendency of work to expand to fill available time—and of human nature to relax when ahead—thus making it less likely that activities will finish early, even if they could. Positive skewness can be represented by a triangle distribution, which is simple to comprehend and build, requiring only three data points per activity: optimistic or *best case value* (BCV), *most likely value* (MLV), and pessimistic or *worst case value* (WCV). The BCV, MLV, and WCV are used to form a triangular PDF, denoted as TriPDF(BCV,MLV,WCV). For simplicity, we normalize the area under a TriPDF to equal one[9]

$$\begin{aligned} TriPDFArea &= \frac{\text{base} \cdot \text{height}}{2} \\ &= \frac{(WCV - BCV) \cdot P(MLV)}{2} = 1. \quad (4) \end{aligned}$$

Table I lists activity duration and cost data for the UCAV project. (Where necessary, the displayed data are rounded, and they are disguised to protect competitive information.) The model assumes that time and effort for all information transfers between activities are included in activity durations and costs and also that activity durations are independent of each other—i.e., that any such dependencies are accounted for by the specified interactions between the activities. Each activity's duration PDF also accounts for any "internal rework" (thereby obviating the need for any on-diagonal rework probabilities in the DSM).

Since activity cost depends somewhat on activity duration, the cost PDF for an activity often has a shape similar to the duration PDF. In fact, we expect substantial correlation for any pair of activity duration and cost samples. We use a sampling correlation coefficient of 0.9 for each activity. Alternatively, each activity could be given its own correlation value based on its individual cost-schedule elasticity [cf., [33]]. Or, the cost and duration of each activity could be represented by a joint cost and schedule PDF similar to the simulation model's output (discussed in Section IV-B).

E. Improvement Curves

Often it takes less effort to rework an activity than to do it the first time. An activity may have a large setup time (e.g., building a simulation model) where, once the infrastructure is

[8]That is, adjacent when ignoring completed activities. See discussion that follows.

[9]Alternatively, one could normalize the area of the TriPDF to 0.8 to assume a 10% margin on each end of the stated BCV and WCV. Estimates of the 10th and 90th percentiles of duration are typically more accurate than estimates of the absolute BCV and WCV durations [46].

TABLE I
ACTIVITY DATA FOR UCAV PRELIMINARY DESIGN PROCESS

Activities		Durations (days)			Costs ($k)			
ID#	Name	BCV	MLV	WCV	BCV	MLV	WCV	IC
1	Prepare UCAV Preliminary DR&O	1.9	2	3	8.6	9	13.5	35%
2	Create UCAV Preliminary Design Architecture	4.75	5	8.75	5.3	5.63	9.84	20%
3	Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings	2.66	2.8	4.2	3	3.15	4.73	60%
4	Perform Aerodynamics Analyses & Evaluation	9	10	12.5	6.8	7.5	9.38	33%
5	Create Initial Structural Geometry	14.3	15	26.3	128	135	236	40%
6	Prepare Structural Geometry & Notes for FEM	9	10	11	10	11.3	12.4	100%
7	Develop Structural Design Conditions	7.2	8	10	11	12	15	35%
8	Perform Weights & Inertias Analyses	4.75	5	8.75	8.9	9.38	16.4	100%
9	Perform S&C Analyses & Evaluation	18	20	22	20	22.5	24.8	25%
10	Develop Balanced Freebody Diagrams & External Applied Loads	9.5	10	17.5	21	22.5	39.4	50%
11	Establish Internal Load Distributions	14.3	15	26.3	21	22.5	39.4	75%
12	Evaluate Structural Strength, Stiffness, & Life	13.5	15	18.8	41	45	56.3	30%
13	Preliminary Manufacturing Planning & Analyses	30	32.5	36	214	232	257	28%
14	Prepare UCAV Proposal	4.5	5	6.25	20	22.5	28.1	70%

in place, it is easy to rerun the activity with new inputs. Also, activity participants may benefit from learning and adaptation. Some existing process models account for improvement curve effects [2], [3], [6], [45], [83]. Studying semiconductor projects at Intel, Osborne [59] found that reworked activities typically exhibited little further improvement curve effect after an initial iteration. Thus, we model the improvement curve for each activity as a step function, where an activity initially takes 100% of its duration and cost to accomplish, while second and subsequent executions of the activity take $x\%$ of the original duration and cost. Improvement curve data for the UCAV project are given in the last column of Table I.

F. Process Cost and Duration

Expected duration is the primary process metric and objective function used by most process models, including the vast majority of the models mentioned above. Quite a bit of work has been done to calculate and bound the completion times of activity networks [e.g., [29]].

Process cost is often treated deterministically or purely as a function of process duration. While cost outcomes may correlate to an extent with duration outcomes, cost is not a simple function of duration [38]. Few models compare a number of individual cost and schedule outcomes.

In industry, process cost estimates utilize either parametrics or "roll-up" techniques. Parametric techniques or *cost estimating relationships* (CERs) such as those in the Cocomo software [9] predict process cost as a function of several factors—e.g., duration, complexity, novelty, etc.—with coefficients determined from regression analysis of historical data. Roll-up or "grass-roots" techniques use a *cost breakdown structure* (CBS) related to the hierarchy of activities [e.g., [11] and [55]]. A cost estimate is made for the smallest activities in the breakdown, and these estimates are aggregated to arrive at a single figure for the process. The low-level cost estimates are typically based on historical costs and can be deterministic or random variables [39].

While providing a valuable perspective on process costs, parametric techniques are not easily integrated with process models. CERs cannot evaluate processes for which inadequate historical data exist, and they do not specifically account for the relationships among activities. The model in this paper uses a type of roll-up technique, with the important extension of accounting for activity interactions, since these are often the primary cost drivers [12].

G. Process Cost and Schedule Uncertainty and Risk

Cost and schedule uncertainty and risk are also important project management metrics. Point estimates of process cost and duration do not convey information about these metrics. PDFs provide additional information about the probabilities of various outcomes, thereby expressing a notion of uncertainty. Given a deadline, the area under the PDF to the right of the deadline represents the probability of unacceptable process durations. Substantial literature addresses the use of activity network models and Monte Carlo simulation to evaluate schedule uncertainty [e.g., [10], [27], [39], and [44]]. However, these methods typically equate risk with uncertainty and do not account for varied consequences. Each outcome has a consequence or impact. Big cost or schedule overruns can have big consequences!

The risk of an outcome is a function of both its probability and its impact

$$\mathcal{R}\ (\mathrm{Outcome}) = P(\mathrm{Outcome}) \cdot I(\mathrm{Outcome}) \tag{5}$$

[cf., (1)]. Thus, an impact function weights each adverse outcome by its consequence, yielding a risk factor, $\mathcal{R}$. Utility curves can be used as impact functions, thereby incorporating information about characteristics such as risk averseness into the risk quantification. In this paper, for example, we will use a simple quadratic impact function for schedule overruns

$$I_S = \kappa_S (S - T_S)^2, \quad S > T_S \tag{6}$$

where κ_S is a normalization constant, S is the duration outcome, and T_S is the schedule target or deadline. A quadratic impact function indicates that the consequence of a schedule overrun increases as the square of the size of the overrun, which makes

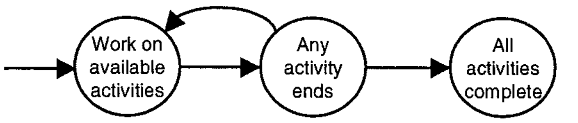


Fig. 6. Simulation event graph.

sense in situations where the "deadline" represents a rough expectation.[10] Alternatively, a customer utility curve for lead time or a step function at the deadline could be used as the impact function for a schedule overrun.

Total schedule risk is the sum of the probability and impact products over all adverse outcomes

$$\mathcal{R}_S = \kappa_S \int_{T_S}^{\infty} f(S_0)\,[S_0 - T_S]^2\, dS_0 \qquad (7)$$

where $f(S_0)$ is the PDF of schedule outcomes. A similar formula applies for cost risk $\mathcal{R}_C$[12]. Since PDFs can have different means, variances, and skewnesses—and these moments can be traded off—the risk factor provides a measure with which to compare PDFs against a common target.

III. The Simulation Algorithm

The model uses a discrete event simulation [51], [62] to compute the distributions of duration and cost for a given set of inputs. Each simulation run begins at system state 0 (initial values), with a randomly sampled[11] duration and cost (with correlation) for each activity. Initially, each activity has 100% of its work to do, as tracked by the "work to be done" vector, $\boldsymbol{W}$ [68]. When $\boldsymbol{W}_i = 1$, the entire activity i remains to be done; when $\boldsymbol{W}_i = 0$, activity i is complete.

In each system state, the model first determines the set of active activities, according to the work policy discussed in Section II-C. The shortest activity in the active set will be the next activity to finish and generate output, so its duration determines the time until the next event, "any activity ends." (Events are summarized in Fig. 6.) Once an activity ends, the event time is added to the cumulative duration, and the cost of the work done on all active activities is added to the cumulative cost. Appropriate fractions of $\boldsymbol{W}$ are decremented from the active activities, and the model checks for any rework (increments to $\boldsymbol{W}$) caused by the completed activity's output.

To determine rework, a probabilistic check is made for potential iterations (rework for upstream activities) and for second-order rework resulting from any such iterations, using the probabilities in $\boldsymbol{DSM}_{\text{xy1}}$. Again, if output from activity j causes rework for activity i, then the amount—a percentage of i, given in $\boldsymbol{DSM}_{ij2}$, and modified by the improvement curve—is added to $\boldsymbol{W}_i$. When all activities are complete, all $\boldsymbol{W}$ entries equal zero, and the simulation outputs cumulative cost and duration as C and S, respectively.[12] Tables II–IV summarize the model inputs, variables, and algorithm.

[10]Taguchi [75] highlighted the usefulness of quadratic quality loss functions.

[11]Monte Carlo simulation using the Latin Hypercube sampling technique [54]

[12]Simulation models of most realistic development situations converge quickly. However, it is possible to use a set of inputs which does not converge because tasks create large amounts of rework for one another. We have not found such unstable situations in any of our practical applications of the model.

TABLE II
Summary of Model and Simulation Inputs

Input Data	Unit of Measure
• List of activities comprising process	(List)
• Activity inputs and outputs	(Binary DSM)
• Activity sequence in DSM (representing process architecture)	(Sequence vector)
• Duration estimates (BCV, MLV, WCV)	Time (e.g., days)
• Cost estimates (BCV, MLV, WCV)	Money (e.g., dollars)
• Improvement curve benefit when repeating activity	% of original work
• Likelihood of a typical change in an output from one activity causing rework in another activity	Probability [0,1]
• Impact to an activity of a typical change in an input	% of activity to be reworked [0,1]
• *C* & *S* sampling correlation for each activity	Correlation [-1,1]
• Output distribution precision/stability criteria	% stability
• Run batch size (between stability checks)	Number of runs
• Cost target or budget	Money
• Schedule target or deadline	Time
• Cost overrun consequences as a function of amount of overrun (impact function)	Penalty/Money
• Schedule overrun consequences as a function of amount of overrun (impact function)	Penalty/Time

As the simulation is run many times, a number of C and S sample pairs are generated. The C outcomes form a cost distribution and the S outcomes form a duration distribution for the given process architecture. Together, the C, S pairs form a joint cost-duration (cost-schedule) distribution.

A number of runs are necessary to get stable distributions. Batches of runs, b, are done until both the means and variances of the C and S distributions stabilize to within precision α, as checked by the following equations (shown only for C):

$$\frac{|E[C_r] - E[C_{r-b}]|}{E[C_{r-b}]} < \alpha \qquad (8)$$

$$\frac{\left|\sigma^2_{C,r} - \sigma^2_{C,r-b}\right|}{\sigma^2_{C,r-b}} < \alpha \qquad (9)$$

where r is the number of simulation runs. Similar equations apply for S. We use $\alpha = 0.01$ and $b = 100$.

IV. Results

This section discusses analytical and simulation results from applying the model to the baseline architecture of the UCAV preliminary design process. The full data set and the data-gathering methodology are discussed in [12].

A. Interface Criticality

Without simulation, the model can be used to analyze the relative criticality of interfaces among activities. While the importance of knowing the most critical activities in a process has been documented [30], interface criticality is also important, since iteration is a major driver of process cost and schedule outcomes. Because rework in one activity can propagate rework in others, the presence of higher order terms makes a closed-form solution for interface criticality difficult to derive for a large

TABLE III
SUMMARY OF MODEL AND SIMULATION VARIABLES

r	Number of simulation runs
n	Number of process activities
C	Cumulative process cost for a given run
S	Cumulative process duration for a given run
t	Time until next event
b	Run batch size between output distribution stability checks
α	Required stability of output distributions (% change allowed between batches of runs)
$\boldsymbol{V}_n$	Activity sequencing vector: a vector of length n used to specify the process architecture
$\boldsymbol{ActS}_n$	Activity duration vector: a vector of length n containing a sampled duration for each activity
$\boldsymbol{ActC}_n$	Activity cost vector: a vector of length n containing a sampled cost for each activity, correlated with the duration sample in $\boldsymbol{ActS}$ according to $\boldsymbol{Corr}$
$\boldsymbol{Corr}_n$	Sampling correlation vector: a vector of length n containing each activity's duration to cost sampling correlation
$\boldsymbol{W}_n$	Work vector: a vector of length n with a [0,1] entry for each activity, indicating work to be done on that activity; initially set to all "1"s to indicate 100% of the work remains for each activity
$\boldsymbol{WN}_n$	"Work now" vector: a vector of length n with a Boolean entry for each activity, indicating the active activities for the current system state
$\boldsymbol{IC}_n$	Improvement curve vector: a vector of length n with a [0,1] entry for each activity, indicating the percentage of the original activity duration and cost required for second and successive executions of the activity
$\boldsymbol{DSM}_{nnk}$	DSM (off-diagonal portion of n x n matrix with third dimension k): dimension $k = 1$: rework probability [0,1] for superdiagonal entries where (row) $i < j$ (column), the probability of activity j causing rework for activity i; for subdiagonal entries where $i > j$, the probability of activity j causing second-order rework for activity i after rework of activity j dimension $k = 2$: rework impact [0,1] the added work (in terms of the increase in $\boldsymbol{W}$) should rework occur

TABLE IV
ALGORITHM FOR EACH RUN OF SIMULATION

```
1. Initialize model variables from the inputs at state 0.
2. Randomly sample duration and cost of each activity (with appropriate correlation, using Latin Hypercube
   sampling technique).
3. For the current system state:
   A. Determine the set of active activities, based on the work policy:
        Set all $WN = FALSE$.
        Find most upstream activity, $i$, with unfinished work—i.e., where $W(i) > 0$. Set $WN(i) = TRUE$.
        Loop through subsequent activities.
            If next activity has unfinished work AND is not dependent on an unfinished, upstream activity, then
                set its $WN$ entry to $TRUE$.
            Otherwise, the complete active set has been found (stop checking activities).
   B. Calculate the time until the next event (shortest active activity ends), $t$
   C. Work on active activities (decrement $W$); increment cumulative time by $t$; increment cumulative cost
   D. Check for rework generated by the completed activity $j$
        Look through column $j$ in $DSM_1$ (above activity $j$) for potential iterations. Evaluate each possibility versus
            a random number; if iteration occurs for activity $i$, then:
            Add appropriate rework (given in $DSM_{ij2}$, adjusted by $IC_i$) to $W_i$. But if $W_i > 1$ now (because activity
                $i$ was not finished), reduce $W_i$ to 0.9 to keep work from expanding beyond original scope and to
                represent some improvement.
            Look through column $i$ in $DSM_1$ (below activity $i$) for potential second-order rework.
            Evaluate each possibility (where $W < 1$) versus a random number; if rework occurs for activity $k$,
                then:
                Add appropriate rework (given in $DSM_{ki2}$, adjusted by $IC_k$) to $W_k$. But if $W_k > 1$ now (because
                    activity $k$ was not finished), reduce $W_k$ to 0.9 to keep work from expanding beyond original
                    scope and to represent some improvement.
4. If any activity has more work to do, then increment system state and repeat 3.
5. Output $C$ and $S$.
```

complex activity network [68], [69]. Using the factors in our model, the criticality of various interfaces to the schedule outcome could be calculated as follows:

$$\boldsymbol{Criticality}_{Sij} = [\boldsymbol{DSM}_{ij1} \cdot \boldsymbol{DSM}_{ij2} \cdot \boldsymbol{IC}_j \cdot \boldsymbol{ActS}_j] \cdot [\text{second-order rework effects}] \cdots \tag{10}$$

Disregarding the second- and higher order terms, and using MLVs for $\boldsymbol{ActS}$ (instead of treating it as a random variable), relative criticalities of the UCAV process interfaces are calculated and shown in Fig. 7. Assuming that the extent of rework propagation can be limited (especially for high visibility design changes), the approximation obtained by disregarding the higher order terms is nevertheless instructive.

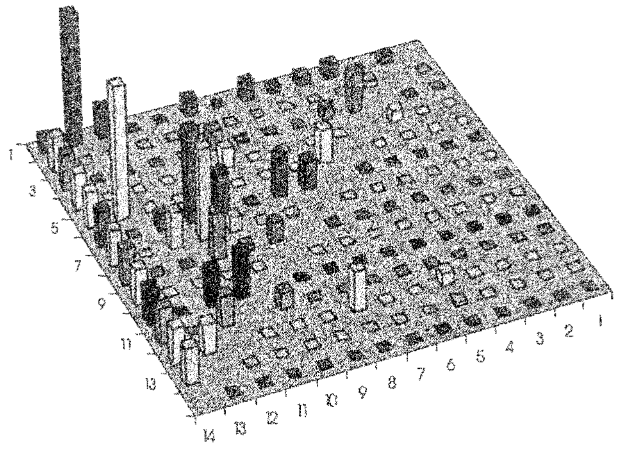


Fig. 7. Interface criticalities (to process duration).

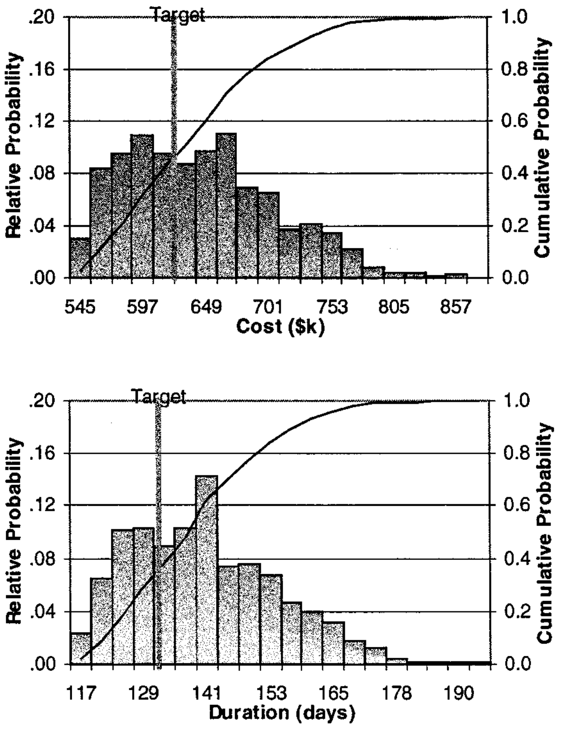


Fig. 8. Results: cost and duration PMFs and CDFs.

In practice, interface criticalities highlight potentially dangerous assumptions being made during process execution; these drive cost and schedule unpredictability and risk. Interface criticalities also draw focus to process failure modes and effects, thereby illuminating opportunities to improve activity robustness. Furthermore, interface criticalities can help determine the impacts of changes and delays coming from outside the process. Thus, they facilitate project replanning. Moreover, interface criticalities can provide a gradient for architecture optimization, as discussed in Section V-B.

B. Simulation Outputs

Fig. 8 shows *probability mass functions* (PMFs) and *cumulative distribution functions* (CDFs) of simulated cost and schedule outcomes ($r = 1100$). The mean cost outcome $E[C]$ is \$637k with a standard deviation σ_C of \$63k. $E[S] = 138$

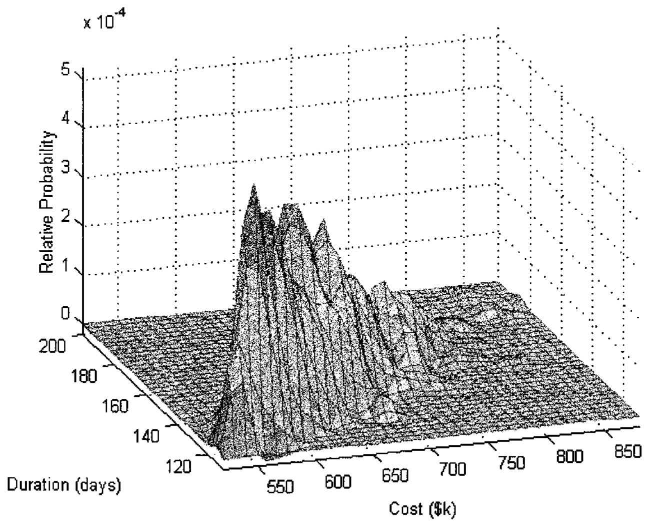


Fig. 9. Joint cost and duration PDF.

days with $\sigma_S = 14$ days. Not surprisingly, both distributions are skewed right ($\gamma_C = 0.58, \gamma_S = 0.55$). Fig. 8 also shows targets of \$630k for cost and 130 days for schedule. Fig. 9 shows the joint cost and schedule PDF resulting from paired cost and schedule outcomes.[13] The upper portion of Fig. 10 shows the joint PDF as a contour plot.

Traditional methods of analyzing the given activity network, ignoring iteration, yield $E[C] = \$615\text{k}$, $\sigma_C = \$55\text{k}$, $E[S] = 133$ days, and $\sigma_S = 13$ days. That is, each metric is underestimated versus when accounting for iteration.

The distributions and targets yield probabilities of overruns: $P_C(\text{overrun}) = 0.51$ and $P_S(\text{overrun}) = 0.67$. Using (7) and letting $\kappa_C = 1/(\$\text{k})^2$ and $\kappa_S = 1/(\text{days})^2$ (so $\mathcal{R}$ will be dimensionless), $\mathcal{R}_C = 2654$ and $\mathcal{R}_S = 255$. If κ is chosen to put $\mathcal{R}_C$ and $\mathcal{R}_S$ in comparable units such as lost dollars of revenue or lost customer utility, $\mathcal{R}_C$ can be compared to $\mathcal{R}_S$ to see which is contributing greater risk. For example, letting

$$\kappa_C = \frac{(\$1\text{k lost profit})}{(\$1\text{k of cost overrun})^2} \tag{11}$$

$$\kappa_S = \frac{(\$50\text{k lost profit})}{(\text{day of schedule overrun})^2} \tag{12}$$

then $\mathcal{R}_C = \$2{,}654\text{k}$ lost profit and $\mathcal{R}_S = \$12{,}775\text{k}$ lost profit. Given these values, schedule risk is a bigger concern than cost risk with the given process architecture, targets, and consequences of overruns.

C. Model Validity

The model has high face validity [51] because it is based on existing theory, extensive observations of the modeled system, and conversations with experts on the modeled system [12]. Furthermore, the model is applied to data from an industrial setting. While extending the validity of the model by applying it in experimental and real environments is a future goal, the current

[13]This PDF was created from the sample C, S pair data using bicubic interpolation across a 48×48 mesh grid from 20×20 histogram bins. We gratefully acknowledge the use of Matlab m-files written by Breivik and Keane in preparing the three-dimensional histogram data.

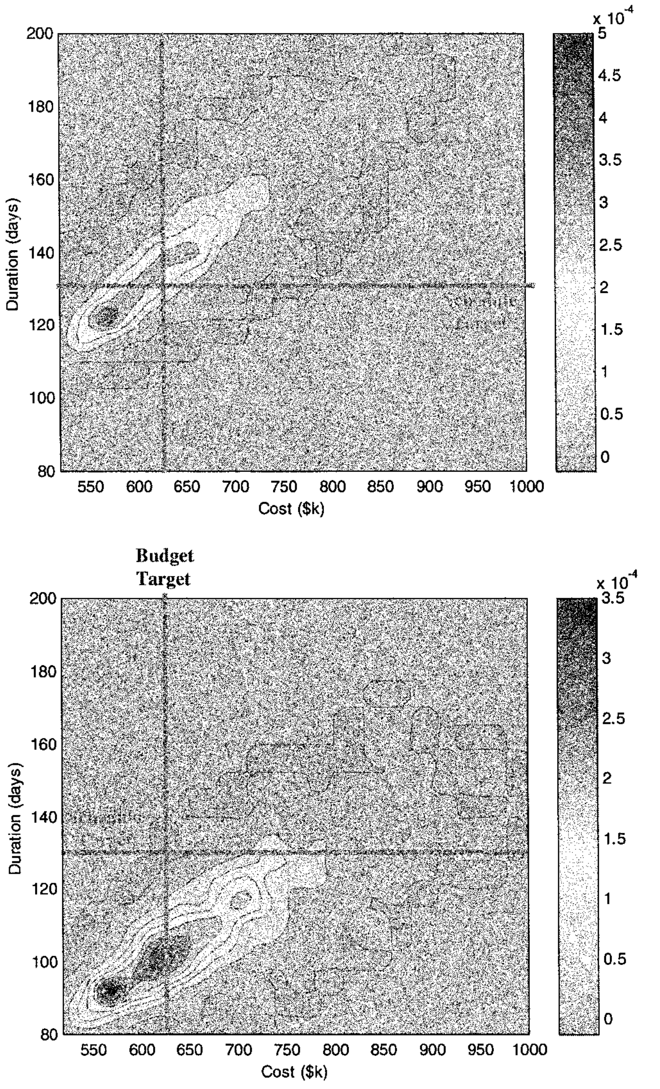


Fig. 10. Joint cost and duration PDF contour plots for (a) process architectures one and (b) three.

TABLE V
COMPARISON OF FIVE PROCESS ARCHITECTURES

	Architecture	1*	1	2	3	4	5
Model Inputs	V	1	1	1	1	1	1
		2	2	2	2	2	2
		3	3	3	3	3	3
		4	4	5	4	4	5
		5	5	6	5	5	8
		6	6	4	13	13	6
		7	7	8	6	6	4
		8	8	9	7	9	9
		9	9	7	8	7	7
		10	10	10	9	8	13
		11	11	11	10	10	10
		12	12	12	11	12	12
		13	13	13	12	11	11
		14	14	14	14	14	14
	T_C	630					
	T_S	130					
Model Outputs	r	1000	1100	1400	1200	1200	1300
	E[C]	615	637	634	660	681	677
	σ_C	55	63	62	72	84	81
	E[S]	133	138	144	108	95	97
	σ_S	13	14	15	14	14	14
	P_C(unacceptable)	39%	51%	49%	61%	70%	69%
	P_S(unacceptable)	55%	67%	78%	7%	2%	2%
	$\mathcal{R}_C$	1081	2654	2430	5253	9000	8138
	$\mathcal{R}_S$	135	255	411	6	2	3

*Without iteration

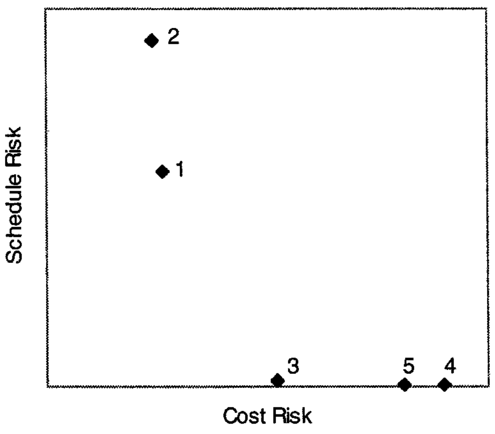


Fig. 11. Cost and schedule risk factors for five process architectures.

level of validity is comparable with models in the established literature [72].

D. Sensitivity Analyses

Some interesting analyses include the sensitivity of $\mathcal{R}_C$ and $\mathcal{R}_S$ to changes in: iteration probabilities, rework impacts, activity durations and costs [cf., [24]], process architecture, target, and impact function. The next two sections discuss some example applications along these lines.

V. The Impact of Process Architecture on $\mathcal{R}_C$ and $\mathcal{R}_S$

We used the simulation model to compare the relative levels of cost and schedule risk in five process architectures (with identical cost and schedule targets). Architecture one is the baseline (shown in Fig. 2). Architecture two was suggested by process engineers at the UCAV company. Architecture three takes advantage of beginning manufacturing analyses (activity 13) earlier, thereby allowing more activity overlap but *increasing* the first-order iteration in the process. (However, the activity's robustness to certain input changes and its large improvement curve reduce the impacts of any rework it undergoes). Architectures four and five are variants of architecture three. Table V shows comparative data for each of the five architectures.

Architectures three through five have more concurrency, more iteration, and higher cost risk, but much less schedule risk. Fig. 10 compares the joint C, S distributions for architectures one and three, clearly showing that architecture three reduces schedule risk at the expense of some cost risk. (In Fig. 10, the goal is to get the distribution in the lower left region, inside the budget and deadline "window.")

Fig. 11 plots $\mathcal{R}_C$ and $\mathcal{R}_S$ for each of the five architectures, revealing potential cost and schedule risk tradeoffs. As in Fig. 1, results for most of the possible process architectures will fall above and to the right of the five points. Perhaps some yet unfound architecture will stretch the tradeoff frontier to the lower left, as discussed in Section V-B. The impact function also influences the shape of the curve. For example, quadratic impact

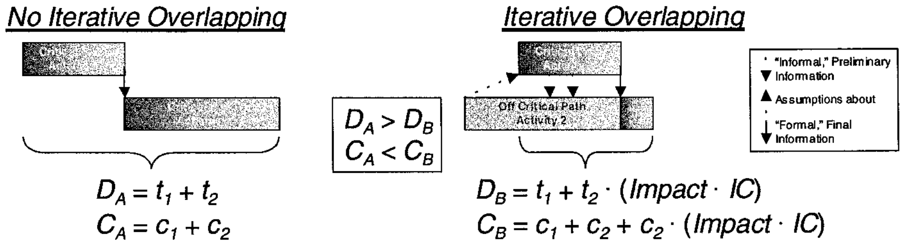

No Iterative Overlapping
Iterative Overlapping
Off Critical Path Activity 2
$D_A > D_B$
$C_A < C_B$
$D_A = t_1 + t_2$
$C_A = c_1 + c_2$
$D_B = t_1 + t_2 \cdot (Impact \cdot IC)$
$C_B = c_1 + c_2 + c_2 \cdot (Impact \cdot IC)$
"Informal," Preliminary Information
Assumptions about
"Formal," Final Information

Fig. 12. Effect of iterative overlapping on cost and duration.

functions reward a Δ improvement in a high-risk architecture over the same Δ improvement in a low-risk architecture.

A. Iterative Overlapping

More iteration can yield a *faster* schedule when it is possible to take advantage of appropriate iterative overlapping [49] or "preemptive iteration." For example, in architecture three, activity 13 is begun earlier and allowed to work concurrently with other activities, saving a large amount of time. Otherwise, it would be executed entirely on the critical path, as shown in Fig. 5. Even though this architecture *increases* the number of superdiagonal marks (and their distance from the diagonal) in the DSM, in this case the increased iteration is worth the price, which is discounted by small rework impacts and large improvement curve effects. Essentially, by beginning certain activities earlier, in parallel with other activities, designers are able to set up, "get their feet wet," and begin learning sooner. Then they can do some amount of rework, rather than their entire activity, on the critical path. Although the cost is slightly higher, the schedule is compressed appreciably [64].

Fig. 12 illustrates the tradeoff. In case A, two activities are done sequentially. Assuming that lengthening either activity will lengthen the overall process by the same amount, both activities are on the project's critical path. The total time and cost spent on both activities is simply the sum of each activity's individual time and cost. In case B, activity two starts earlier. When the results arrive from activity one, activity two must do rework, but the time required for this rework is less than the time required to do the entire activity. Thus, the total time on the critical path is reduced, although overall cost is increased by the cost of the rework. The model facilitates making intelligent choices about concurrency, providing an understanding of the implications for rework.[14] DSM optimization approaches that focus solely on the number of superdiagonal interfaces and their distance from the diagonal will miss these types of opportunities to suggest iterative overlapping.

B. Insights for Architecture Optimization

Although we compared several process architectures, we did not determine an optimal one. The model could be extended with a control loop that would suggest new architectures to test (changes to $\boldsymbol{V}$) using a genetic algorithm [cf., [65]]. If the change provides an improvement in the objective(s), it would be allowed to stand; otherwise, a new $\boldsymbol{V}$ mutation would be tried. This process would continue until an optimum was found (or until marginal improvement dropped and remained below a certain threshold for several comparisons, etc.).

We offer two insights for future efforts toward optimization. The first regards an appropriate improvement gradient to guide the optimization algorithm to quicker accurate convergence. The only real way to optimize the process architecture is to compare all possible sequences of $\boldsymbol{V}$. Yet this is computationally intensive for interesting numbers of activities, an $O(n!)$ operation. Thus, several authors [50], [69], [74] have offered heuristics that focus on reducing iteration, expecting that the optimal process architecture will have a minimal amount of iteration. However, our model disqualifies that assertion. Section IV-A discussed a new alternative for an improvement gradient, interface criticality.

Second, a major optimization challenge is determining an objective function rich enough to account for all of the significant influences on the process. The traditional objective is to minimize the expected duration of the process [65], [68]–[70]. However, minimizing variance[15] and risk are also worthwhile objectives, and we want to do so for both cost and duration. Hence, we suggest the risk factors as objective functions instead of expected durations and costs.

VI. Other Applications

This section presents some additional applications of the model, using architecture *one* of the UCAV process to demonstrate.

A. Project Planning: Setting Budgets and Deadlines

Project planners may face the task of choosing appropriate cost and schedule targets. As shown in (3), the risk factor is partly a function of the chosen target. When choosing or negotiating targets, one wants to ensure an acceptable level of risk. In Fig. 13, $P_C(\text{overrun})$ and $\mathcal{R}_C$ are plotted versus various possible targets (budgets). For example, choosing a cost target of \$600k implies about a 67% chance of an unacceptable outcome and a cost risk factor of about 4900. In comparison, letting $T_C = \$700\text{k}$ yields $P_C(\text{overrun}) \approx 0.16$ and $\mathcal{R}_C = 450$. These evaluations come in handy when negotiating cost and schedule targets, choosing an acceptable level of risk when planning projects, or when analyzing what premium a customer should pay for faster delivery, etc. The analysis can also help anticipate the risks of uncertain or changing targets.

[14]When knowledge of potential effects is high (i.e., the outputs are easily anticipated), it is good to "learn before doing" [61]. But when knowledge of effects is low (the outputs are hard to forecast), iterative overlapping is not recommended.

[15]While the expected value of and variance in duration correlate in some model outputs (e.g., [69]), minimizing one does not necessarily minimize the other; activities could be sequenced so as to minimize one at the expense of the other.

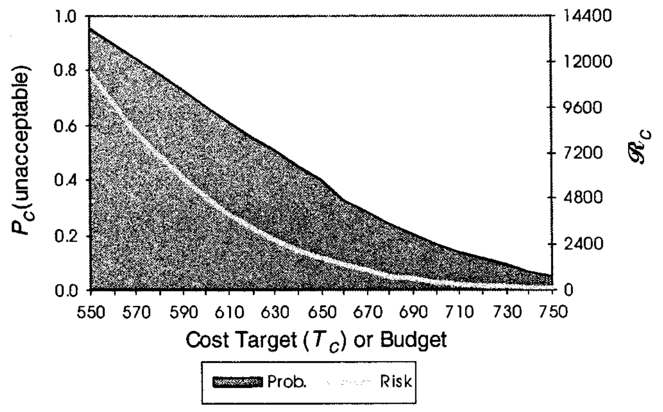


Fig. 13. P_C (unacceptable) and $\mathscr{R}_C$ for various cost targets (T_C) or budgets.

B. Project Management: Replanning

The simulation model can be reapplied once a process is underway. Activity cost and schedule estimates can be updated or replaced with actual values as they become available. Then, the model can be used to explore "what if" questions as they materialize. For example, if a low-probability high-impact iteration becomes more likely or occurs, affected activities can be identified quickly, and the impact on cost and schedule risk can be ascertained. The cost and schedule risks of adding new activities to the process—or of external changes affecting the process—can also be explored. The remaining portion of a process can then be replanned.

C. Process Improvement and Risk Reduction

By studying the sensitivities of the risk factors to individual activity costs and durations, iteration probabilities and impacts, and other factors, one can investigate the effects on the overall process of improving one or more of its activities or interfaces. For example, suppose new software tools are available that could decrease the time and cost of creating the initial UCAV structural geometry (Activity 5) by half—a face value savings of \$68k and 7.5 days for the individual activity. Rerunning the simulation results in a reduction of \$89k in expected project cost and 10 days in expected project duration, where the higher savings result from the compounding of the savings during rework. If the new tools cost \$80k, the individual activity may have difficulty justifying the investment, yet investment makes sense for the project as a whole (even more so for multiple projects). Significantly, this change in activity five reduces $\mathscr{R}_C$ to 77 and $\mathscr{R}_S$ to 79. The value of the risk reduction provided can also figure into the business case for the investment decision.[16]

Of course, the size of the improvement in overall process time, cost, and risk depends on the process architecture, which influences how many times the activity iterates, and therefore how many times its savings will be compounded.

VII. Next Steps and Possible Extensions

Extendibility is a significant strength of the model. For instance, the model provides a framework for an options-based approach [cf., [43]] with contingent activities. Interfaces (off-diagonal DSM entries), which in the basic model have attributes of probability and impact of rework, can have additional attributes such as a counter for the number of times particular iterations occur. Iteration probability could be modeled as a function of current C, S, $\mathscr{R}_C$, $\mathscr{R}_S$, level of design performance, or the iteration counter. Thus, the model could represent dynamic interfaces among activities.[17] As a real-time project replanning aid, the model could be extended to support decision making about the efficacy of additional design iterations.

The model can also be extended to account for resource constraints in the determination of active activities [85], using available algorithms (e.g., [8], [18], and [32]). This extension would further emphasize the advantages of iterative overlapping. Furthermore, the model can be used to explore alternative work policies, such as selective exchange of preliminary information [48], [52], [76], or using generalized precedence relations [33]. The model also provides an excellent tool for generating hypotheses for empirical research [20]. Finally, Goldratt discusses how schedule risk can be addressed by inserting a project buffer and feeder buffers [37]. Our model can be used to compute necessary buffer sizes in a more sophisticated manner than Goldratt's 50% heuristic. Some of these extensions are explored by Cho and Eppinger [23].

VIII. Conclusion

PD firms are keenly interested in improving the value they provide to customers, since that translates to increased profitability. In addition to product technical performance, two other aspects of customer value, product affordability and lead time, are directly affected by PD cost and duration [15]. Since adding activities to mitigate uncertainty in PD causes increased time and cost [12], [13], merely making the PD process more *predictable* can increase value. Moreover, customer value depends on *balancing* product technical performance, affordability, and timeliness in a way that the market or customer prefers. Balancing requires trading off not only product attributes but also process attributes such as cost and duration [15]. This paper has provided insights on how to navigate tradeoffs in process efficiency and predictability.

We have presented a rich model that provides practical insight into process architecture, cost, duration, uncertainty, and risk. The model accounts for a number of PD process characteristics, including interdependency, iteration, uncertain activity cost and duration, rework probability and impact, improvement curves, and work policy. The model is used to explore the effects of varying the process architecture. It provides valuable aid in project planning and replanning. The model is applicable at any level in a process hierarchy, and its results can be used as inputs to higher level models in a nested fashion. Analyzing the level of cost and schedule risk characteristic of a process architecture provides the capability to compare alternative work flows. With appropriate weighting factors, the model illuminates cost and schedule risk tradeoffs. Furthermore, the model shows the schedule advantage of iterative overlapping and accounts for the

[16]Section IV-B showed how to "dollarize" the risk factor units.

[17]Yet, all such contingencies and dynamic effects must still be outlined *a priori*, because the model can handle contingencies only on a case-by-case basis. This is reasonable since contingencies are usually studied through evaluation of a finite number of predetermined scenarios.

TABLE VI
MANAGERIAL INSIGHTS

- The cascading effects of rework have a major impact on process cost, duration, variance, and risk. Different process architectures have different information flow paths and amounts of rework. Much of the value and efficiency in PD processes is attributable to the process architecture, *regardless* of the value and efficiency of the individual activities, because the architecture accounts for the relationships between activities.
- Process cost, duration, variance (predictability), and risk can be traded off through manipulation of the process architecture.
- In general, seek to reduce iterations and their impacts. However, in certain cases, *iterative overlapping* is a means of decreasing process duration at the expense of process cost. Several small iterations can be better than a few large ones.
- Paying attention to the full deliverable flow among activities yields a more realistic picture of the PD process architecture than merely acknowledging precedence constraints.
- A few, critical activity interfaces—perhaps where assumptions are made by upstream activities about the results of downstream activities, or where verifications fail—account for much process rework and cost and schedule risk.
- Interface criticalities highlight significant *process failure modes and effects*, thereby pointing to opportunities to improve activity robustness. Many process designers ignore process failure modes, perhaps because noting them crowds a flowchart, or because they want to "plan to succeed." However, to really succeed, they should "plan for failure" (not "to fail")—i.e., process risk analysis and mitigation.
- Cost and duration targets (budgets and schedules) are a large determinant of process risk. When risk remains high despite process improvements, different targets may have to be negotiated.
- A process model provides a basis for situation visibility and discussion among process participants, enabling them to anticipate with greater fidelity the effects of assumptions and delays in one part of the process on other parts.

variables that influence its applicability. Overall, the model provides a framework in which to examine the impacts of a variety of effects on process cost, duration, and risk—yielding several important managerial capabilities and reinforcing significant managerial insights, such as those summarized in Table VI. Plus, the basic model is extensible toward providing additional realism, analyses, and insights.

In the future, as process capabilities are more widely recognized as significant sources of competitive advantage, *process* systems engineering (*vice* product systems engineering) will become more important. Organizations developing large, novel, complex systems will benefit especially from being able to convince their customers that their PD process has an acceptable level of risk. All PD organizations can benefit from low-risk process architectures by reducing the costs (of mitigating uncertainty) that they pass on to their customers.

ACKNOWLEDGMENT

The authors would like to thank D. E. Whitney and J. J. Deyst, Jr., who provided helpful comments on the model and applications. They also appreciate the comments of several anonymous reviewers. This paper updates work presented in [12, Ch. 6] and [17].

REFERENCES

[1] P. S. Adler, A. Mandelbaum, V. Nguyen, and E. Schwerer, "From project to process management: An empirically-based framework for analyzing product development time," *Mgt. Sci.*, vol. 41, no. 3, pp. 458–484, 1995.
[2] R. H. Ahmadi, T. A. Roemer, and R. H. Wang, "Structuring product development processes," *Eur. J. Oper. Res.*, vol. 130, pp. 539–558, 2001.
[3] R. H. Ahmadi and H. Wang, "Rationalizing product design development processes," in UCLA Anderson Graduate School of Management working paper, Los Angeles, 1994.
[4] F. AitSahlia, E. Johnson, and P. Will, "Is concurrent engineering always a sensible proposition?," *IEEE Trans. Eng. Mgt.*, vol. 42, pp. 166–170, June 1995.
[5] C. Alexander, *Notes on the Synthesis of Form*. Cambridge, MA: Harvard Univ. Press, 1964.
[6] J. Andersson, J. Pohl, and S. D. Eppinger, "A design process modeling approach incorporating nonlinear elements," in *Proc. ASME Tenth Int. Conf. Design Theory and Methodology*, Atlanta, GA, 1998.
[7] C. Y. Baldwin and K. B. Clark, *Design Rules: The Power of Modularity*. Cambridge, MA: MIT Press, 2000.
[8] U. Belhe and A. Kusiak, "Resource constrained scheduling of hierarchically structured design activity networks," *IEEE Trans. Eng. Mgt.*, vol. 42, pp. 150–158, June 1995.
[9] B. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
[10] B. W. Boehm, *Software Risk Management*. Washington, D.C.: IEEE Computer Society Press, 1989.
[11] J. A. Brimson, *Activity Accounting: An Activity-Based Costing Approach*. New York: Wiley, 1991.
[12] T. R. Browning, "Modeling and analyzing cost, schedule, and performance in complex system product development," Ph. D. dissertation, Massachusetts Inst. Technology, Cambridge, MA, 1998.
[13] ——, "Sources of schedule risk in complex system development," *Syst. Eng.*, vol. 2, no. 3, pp. 129–142, 1999.
[14] ——, "Applying the design structure matrix to system decomposition and integration problems: A review and new directions," *IEEE Trans. Eng. Mgt.*, vol. 48, pp. 292–306, Sept. 2001.
[15] ——, "On customer value and improvement in product development processes," *Syst. Eng.*, vol. 6, no. 1, 2003.
[16] ——, "Process integration using the design structure matrix," *Syst. Eng.*, vol. 5, no. 3, pp. 180–193, 2002.
[17] T. R. Browning and S. D. Eppinger, "A model for development project cost and schedule planning," in MIT Sloan School of Management working paper 4050, Cambridge, MA, Nov. 1998.
[18] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *Eur. J. Oper. Res.*, vol. 112, pp. 3–41, 1999.
[19] T. Burns and G. M. Stalker, *The Management of Innovation*. London, U.K.: Tavistock, 1961.
[20] K. M. Carley, "On generating hypotheses using computer simulations," *Syst. Eng.*, vol. 2, no. 2, pp. 69–77, 1999.

[21] M. Carrascosa, S. D. Eppinger, and D. E. Whitney, "Using the design structure matrix to estimate product development time," in *ASME Design Eng. Technical Conf. (Design Automation Conf.)*, Atlanta, GA, 1998.

[22] L. C. Cheung, R. P. Smith, and Z. B. Zabinsky, "Optimal scheduling in the engineering design process," in Univ. Washington working paper, Seattle, WA, Aug. 1998.

[23] S.-H. Cho and S. D. Eppinger, "Product development process modeling using advanced simulation," in *ASME 2001 Design Engineering Technical Conf. (DETC)*, Pittsburgh, PA, 2001.

[24] A. D. Christian, "Simulation of information flow in design," Ph.D. dissertation, Massachusetts Inst. Technology, Cambridge, MA, 1995.

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

[26] K. B. Clark and S. C. Wheelwright, *Managing New Product and Process Development*. New York: Free Press, 1993.

[27] E. H. Conrow and P. S. Shishido, "Implementing risk management on software intensive projects," *IEEE Software*, pp. 83–89, May/June 1997.

[28] K. G. Cooper, "The rework cycle: Why projects are mismanaged," *PMNETwork*, Feb. 1993.

[29] B. Dodin, "Bounding the project completion time distribution in PERT networks," *Oper. Res.*, vol. 33, no. 4, pp. 862–881, 1985.

[30] B. M. Dodin and S. E. Elmaghraby, "Approximating the criticality indices of the activities in PERT networks," *Mgt. Sci.*, vol. 31, no. 2, pp. 207–223, 1985.

[31] R. M. Eastman, "Engineering information release prior to final design freeze," *IEEE Trans. Eng. Mgt.*, vol. 27, pp. 37–41, June 1980.

[32] S. E. Elmaghraby, "Activity nets: A guided tour through some recent developments," *Eur. J. Oper. Res.*, vol. 82, pp. 383–408, 1995.

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

[34] S. D. Eppinger, M. V. Nukala, and D. E. Whitney, "Generalized models of design iteration using signal flow graphs," *Res. Eng. Design*, vol. 9, pp. 112–123, 1997.

[35] S. D. Eppinger, D. E. Whitney, R. P. Smith, and D. A. Gebala, "A model-based method for organizing tasks in product development," *Res. Eng. Design*, vol. 6, pp. 1–13, 1994.

[36] D. N. Ford and J. D. Sterman, "Dynamic modeling of product development processes," *Syst. Dynamics Rev.*, vol. 14, no. 1, pp. 31–68, 1998.

[37] E. M. Goldratt, *Critical Chain*. Great Barrington, MA: North River, 1997.

[38] S. B. Graves, "The time-cost tradeoff in research and development: A review," *Eng. Costs Production Econ.*, vol. 16, pp. 1–9, 1989.

[39] S. Grey, *Practical Risk Assessment for Project Management*. New York: Wiley, 1995.

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

[41] R. M. Henderson and K. B. Clark, "Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms," *Administrative Sci. Quart.*, vol. 35, pp. 9–30, 1990.

[42] G. M. Hoedemaker, J. D. Blackburn, and L. N. V. Wassenhove, "Limits to concurrency," in INSEAD working paper, Fontainebleau, France, Jan. 1995.

[43] A. Huchzermeier and C. H. Loch, "Project management under risk: Using the real options approach to evaluate flexibility in R&D," *Mgt. Sci.*, vol. 47, no. 1, pp. 85–101, 2001.

[44] D. T. Hulett, "Schedule risk analysis simplified," *PM Network*, pp. 23–30, July 1996.

[45] O. Isaksson, S. Keski-Seppälä, and S. D. Eppinger, "Evaluation of design process alternatives using signal flow graphs," *J. Eng. Design*, vol. 11, no. 3, pp. 211–224, 2000.

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

[47] S. J. Kline, "Innovation is not a linear process," *Res. Mgt.*, pp. 36–45, July–Aug 1985.

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

[49] ——, "Simplifying iterations in cross-functional design decision making," *J. Mechanical Design*, vol. 119, no. 4, pp. 485–493, 1997.

[50] A. Kusiak and J. Wang, "Efficient organizing of design activities," *Int. J. Production Res.*, vol. 31, no. 4, pp. 753–769, 1993.

[51] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, Second ed. New York: McGraw-Hill, 1991.

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

[53] C. D. McDaniel, "A linear systems framework for analyzing the automotive appearance design process," Master's dissertation, MIT, Cambridge, MA, 1996.

[54] M. D. McKay, R. J. Beckman, and W. J. Conover, "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.

[55] J. J. Moder, C. R. Phillips, and E. W. Davis, *Project Management With CPM, PERT and Precedence Diagramming*, Third ed. New York: Van Nostrand Reinhold, 1983.

[56] *NASA Syst. Eng. Handbook*, NASA Headquarters, Code FT, SP-6105, NASA, Washington, DC, 1995.

[57] K. Neumann, *Stochastic Project Networks: Temporal Analysis, Scheduling and Cost Minimization*. Berlin, Germany: Springer-Verlag, 1990, vol. 344.

[58] P. Nightingale, "The product-process-organization relationship in complex development projects," *Res. Policy*, vol. 29, pp. 913–930, 2000.

[59] S. M. Osborne, "Product development cycle time characterization through modeling of process iteration," Master's dissertation, Massachusetts Inst. Technology, Cambridge, MA, 1993.

[60] H. Petroski, *To Engineer is Human: The Role of Failure in Successful Design*. New York: St. Martin's, 1985.

[61] G. P. Pisano, *The Development Factory: Unlocking the Potential of Process Innovation*. Boston, MA: Harvard Business School, 1997.

[62] A. A. B. Pritsker, J. J. O'Reilly, and D. K. LaVal, *Simulation With Visual SLAM and AweSim*. New York: Wiley, 1997.

[63] A. A. B. Pritsker and C. E. Sigal, *Management Decision Making: A Network Simulation Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

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

[65] J. L. Rogers, *Integrating a Genetic Algorithm Into a Knowledge-Based System for Ordering Complex Design Processes—Technical Manual TM-110 247*. Hampton, VA: NASA, 1996.

[66] M. J. Safoutin and R. P. Smith, "The iterative component of design," in *IEEE Int. Eng. Management Conf.*, Vancouver, 1996, pp. 564–569.

[67] K. J. Singh, J. W. Erkes, J. Czechowski, J. W. Lewis, and M. G. Issac, "DICE approach for reducing product development cycle," in *Worldwide Passenger Car Conf. Exposition*, Dearborn, MI, 1992, pp. 141–150.

[68] R. P. Smith and S. D. Eppinger, "Identifying controlling features of engineering design iteration," *Mgt. Sci.*, vol. 43, no. 3, pp. 276–293, 1997.

[69] ——, "A predictive model of sequential iteration in engineering design," *Mgt. Sci.*, vol. 43, no. 8, pp. 1104–1120, 1997.

[70] ——, "Deciding between sequential and parallel tasks in engineering design," *Concurrent Eng.: Res. Applicat.*, vol. 6, no. 1, pp. 15–25, 1998.

[71] R. P. Smith and A. Leong, "An observational study of design team process: A comparison of student and professional engineers," *J. Mechanical Design*, vol. 120, pp. 636–642, 1998.

[72] R. P. Smith and J. A. Morrow, "Product development process modeling," *Design Studies*, vol. 20, no. 3, pp. 237–261, 1999.

[73] D. V. Steward, "The design structure system: A method for managing the design of complex systems," *IEEE Trans. Eng. Mgt.*, vol. 28, pp. 71–74, Sept. 1981.

[74] ——, *Systems Analysis and Management: Structure, Strategy, and Design* New York, PBI, 1981.

[75] G. Taguchi and Y. Wu, *Introduction to Off-Line Quality Control*. Nagoya, Japan: Central Japan Quality Assoc., 1980.

[76] C. Terwiesch and C. H. Loch, "Management of overlapping development activities: A framework for exchanging preliminary information," in INSEAD working paper #97/117/TM, Fontainebleau, France, Nov. 1997.

[77] ——, "Managing the process of engineering change orders: The case of the climate control system in automobile development," *J. Product Innovation Mgt.*, vol. 16, no. 2, pp. 160–172, 1999.

[78] S. H. Thomke, "The role of flexibility in the development of new products: An empirical study," *Res. Policy*, vol. 26, pp. 105–119, 1997.

[79] R. Verganti, "Leveraging on systematic learning to manage the early phases of product innovation projects," *R&D Mgt.*, vol. 27, no. 4, pp. 377–392, 1997.

[80] E. von Hippel, "Task partitioning: An innovation process variable," *Res. Policy*, vol. 19, pp. 407–418, 1990.

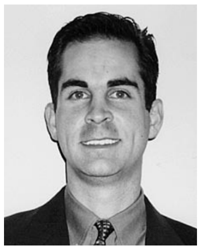
[81] D. E. Whitney, "Designing the design process," *Res. Eng. Design*, vol. 2, pp. 3–13, 1990.

[82] T. M. Williams, “Practical use of distributions in network analysis,” *J. Oper. Res. Soc.*, vol. 43, no. 3, pp. 265–270, 1992.

[83] P. M. Wolfe, E. B. Cochran, and W. J. Thompson, “A GERTS-based interactive computer system for analyzing project networks incorporating improvement curve concepts,” *AIIE Trans.*, vol. 12, no. 1, pp. 70–79, 1980.

[84] A. Yassine, D. Falkenburg, and K. Chelst, “Engineering design management: An information structure approach,” *Int. J. Production Res.*, vol. 37, no. 13, 1999.

[85] A. A. Yassine and T. R. Browning, “Analyzing multiple product development projects based on information and resource constraints,” in working paper, Cambridge, MA, 2001.



Tyson R. Browning received the B.S. degree in engineering physics from Abilene Christian University, Abilene, TX, and two S.M. degrees and the Ph.D. degree in technology, management and policy (systems engineering and operations management) from Massachusetts Institute of Technology, Cambridge.

He holds the position of Senior Project Manager in Integrated Company Operations at Lockheed Martin Aeronautics Company in Fort Worth, TX. He is the technical lead and chief integrator for a number of teams in developing the enterprise process architecture for the Aeronautics Company. He is also the lead author of company policies and processes driving the transition to a process-based company. He previously worked with the Product Development Focus Team of the Lean Aerospace Initiative at MIT, conducting research at Lockheed Martin, General Electric, Boeing, Raytheon, Sundstrand, and Daimler Chrysler. He has published papers on organizational integration, risk management, the design structure matrix, and process modeling.

Dr. Browning is a member of INCOSE, INFORMS, and AIAA.

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

He is currently the General Motors Professor of Management Science and Engineering Systems at the MIT Sloan School of Management. He serves as Co-Director of the Leaders for Manufacturing Program and the System Design and Management Program, both at MIT. His research deals with the management of complex engineering processes. He has published numerous articles and is co-author of the textbook entitled *Product Design and Development* (New York: McGraw-Hill, 1995 and 2000).

Dr. Eppinger is a member of INFORMS.