

Generalised Models of Design Iteration Using Signal Flow Graphs

Steven D. Eppinger, Murthy V. Nukala and Daniel E. Whitney

Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract. *Changing customer preferences, demand for quality, and new technologies have led to very short product life cycles. This requires firms to have short product development lead times while keeping product costs low and quality high in order to stay competitive. In this context, we focus on improved understanding of time-consuming design iterations. We are creating tools for modeling product development projects in order to predict the performance of product development organisations. In this paper, signal flow graphs are presented as a flexible tool for design process modeling, and illustrated using an industrial example. Analysis of the model allows computation of the probability distribution of lead time and identification of the key drivers of lead time.*

Keywords. Design iteration; Design process modeling; Product development lead time; Signal flow graphs

1. Introduction

One of the sources of sustainable competitive advantage in manufacturing-based industries is the capability to develop superior products. Today's technology-based companies are attempting to develop higher quality products faster than ever before. One of the major challenges faced in product development is to recognise and manage the iterative nature of the design process. In this paper we model design processes as discrete-time dynamic systems in order to represent and understand the iterative behaviour of project execution.

The paper is organised as follows: Section 1 motivates the iteration modeling problem and surveys related literature. Section 2 presents the use of signal flow graphs to analyse engineering design processes. Section 3 applies the method to a design process example from a US automotive manufacturer. Sec-

tion 4 develops two methods to gain further insight into the iteration process, sensitivity analysis and participation factors. Section 5 presents several model extensions which help to generalise the approach. Sections 6 and 7 offer discussion, conclusions, and scope for future work. The Appendix contains a tutorial description of signal flow graphs.

1.1. Design Iteration

Iteration is fundamental to the design process and its importance has been recognised by several authors [4–6, 14, 19–21]. Iteration can be defined as the repetition of activities to improve an evolving design. Smith and Eppinger [15, 16] explain that iterations occur for two reasons:

- The design fails to meet established criteria.
- New information is obtained since a prior iteration.

Osborne [12] found that iteration accounted for between one third and two thirds of total development time for projects at a major semiconductor producer. Osborne also found that unpredictable iteration is the main cause of variability in the lead times of projects at this firm. Standard project management techniques like PERT/CPM are not able to consider iteration or feedback, where tasks may have to be reworked. We are therefore motivated to develop new analytical techniques to model iteration and help understand iterative development processes.

1.2. Design Process Models

The view of product development as a modelable process and not as a unique craft has recently gained in popularity [18, 20, 21]. Common design process models can be grouped into two categories: performance evaluation models and optimisation models; the focus of our work is on the former. The key dimensions of design process performance modeling concern the physics of the work flow (how design

Correspondence and offprint requests to: Professor Steven D. Eppinger, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02142-1347, USA. E-mail: eppinger@mit.edu

activities are executed and repeated – iteration), the topology of the network representing the process (how activities are interconnected), and the analytical technique used (how the network is solved). A review of relevant literature is presented below.

Smith and Eppinger [16] present a sequential iteration model where coupled design tasks are executed one after the other, and rework is governed by a probabilistic rule. Repetition probabilities and task durations are assumed constant in time. The process is modeled as a Markov chain and the analysis can be used to compute lead time for the purely sequential case and to identify an optimal sequence of the coupled tasks to minimise iteration time. Ahmadi and Wang [2] extend the sequential iteration model by incorporating dynamic effects (iteration probabilities change with time) and learning (task durations change with iteration number). In another paper, Smith and Eppinger [15] present a parallel iteration model, where the coupled design tasks are all executed in parallel and iteration is governed by a linear rework rule. This model identifies the iteration drivers and the nature and rate of convergence of the process.

Adler et al. [1] model product development processes in a multi-project setting where there is competition for scarce resources and task queuing effects are significant. Ha and Porteus [7] model concurrent design as a dynamic programming problem, where the frequency of design reviews is optimised to shorten lead times and improve design quality. Krishnan et al. [9] present the characteristics of information dependencies between tasks and develop strategies to profitably overlap sequential tasks while avoiding adverse effects on product quality and development effort.

Each of the models presented in the literature only captures some of the effects evident in design processes in industry. The sequential iteration model [16] considers iteration explicitly using probabilities of rework and assuming constant task times. It attempts to reorder tasks to reduce lead time, which may not always be possible. The parallel iteration model [15] analyses the eigenstructure of the design system and identifies iteration drivers, but assumes complete parallelism in task execution. The two-activity models [7, 9] consider only the interface between upstream design and downstream engineering with uni-directional information transfer. This provides focus on the issues of using incomplete information and of detecting technical problems, thereby reducing development time by starting downstream tasks earlier. However, these two-activity models do not easily scale up to multi-task and bi-directional information

transfer scenarios. Queuing models [1] may neglect many of the characteristics of the iteration process. Hence we feel the need for a flexible modeling tool that can incorporate more of the effects found in complex projects and yield more valuable managerial insights. Signal flow graphs are presented here as such a tool.

1.3. Motivation for New Modeling Tools

Figure 1 shows a block diagram of the body panel die design process at a US automotive firm. While only a portion of the entire vehicle development process, stamping die development itself is a multi-stage activity involving hundreds of people and taking many thousands of engineer-hours to complete. The process is highly iterative because the panels need to be analysed and redesigned for manufacturability before prototype stamping dies are made. Furthermore, iterations involving prototype dies are very slow and must be avoided if possible. As a result, the activity flow of the project is somewhat uncertain, as are the activity durations. The problem is not one of sequencing, as it would be difficult to re-order the tasks to achieve superior performance. Rather, it is crucial to manage the timing of the tasks and the interactions between tasks in order to shorten lead times without adversely affecting quality.

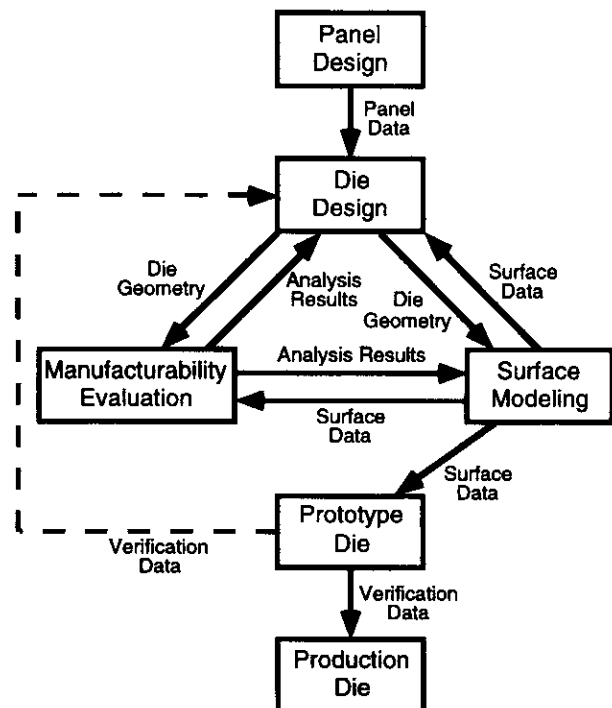


Fig. 1. Automotive stamping die design process.

2. Signal Flow Graphs

The signal flow graph is a well-known tool for circuit and systems analysis in electrical engineering and for modeling discrete event systems. It begins as a diagram of relationships among a number of variables. When these relationships are linear, the graph represents a system of simultaneous linear algebraic equations. The signal flow graph, as shown in Fig. 2, is composed of a network of directed *branches* which connect at the *nodes*. A branch jk , beginning at node j and terminating at node k , indicates its direction from j to k by an arrowhead on the branch. Each branch jk has associated with it a quantity known as the *branch transmission* P_{jk} .

For our modeling purposes, the branches represent the tasks being worked (an activity-on-arc representation). The branch transmissions include the probability and time to execute the task represented by the branch:

$$P_{jk} = p_{jk} z^{t_{jk}}$$

where p_{jk} is the probability associated with the branch, t_{jk} is the time taken to traverse the branch, and z is the transform variable used to connect the physical system (time domain) to the quantities used in the analysis (transform domain). The z transform simplifies the algebra, as it enables us to incorporate the quantities to be multiplied (probabilities) in the coefficient of the expression, and to include the quantities to be added (task times) in the exponent. The resulting system is then analogous to a discrete sampled data system, and the body of literature on this subject can be applied for the analysis thereof.

2.1. Significance of the Graph Transmission

The Appendix presents a brief tutorial on signal flow graphs and explains the absorption of nodes in a graph, an algebraic technique for graph simplification. The *path transmission* is defined as the product of all branch transmissions along a single path. The graph transmission is the sum of the path transmis-

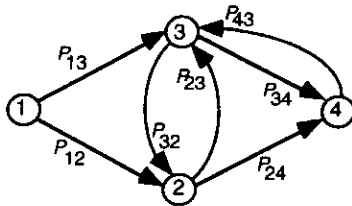


Fig. 2. A linear signal flow graph.

sions of all the possible paths between two given nodes. (When there are cycles in the system due to iteration, the number of paths is infinite.) The graph transmission is also the resulting expression on an arc connecting the two given nodes when all of the other nodes have been absorbed. In particular, we are interested in computing the graph transmission from the start to finish nodes. Henceforth, *graph transmission* shall refer to the graph transmission between the start and the finish nodes, and is denoted by T_{sf} .

The coefficient of each term in the graph transmission is the probability associated with the path(s) it represents, and the exponent of z is the duration associated with the path(s). The graph transmission can be derived using the standard operations for signal flow graphs [8, 17], which are summarised in the Appendix. The impulse response of the graph transmission is then a function representing the probability distribution of the lead time of the process.

Since the graph transmission is the moment-generating function of the lead time, it can be differentiated to obtain the expected value of the lead time:

$$E[L] = \left. \frac{dT_{sf}}{dz} \right|_{z=1}$$

This result is possible because each term of T_{sf} is of the form $p_i z^{t_i}$, where t_i is the time associated with the path(s) represented by the term, and p_i is the probability. When differentiated, this term becomes $p_i t_i z^{t_i-1}$. Evaluation at $z = 1$ yields $p_i t_i$ and summed over all terms (paths), this becomes the expected value of the lead time of the process. Similar arguments lead to the variance of lead time:

$$\text{Var}[L] = \left. \frac{d \left(z \frac{dT_{sf}}{dz} \right)}{dz} \right|_{z=1} - \left\{ \left. \frac{dT_{sf}}{dz} \right|_{z=1} \right\}^2$$

2.2. Determinant of the Flow Graph

If we assume unit task times, the signal flow graph becomes equivalent to the set of linear equations $\mathbf{x} = \mathbf{P}\mathbf{x}$, where \mathbf{x} is the node state vector, and \mathbf{P} is the signal flow graph matrix comprising the branch probabilities p_{jk} . The determinant of the flow graph is the expression which appears in the denominator of the graph transmission expression. Mason and Zimmerman [11] explain some interesting properties of the graph determinant. The roots of the graph

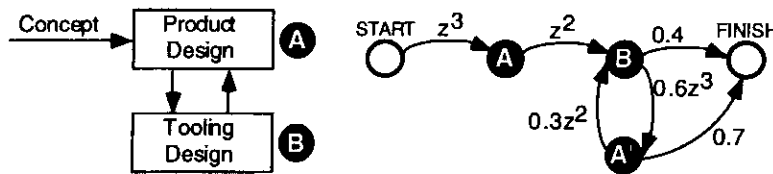


Fig. 3a. Simplified design process and signal flow graph.

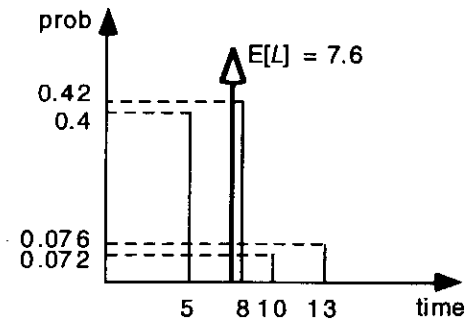


Fig. 3b. Derived probability distribution of lead times.

determinant are the poles of the system which govern its natural response to external inputs. The characteristic equation is

$$\det[\mathbf{I} - \mathbf{P}z] = 0$$

The graph determinant will be useful in analysing the eigenstructure of the system, from which we will compute the participation factors.

2.3. Numerical Example

A simple example is shown in Fig. 3. The hypothetical design process is represented by the graph shown in Fig. 3a. The two tasks A and B (product design and tooling design) take 3 and 2 units of time, respectively. Once B is attempted, A is reworked with probability 0.6, and once A is attempted, B is reworked with probability 0.3. Iterative repetitions of A are represented by the dummy A'.

The graph transmission may be found using the node elimination techniques explained in the Appendix or by writing the set of node equations and eliminating the intermediate variables. This graph transmission is given by

$$T_{sf} = \frac{z^5(0.4 + 0.42z^3)}{1 - 0.18z^5}$$

The expected value of the project lead time $E[L]$ is 7.6 units of time, which is computed using the formula given above.

The first few terms of the probability distribution function of lead times are represented graphically in Fig. 3b. The distribution can be found for this simple

example by performing synthetic division on T_{sf} to obtain the first few terms of the infinite series. The nominal (once through) time for A and B in series is 5 units of time, which occurs with probability 0.4. It is more likely (probability 0.42) that the lead time L of the process will be 8 units of time.

3. Application of Signal Flow Graphs to Design Process Modeling

An industrial example is now presented and analysed to illustrate use of the model, and to motivate the need for further extensions. Referring to the block diagram in Fig. 1 depicting the body panel die design process, we focus our attention on the highly iterative portion of this design process which precedes the construction of prototype and production stamping dies. Acceleration of this development process is of tremendous importance because die design and manufacturing lie on the critical path of the automotive development process. Significant delays in this process occur when manufacturability problems are not discovered until the die production phase. Therefore, the panel design needs to be evaluated for manufacturability as soon as possible in order to provide early feedback to the product and die designers.

The actual die development process is extremely complex, however, a simplified description will suffice here. The die development group receives input information from the panel designers. The primary iterative loop involves three activities: die design, manufacturability evaluation, and surface modeling. The objective of this three-part process is to identify

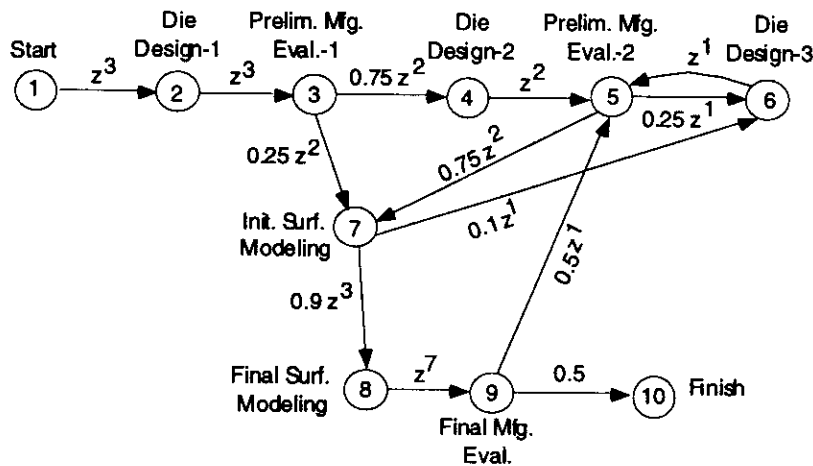


Fig. 4a. Signal flow graph representation of the die design process.

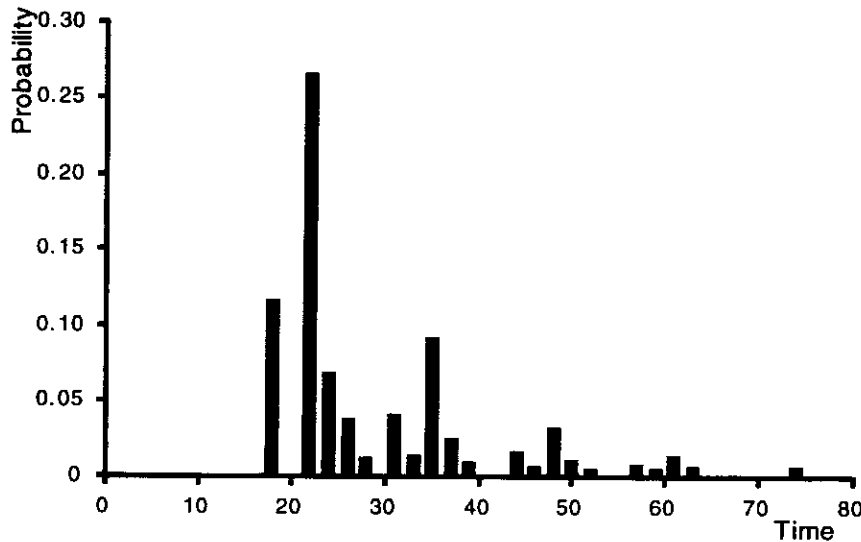


Fig. 4b. Computed distribution of lead times.

manufacturability problems well before the die development is completed so that necessary changes can be made to the panel design and/or to the process design. Each stage may be repeated a number of times, and when the die design is completed, prototype and production dies are then built.

With much assistance from engineers and managers at the study company, we were able to characterise the iterative die development process in terms of activity durations and transition probabilities. We found it helpful to draw iteration process maps [12], with which the necessary design process information could be collected in terms of first-pass times, repeat times, and probabilities of various situations. (Note that the timing data presented here have been scaled in order to conceal the actual development process timing at our study company.)

The signal flow graph of this process is shown in Fig. 4a, where the times shown (the exponents of the branch transmissions) are in scaled work days

required for a certain range of panel complexity. The signal flow graph shows the tasks of manufacturability evaluation and die design replicated since the rework probabilities and task times change after the first iteration. For example, die design is represented three times: as tasks 2, 4, and 6, with 3, 2, and 1 day durations. Similarly, manufacturability evaluation occurs in three places: as tasks 3 and 5, taking 3 and 2 days for the preliminary manufacturability evaluation; and as task 9, taking 7 days for the final evaluation. The probabilities are also interpreted easily from the graph: for example, there is a 0.75 likelihood that after the second round of preliminary manufacturability evaluation (task 5) the process continues with surface modeling (task 7), and a 0.25 chance that a third round of die design is done (task 6).

Inspection of the signal flow graph reveals that the shortest possible completion time is 18 days, in which case the preliminary manufacturability evaluation is

only performed once. The most likely completion time is 22 days, where the preliminary evaluation is done twice. Using the analytical method introduced above, the expected value of the lead time of the design process is found to be 36.2 days, with standard deviation of 22.0 days. The computed distribution is given in Fig. 4b.

To compare the model results with actual projects, lead time data for various panel types were collected from the die development group at our study company. Panels vary tremendously in complexity, so we focused our attention on the panel types representative of the process modeled. Data were collected for these panel types across a number of car platforms. The aggregate lead time for the process modeled was found to be 61.8 days, and the standard deviation of this lead time was found to be 19.8 days. (Note that these timing data have been scaled for comparison by the same factor used in the model, again so as not to reveal the actual aggregate panel lead time experienced at the study company.)

The model predicts the process lead time with disappointing accuracy (41% low). Possible sources of error include incorrect modeling assumptions and imprecise data collection methods. Indeed there are several effects in real projects that are not represented in this model, such as parallel activity flow and delay time (work queueing). The task timing and probabilities of iteration were inferred from interviews with engineers working on the project, and it is possible that their time estimates were optimistic and/or they underestimated the occurrence of iteration. These are all potentially significant sources of error.

4. Analysis of Sensitivities and Participation Factors

Two methods are now presented to gain further insight into the portions of the process that determine the lead time and its variations. First we present sensitivity analysis and then the computation of participation factors.

4.1. Sensitivity Analysis

The expected value and variance of the lead time of the design process are directly dependent on the probabilities of iteration and the task times. The sensitivity of the expected value and variance of lead time to each parameter can be calculated as the change in value of the quantity in response to a small change in the value of the parameter. If L represents the lead time of the process and k a parameter on

Task	3	4	5	6	7	8	9
2	0.08						
3		0.04			0.01		
4			0.04				
5				0.02	0.11		
6			0.02				
7				0.01		0.16	
8							0.38
9			0.03				

Fig. 5. Lead time sensitivities to variations in task times.

Task	3	4	5	6	7	8	9
3		0.10			-0.03		
4							
5				0.05	-0.15		
6							
7				0.03		-0.28	
8							
9			0.79				

Fig. 6. Lead time sensitivities to variations in branch probabilities.

which it depends, the sensitivity of L to changes in k , denoted S_k^L is given by

$$S_k^L = \frac{\Delta E[L]/E[L]}{\Delta k/k}$$

Calculation of the sensitivity of the expected value of lead time to changes in task times yields the lead time sensitivity matrix in Fig. 5. The j, k term of the matrix is the sensitivity of the lead time $E[L]$ to changes in t_{jk} . Similarly, calculation of the sensitivity of the expected value of lead time to changes to branch probabilities yields the lead time sensitivity matrix in Fig. 6. The j, k term of this matrix is the sensitivity of the lead time $E[L]$ to changes in the probability p_{jk} . (In computing these latter sensitivities, the sum of probabilities on branches emanating from a node were constrained to 1.0, which assures pure sequential iteration.)

The expected value of lead time is found to have positive sensitivity to small changes in each of the task times, while there are both positive and negative sensitivities to small changes in the branch probabilities. The negative sensitivities identify the probabilities on branches 3-7, 5-7, and 7-8 which if increased would avoid additional iterations and reduce expected lead time. Sensitivities to all of the other probabilities, and of course all of the durations, have the opposite sign and we would desire to decrease these parameters to reduce iteration time.

The highest sensitivity is to the probability on branch 9-5, leading to iterations back to preliminary manufacturability evaluation after surface modeling and final manufacturability evaluation have already been done. This high sensitivity can be explained by the large duration (13 days) of the rework loop initiated by this arc. This suggests that the model error of 41% could possibly be due to errors in a few probabilities. For example, with a sensitivity of 0.79, underestimation of approximately 50% in the probability of iteration (from task 9 to task 5) would cause a model error of 41%.

The sensitivity analysis confirms a general insight that the results of tasks that divert the project flow from one path to another will have the greatest impact on the overall project time. This effect is detrimental when the diversion is from a short path of completion to a long path of iteration. In our example, these are tasks 3, 5, 7, and 9, which involve preliminary and final manufacturability evaluation and initial surface modeling. This insight makes intuitive sense as these are tasks where verification is performed and problems of various types are discovered. In particular, the results of final manufacturability evaluation (task 9) have the greatest effect on lead time. Note also that there are three interacting loops, consisting of tasks 5-7-6, tasks 5-7-8-9, and tasks 5-6.

4.2. Calculation of Participation Factors Assuming Unit Task Times

An alternative analysis yields similar insights about the driving factors of the iteration process. In particular, we can identify the dominant design modes. A design mode is defined as a group of design tasks that are very closely related, and working on any one of them creates significant work, directly or indirectly, for each of the other tasks within the mode [15]. The smallest poles of the system are associated with the slowest design modes and dominate convergence of the iteration process. Identification of these modes can help focus engineering and management attention on the most important interactions in a design project. This can be accomplished by calculating the *participation factors* of the system.

The concept of participation factors is well developed in linear system theory [10, 13]. Assuming unit task times in order to create the signal flow matrix \mathbf{P} , substituting in the characteristic equation for the transform variable, $s = 1/z$, the eigenvalues of the system then are the roots of

Task	2	3	4	5	6	7	8	9
2	0	1	0	0	0	0	0	0
3	0	0	0.75	0	0	0.25	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	0.25	0.75	0	0
6	0	0	0	1	0	0	0	0
7	0	0	0	0	0.1	0	1	0
8	0	0	0	0	0	0	0	0.9
9	0	0	0	0.5	0	0	0	0

Fig. 7. Signal flow graph matrix \mathbf{P} of the design process.

	1st	2nd	3rd	4th	5th	6th and 7th	8th
Eigenvalues	1.00	1.00	1.00	0.121	1.816	1.032 ± 0.685i	1.00
Eigenvectors							
Task 2	1	0	0	0	0	0	0
Task 3	0	1	0	0	0	0	0
Task 4	0	0	1	0	0	0	0
Task 5	0	0	0	0.31	0.30	-0.19 ± 0.01i	0
Task 6	0	0	0	0.13	0.07	-0.10 ± 0.05i	0
Task 7	0	0	0	0.21	0.18	0.30 ± 0.03i	0
Task 8	0	0	0	0.17	0.23	-0.30 ± 0.02i	0
Task 9	0	0	0	0.17	0.23	0.30 ± 0.04i	1

Fig. 8. Participation factor matrix.

$$\det[s\mathbf{I} - \mathbf{P}] = 0$$

in which the largest eigenvalues correspond to the smallest poles. For each eigenvalue, we can then calculate the participation factors by computing the termwise product of the corresponding left and right eigenvectors, as shown by Pérez-Arriaga et al. [13]. The largest participation factors correspond to the components of the system contributing to the slowest modes. The analysis is similar to that demonstrated by Smith and Eppinger [15] for a parallel iteration model.

The signal flow matrix \mathbf{P} for the example die design process of Fig. 4a (assuming unit task times and without the start and finish nodes) is shown in Fig. 7. The eigenvalues of the matrix \mathbf{P} are given in Fig. 8. The corresponding participation factors obtained by the termwise multiplication of the right and the left eigenvectors are also shown in Fig. 8. Looking at the column of the participation factors matrix corresponding to the three largest eigenvalues (the 5th, 6th, and 7th), we see that these slowest modes of the system are affected by the tasks 5, 7, 8, and 9. This would indicate that the loop involving the preliminary manufacturability evaluation, surface modeling, and the final evaluation dominates the convergence of the design process, and needs to be

managed effectively to reduce the lead time of the process. Refer to Smith and Eppinger [15] for a detailed explanation of eigenstructure interpretation for design processes.

4.3. Extension for Participation Factors with General Integer Task Times

The assumption of unit task times distorts the behaviour of the system in the participation factor analysis. This is because assuming unit delays, the lead time of a particular path is equal to the number of branches traversed, and does not reflect the task times. However, for a more accurate result, we can introduce $n-1$ dummy states on each arc of the signal flow graph, where n is the integer task time associated with the branch. In the case of non-integer task times, they are all approximated as integers, which can be done either by scaling or by rounding, depending on the situation. The transmission of the first in each set of dummy arcs is assigned a coefficient equal to the probability of traversing that branch, and unit delay. Each of the other dummy arcs is given unit delay and unit probability. In the expanded graph, each task then is represented by multiple nodes. Once this transformation has been made, the above analysis may be directly applied. The eigenstructure analysis of the expanded system would yield the states and tasks most strongly affecting lead time.

5. Model Extensions

The model allows flexibility along a number of additional dimensions. We first discuss non-deterministic durations and rework probabilities, and then present an extension to handle parallel task structures.

5.1. Non-Deterministic Durations and Probabilities

The case where the magnitude of work is not deterministic, but distributed according to some discrete

probabilistic rule, can be easily handled as illustrated in Fig. 9. Here the nodes B, B' and B'' are not representing different tasks, but rather instances of different amounts of work or rework. This phenomenon is typical, for example, of validation situations where the magnitude of redesign work depends on the type of problem detected.

Another possible extension is to handle cases where the rework probabilities and task durations change dynamically. This is a more difficult problem which Ahmadi and Wang [2] have addressed by expanding the state space and restricting the number of iterations. The models presented here can handle the case where the first few iterations exhibit changing conditions, after which the parameters remain constant. This is also done using state-space expansion, however, the number of iterations is not restricted. We believe this to be a reasonable approximation in many design situations because it is typical that the duration of a task will not reduce infinitely with increase in iteration number, but settle at some minimum rework time after a few iterations.

5.2. Parallel Task Execution

Extension of the model to incorporate parallel task structures requires the following modification of the definition of a state in the probabilistic model: after a given state is reached (a task has been completed), all the tasks dependent on this task for information are now attempted based on an individual probabilistic rule. The probability of traversing a branch is governed only by the coupling between the two tasks involved and the stage of the process, rather than by a probability distribution over all tasks as in the sequential iteration model. Figure 10 depicts such a situation, where the individual probabilities are now restricted to be less than or equal to 1, but the sum of the probabilities is not constrained.

In Fig. 10, the two systems represented are identical in an analytical sense. If the sum of probabilities p and q is restricted to 1, then the graph on the left represents a sequential iteration process. In this case,

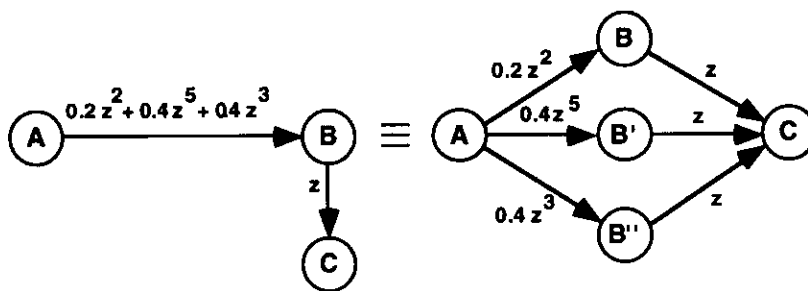


Fig. 9. Probability distribution over rework.

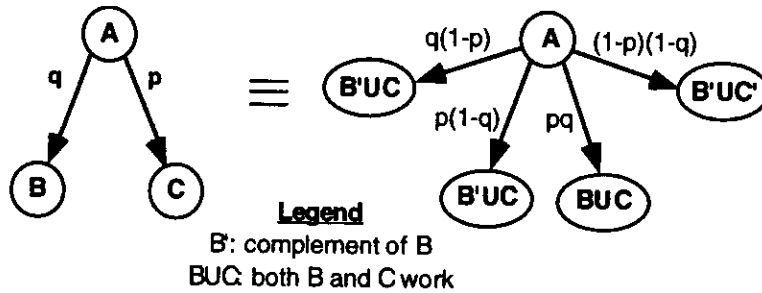


Fig. 10. Redefinition of state for parallel task execution.

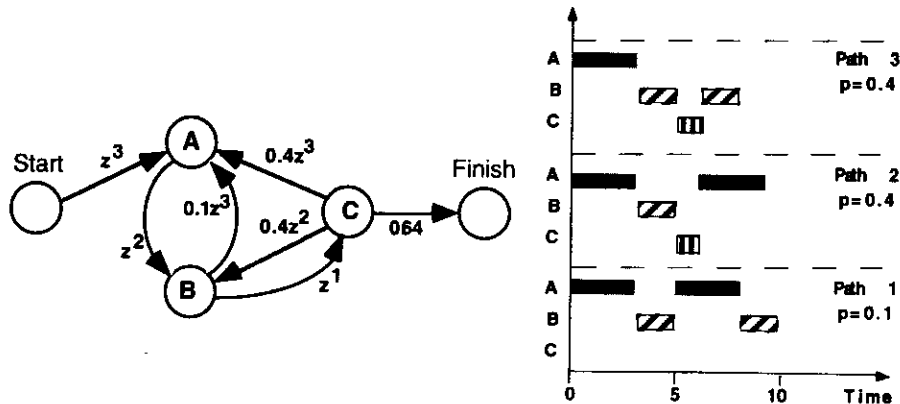


Fig. 11. Timing of parallel design activities.

the graph on the right would also yield the same results, though the physical interpretation is different as we allow multiple tasks to work at the same time. When the sum of probabilities p and q is not restricted, the graph on the left no longer represents a sequential process, but the analogous representation on the right can be used to model parallelism in a probabilistic sense.

With the new definition of a state, any number of tasks across different paths can now be working at the same instant in time, with a probability associated with each one. Figure 11 illustrates how this situation arises in the parallel case. Three of the infinity of paths possible from start to finish are shown on a timeline. Task A is first attempted and takes 3 units of time. Task B is then attempted and takes 2 units of time. At this point, task C is attempted with probability 1, and task A is reworked with probability 0.1. Hence there is a probability of 0.1 that both A and C are working at the same time, i.e., that from time 5 to time 6, both tasks A and C are active. Similarly, in a complex system, there is a probability associated with the event that any subset of all the tasks that could be working at that point in time are active.

The individual paths concerned each have a path transmission expression, as such the sum of the path transmissions of all the possible paths gives the graph

transmission. Hence the same method of analysis outlined above still applies to the parallel activity situation, as all the paths possible from start to finish are still captured in the transmission, and each path is associated with a lead time and a probability. Since multiple paths can be active, it is a probabilistic representation of parallelism, with the calculations taking an expected value form. The key difference is that when the signal flow graph is now solved for the graph transmission expression between start and finish, and the operation to calculate lead times performed, the resulting quantity is the total *effort* expended during the design process, which, due to parallelism, is greater than the lead time.

For our die design process example, we learned that after initial surface modeling (task 7), final surface modeling (task 8) is always begun in parallel with the possible repetition of die design (task 6). This changes the probability p_{78} from 0.9 to 1.0 in Fig. 4a. The resulting effort of the process becomes 48.4 days with a standard deviation of 17.4 days. Note that the parallel structure increases work as multiple activities are now executed simultaneously. This is due to the nature of the model change which increases the work to be done in the process. This expected duration is not directly comparable to the observed lead time since in the parallel model the quantity computed is

the total effort (engineering days) rather than the lead time.

6. Discussion

This section provides some discussion of the assumptions and limitations of the model, challenges of data collection, managerial insights available through application of the model, and directions for further work.

6.1. Assumptions and Limitations

The design process modeling method presented here relies on several assumptions, the accuracy of which determine the fidelity of the analysis. The primary assumption of the signal flow graph model is that process flow is determined by a probabilistic rule, with probabilities assigned to the likelihood that tasks will be executed or repeated in order to resolve conflicts. It is further assumed that the durations of the design process tasks are deterministic or distributed according to a discrete probabilistic rule, and that there are no resource constraints (queueing of work) and no delays due to causes such as the inefficiency of information transfers. The assumption of integer task times for the participation factor analysis is not very restrictive, as the sensitivity analysis confirms.

Extensions to the basic model are helpful for depicting certain design situations. Multiple tasks can be allowed to work at the same time, as explained in the section on parallelism. The case of design process parameters such as iteration probabilities and task durations changing dynamically in time or with iteration number can be handled by expanding the state space. This approach is tractable in situations where conditions change over the first few iterations and then remain constant, but is not reasonable when conditions change with every iteration, since the number of states in the expanded graph representing the system then increases continuously.

The logical relationship at the exit from any node is an OR relationship. This has several implications. One is that the decision rule determining the subset of tasks to work after the completion of a given task is implemented after the first instance of the state being reached. This means that AND rules, where a number of tasks have to be completed before a task starts, cannot be modeled. This is certainly a drawback of the model; however, the model is quite reasonable in situations where preliminary information is trans-

ferred upon availability and changes in this information and additional information are incorporated into the downstream tasks. Belhe and Kusiak [3] have addressed the problem of considering different logical relationships in design processes by enumerating the various possibilities and analysing each one. This approach is similar in complexity to the state space expansion proposed to handle some of the above problem structures.

An implication of the parallel model is that it is possible for a task to work and be caused to rework at the same instant in time. This is a situation where a task is active at the same point in time in more than one of the active paths. This is another potential drawback of the model; however, we have observed in some industrial situations that changes are requested of a task before the task completes working, so rework iterations and first iterations are not easily distinguishable. In such vague situations, we believe the model is reasonable.

Collection of project data about design activity durations and interactions modeled as probabilities proved to be difficult in practice, as could be expected. However, it was possible for the engineers to assign times for the first and subsequent repetitions of each activity, and to estimate the probability of each type of iteration based upon the known rework types. We would expect that it would be more difficult to obtain reliable information for a novel or less familiar development process.

In applying the model, we found that queueing effects were significant. In some cases the queueing delays were even greater than the task durations, particularly in cases where the engineers were working on several development jobs at once. Such delays may account for a large portion of the discrepancy between the average lead time predicted by the model and the actual mean lead time for the process modeled by our example. Queueing delays can be modeled in a number of ways. An average wait time can be added to any branch simply by increasing its duration; however, this does not represent the variance of such queueing delays. Alternatively, a self loop can be added (see the Appendix) to represent in a more stochastic fashion the effect of work queueing at a given task.

Computation of the expected value and variance of lead times involves differentiation of a polynomial. The computation is efficient for small problems, and took about 2 min (on a DEC 5000 workstation) for each computation of our 10-task case. However, the computation time would grow quickly as the number of tasks increases. For large models, numerical

approximation techniques can be used to calculate the moments of lead time.

6.2. Engineering Management Insights

The model can be used by engineering managers to gain insight into the process through lead time computation and sensitivity analysis, and by modeling a variety of scenarios, in the following ways:

- (a) *Evaluation of alternative project structures:* The model can be used to evaluate and analyse changes in project structures. Examples include evaluation of the benefits of a proposed CAD system, the effects of co-location of teams working on strongly connected sub-problems, the effects of shortening or eliminating a step in the process, etc.
- (b) *Evaluation of radical changes in the product development process:* The model can also be used to evaluate more drastic changes in the development process, such as eliminating a prototyping or analysis step, or building and testing a prototype before the engineering analysis. These situations would be more difficult to model accurately without substantial experience with the new process. However, the model may be able to identify the potential benefits of proposed changes expressed in terms such as, 'If the more detailed analysis program can reduce the need for testing by even 20%, then is it worthwhile?'
- (c) *Managerial control:* The model identifies the groups of closely coupled, iterative tasks that interact to delay project completion. Highly coupled activities may be accelerated by facilitating information exchanges, perhaps by co-locating individuals involved in performing these tasks or by providing better analytical tools and information systems.
- (d) *Evaluation of schedule risk:* The expected distribution of lead times can be obtained from the model. Knowing the variance of the lead time distribution can aid in understanding the magnitude of schedule risk involved, as well as likelihood and range of project slippage.

6.3. Scope for Future Work

This work leaves several directions open for future work. One important area is to combine these iteration models with the overlapping model developed by Krishnan *et al.* [9]. Such a combination will provide a rich framework on which to extend the frontier of design process models, serving not only to better

characterise these processes, but also to design and optimise them for the best use of time, effort, and money. Further work can be attempted on the use of probabilistic techniques to calculate the duration of the design process rather than the total effort when parallelism is modeled. Resource constraints and queuing effects are not considered here and need to be taken into account.

7. Conclusion

Signal flow graphs provide a powerful, flexible modeling tool for the purpose of analysing product development processes. The modeling method is highly generalised and allows the modeling of dynamically changing design conditions in a limited way and the occurrence of parallelism in a probabilistic sense. The model can be used to compute the distribution of project durations and can easily predict important project metrics such as the expected mean and variance of lead time. The model also provides information regarding the iterative structure of the project, and about the sensitivity of the lead time to model parameters.

Acknowledgments

Funding for this research has been provided by the MIT Leaders for Manufacturing Program. An earlier form of this paper was presented at the 1995 ASME Conference on Design Theory and Methodology.

References

1. Adler PS, Mandelbaum A, Nguyen V, Schwerer E. From project to process management in engineering: an empirically-based framework for analyzing product development time. *Management Science*. March 1995; 41, 3: 458-484
2. Ahmadi RH, Wang H. Rationalizing product design development processes. UCLA Anderson School of Management, Working Paper 1994
3. Belhe U, Kusiak A. Modeling relationships among design processes. University of Iowa, Intelligent Systems Laboratory Working Paper, 08/95, February 1995
4. Bucciarelli LL. *Designing engineers*. MIT Press, Cambridge, MA 1994
5. Clark KB, Fujimoto T. *Product development performance: strategy, organization and management in the world auto industry*. Harvard Business School Press, Boston, 1991
6. Eppinger SD, Whitney DE, Smith RP, Gebala DA. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 1994; 6, 1: 1-13
7. Ha AY, Porteus EL. Optimal timing of reviews in concurrent Design for Manufacturability. *Management Science*, September 1995; 41, 9: 1431-1447
8. Howard RA. *Dynamic probabilistic systems*. John Wiley, New York, 1971

9. Krishnan V, Eppinger SD, Whitney DE. A model-based framework to overlap product development activities. Management Science. Also MIT Sloan School of Management Working Paper no. 3635, rev. September 1996
10. Luenberger DG. Introduction to dynamic systems: theory, models, and applications. John Wiley, New York 1979
11. Mason SJ, Zimmermann HJ. Electronic circuits, signals, and systems. John Wiley, New York 1960
12. Osborne SM. Product development cycle time characterization through modeling of process iteration. Masters Thesis, MIT 1993
13. Pérez-Arriaga IJ, Verghese GC, Pagola FL, Sancha JL, Schweppe FC. Developments in selective modal analysis of small-signal stability in electric power systems. Automatica 1990; 26, 2: 215–231
14. Smith P, Reinertsen D. Developing products in half the time. Van Nostrand Reinhold, New York 1991.
15. Smith RP, Eppinger SD. Identifying controlling features of engineering design iteration. Management Science. Also MIT Sloan School of Management Working Paper no. 3348, rev. October 1995
16. Smith RP, Eppinger SD. A predictive model of sequential iteration in engineering design. Management Science. Also MIT Sloan School of Management Working Paper no. 3160 rev. March 1996
17. Truxal JG. Automatic feedback control system synthesis. McGraw-Hill, New York 1955
18. Ulrich KT, Eppinger SD. Product design and development. McGraw-Hill, New York 1995
19. von Hippel E. Sticky information and the locus of problem solving: implications for innovation. Management Science, April 1994; 40, 4: 429–439
20. Wheelwright SC, Clark KB. Revolutionizing product development: quantum leaps in speed, efficiency, and quality. Free Press, New York, 1992
21. Whitney DE. Designing the design process. Research in Engineering Design 1990; 2, 1: 3–13

Appendix

This Appendix provides an introduction to the manipulation of signal flow graphs (adapted from Howard [8] and Truxal [17]).

A.1. Rules and Definitions of Signal Flow Graphs

Signal flow graphs follow four rules:

1. Signals travel along branches only in the direction of the arrows.
2. A signal travelling along any branch is multiplied by the transmission of that branch.
3. The value of any node variable is the sum of all signals entering the node.
4. The value of any node variable is transmitted on all branches leaving that node.

A *path* is a continuous succession of branches, traversed in the indicated branch directions. The *path transmission* is defined as the product of branch transmissions along the path. A *loop* is a simple closed path, along which no node is encountered more than once per cycle. The *loop transmission* is defined as the product of the branch transmissions in the loop.

The *transmission* T of a flow graph is defined as the signal appearing at some designated dependent node per unit of signal originating at a specified source node. Specifically, T_{jk} is

defined as the signal appearing at node k per unit of external signal injected at node j . There are a number of ways of computing transmissions.

A.2. Basic Operations on Signal Flow Graphs

Solution of signal flow graphs requires knowledge of certain of their topological properties. The basic operations of addition, multiplication, distribution, and factoring can be used to reduce the number of branches and nodes in the system. At first glance, it might appear that by successive application of such transformations a graph could be reduced to a single branch connecting any two given nodes. However, if the graph contains a closed loop of dependencies, as when modeling iteration, one or more self loops will eventually appear.

A.3. The Effect of a Self Loop

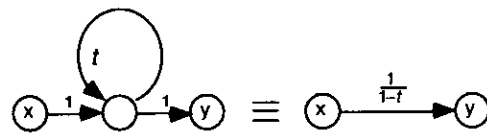
The effect of a self loop at some node on the transmission through that node is analysed in Fig. A.1.

The node signal at the first node is x and the signal returning around the self loop is xt . Since the node signal is the algebraic sum of the signals entering that node, the external signal arriving from the left must equal $y(1 - t)$. Hence, the effect of a self loop t is to divide an external signal by the factor $(1 - t)$ as the signal passes through the node. This holds for all t .

A.4. Solution by Node Absorption

Node absorption corresponds to the elimination of a variable by substitution in the associated algebraic equations. With the aid of the basic transformations and the self loop replacement, any node in a graph can be absorbed and the equivalent expressions for the transmission between two other nodes calculated. Although the branch is no longer shown, its effect is included in the new branch transmission values, as shown in Fig. A.2.

To compute the overall graph transmission, all the intermediate nodes are absorbed in turn, yielding the transmission between the start and finish nodes. Reduction of graphs is computationally intensive and manual solution of graphs of even moderate size can be difficult.



$$\frac{y}{x} = 1 + t + t^2 + t^3 + \dots = \frac{1}{1-t}$$

Fig. A.1. Effect of a self loop.

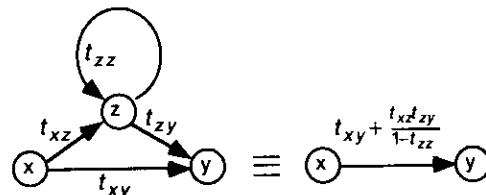


Fig. A.2. Absorption of a node.