Generalized Models of Design Iteration Using Signal Flow Graphs

by

Murthy V.R.K.N. Nukala

Bachelor of Technology in Mechanical Engineering
Indian Institute of Technology, Madras, 1993

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of

Master of Science without Specification

at the
Massachussets Institute of Technology
May 1995

Signature of Author _____
Department of Mechanical Engineering
May 19, 1995

Certified by _____
Steven D. Eppinger, Associate Professor of Management Science
Sloan School of Management
Thesis Supervisor

Accepted by _____
Ain A. Sonin
Chairman, Graduate Committee
Department of Mechanical Engineerng

GENERALIZED MODELS OF DESIGN ITERATION USING SIGNAL FLOW GRAPHS

by

Murthy V.R.K.N. Nukala

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Science

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 cost low and quality high in order to stay competitive. In this context, we focus on improved understanding of design iterations. We are creating tools for modeling product development projects in order to predict the performance of product development organizations. In this paper, signal flow graphs are presented as a flexible tool for design process modeling. Illustrated using an industrial example, key project performance metrics including the probability distribution of lead time, and to identify key drivers of lead time.

Thesis Advisor: Steven D. Eppinger
Associate Professor of Management Science
Sloan School of Management

Table of Contents

1. Introduction

Design Iteration 4

Motivation for new modeling tools 5

2. Signal flow graphs 8

Significance of the graph transmission 9

Determinant of the flow graph 11

3. Application of the signal flow graphs to design process modeling 11

Sensitivity analysis 14

4. Calculation of participation factors 16

Calculation of of participation factors assuming unit task times 17

Extension for participation factors with integral task times 19

5. Discussion 20

Assumptions 20

Further extensions 21

Engineering management insights 22

Scope for future work

6. Conclusions 23

7. Acknowledgements 24

8. References 24

Appendix 1: Signal flow graphs 27

Appendix 2: Parallel Execution of Tasks 30

Appendix 3: Die and Part Geometry 35

1. INTRODUCTION

One of the sources of sustainable competitive advantage in manufacturing-based industries is the capability for superior product design. Today's technology based companies are trying to develop higher quality products faster than ever before while simultaneously improving manufacturability, servicability, recyclability, etc.

The paper is organized as follows: Section 1 motivates the iteration modeling problem and surveys relevant literature. Section 2 presents how signal flow graphs can be used to analyze engineering design processes. Section 3 applies the method to a design process example from General Motors. Section 4 presents a method to compute participation factors for each task to gain further insight into the issues concerning iterations. Sections 5 and 6 present a discussion, conclusions, and scope for future work. The appendices contain a tutorial description of signal flow graphs, and a model extension to incorporate parallelism.

Design Iteration

Iteration is fundamental to the design process. It is defined as the repetition of activities to improve an evolving design. Smith and Eppinger [12 and 13] propose that iterations occur for the following two reasons:

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

Osborne [10] finds that iteration accounts for between one third and two thirds of total development time for projects at a major semiconductor manufacturing company. Osborne also finds that 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 geared to handle iteration or feedback, where tasks may have to be reworked. We are therefore motivated to develop new analytical techniques to model iteration in order to understand iterative development processes.

Design Process Models - An Overview

The view of product development as a modelable process and not as a unique craft has recently gained in popularity [15], [17], [18]. Design process models are usually of two types: performance evaluation models and optimization models. The focus in our work is on performance evaluation.

The key dimensions of design process modeling concern the physics of the work flow (how design 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 [12] 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 using Markov chains and the analysis is used to identify an optimal sequencing of coupled design tasks to minimize iteration time. In another paper, Smith and Eppinger [13] present a parallel iteration model, where the 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. [2] model product development processes in a multiproject setting where there is competition for scarce resources and task queuing effects are

manifest. Ahmadi and Wang [1] extend the sequential iteration model presented by Smith and Eppinger by incorporating dynamic effects (iteration probabilities change with time) and learning (task durations change with iteration number). Ha and Porteus [5] model concurrent design as a dynamic programming problem, where the frequency of design reviews is optimized to shorten lead times and improve design quality. Krishnan [7] studies the characteristics of information dependencies between tasks and develops strategies to profitably overlap sequential tasks while minimizing adverse effects on product quality and development effort.

Motivation for New Modeling Tools

Figure 1 shows a block diagram of the body panel and die design process at a US automotive company. While only a portion of the entire vehicle development process, die design itself is a multi-stage project involving hundreds of people and taking thousands of engineer-hours to complete. The process is highly iterative because the panels need to be optimized for manufacturability during design. As a result, the path followed by the project is uncertain, as are the task durations. The problem is not one of sequencing, as it would be difficult to re-order the tasks to achieve superior performance. It is crucial to manage the interactions between tasks in order to shorten lead times without affecting product quality.

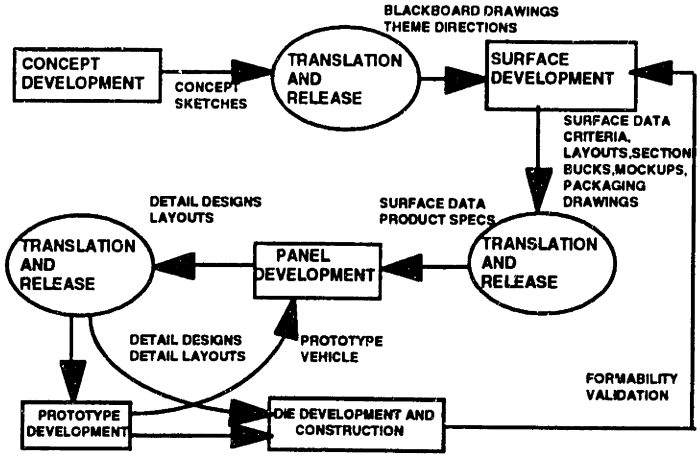


FIGURE 1. AUTOMOTIVE DIE DESIGN PROCESS

In this context, the models presented in the literature fail to capture the effects manifest in design processes in industry. The models in [12] model iteration explicitly using probabilities of rework assuming constant task times. It attempts to reorder tasks to reduce lead time, which may not always be possible. The models in [13] analyze the eigenstructure of the design system and identify iteration drivers, but assume complete parallelism in task execution. The two activity models [5, 7] consider only the interface between upstream design and downstream manufacturing engineering with uni-directional information transfer, and focus on the issues of using imperfect information and detecting manufacturability problems and reducing time to market by starting downstream process design early. These two-activity models [5, 7] do not scale up well to multi-task and bi-directional information transfer scenarios. The queuing models [2] neglect the details of the iteration process. Hence the need for a flexible modeling tool which 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.

2. SIGNAL FLOW GRAPHS

The signal flow graph is a well known tool for circuit and systems analysis in electrical engineering. It is 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 flow graph, shown in Figure 2, is composed of a network of directed branches that connect at points or nodes. Any branch *jk* begins at node *j* and terminates at node *k*, the direction from *j* to *k* being indicated by an arrowhead on the branch. Each branch *jk* has associated with it a quantity known as the *branch transmission* P_{jk}.

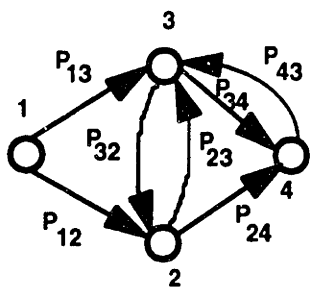


FIGURE 2. A LINEAR SIGNAL FLOW GRAPH

For our modeling purposes, the branches represent the tasks being worked (activity-on-arc representation). The branch transmissions are $\{P_{ij} = p_{ij} z^{t_{ij}}\}$, where p_{ij} is the probability associated with the branch, and t_{ij} is the time taken to traverse the branch (the probability and time to execute the task represented by the branch). z is the transform variable used to connect the physical system (time domain) to the quantities used in the analysis (transform domain). (See Appendix 1 for a definition of this geometric transform). 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 used for analysis.

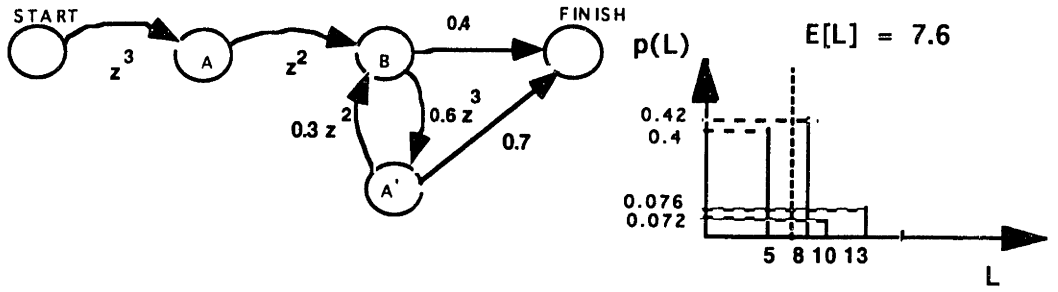
Appendix 1 presents a tutorial on signal flow graphs, and explains the absorption of nodes in a graph, an algebraic technique for graph simplification. The *graph transmission* between two given nodes is the resulting expression on an arc connecting the two nodes when all the other nodes are absorbed.

Significance of the Graph Transmission

Let T_{ij} represent the graph transmission between nodes *i* and *j*. If T_{ij} is expanded as an infinite series, each term in the series represents one path between the two nodes. When the two nodes *i* and *j* are the start and the finish nodes of a design process, each path represents one possible route through the entire process. Henceforth, *graph transmission* shall refer to the graph transmission between the start and the finish nodes, denoted by T_{sf}. A path transmission is defined as the product of all branch transmissions along the path. The graph transmission is the sum of the path transmissions of all the possible paths between the start and finish nodes. (As there are cycles in the system due to iteration, the number of paths is infinite.) The coefficient of each term in the graph transmission is the probability associated with the path it represents, and the exponent of z is the process lead time associated with this path. Hence, the graph transmission is a function representing the probability distribution of the lead time of the process.

A numerical example is presented in Figure 3. A hypothetical design process is represented by the graph shown. It includes tasks A and B, which 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 nominal (once through) time for A and B in series is 5, 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. The

graph transmission is the sum of the infinite number of paths the system can take, each of which is associated with a lead time and a probability. The first few terms of the probability distribution function of lead times are represented graphically on the right-hand side of Figure 3.



Probability density function of lead time L

FIGURE 3. DISTRIBUTION OF LEAD TIMES

The sum of the infinite number of path transmissions is the graph transmission, which is given by

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

The graph transmission can be differentiated and manipulated to obtain the expected value and variance of lead times.

$$E[L] = \frac{\partial T_{sf}}{\partial z} \Bigg|_{z=1}$$

This is 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. When differentiated, this term becomes $(p_i\, t_i)\, z^{t_i - 1}$. Evaluated at z=1 and summed over all terms (paths), this is the expected value of the lead time of the process. Similar arguements lead to:

$$E[L^2] = \frac{\partial \left(z \dfrac{\partial T_{sf}}{\partial z} \right)}{\partial z} \Bigg|_{z=1}$$

$$\text{Variance}[L] = E[L]^2 - E[L^2]$$

Determinant of the Flow Graph

The determinant of the flow graph is the expression which appears in the denominator of the graph transmission expression. Mason [8] explains some interesting properties of the graph determinant. The roots of the graph determinant identify the poles of the system which govern its transient response to external input signals. The determinant of the flow graph is the determinant of the coefficient matrix for the analogous set of linear equations represented by the flow graph. (See Appendix 1 for the representation of the signal flow graph as a system of linear equations).

3 APPLICATION OF SIGNAL FLOW GRAPHS TO DESIGN PROCESS MODELING

Iteration plays a central role in design and development projects. The primary assumption of the proposed signal flow graph model is that iteration is determined by a probabilistic rule, with probabilities assigned to the event that coupled tasks will have to be repeated in order to resolve conflicts and manage complex information dependencies. It is further assumed that there are no resource constraints and no delays due to causes like the inefficiency of information transfers. An industrial example is now presented and analyzed to illustrate use of the model, and to motivate the need for further extensions.

The block diagram in Figure 4 depicts the body panel die design process at General Motors. Acceleration of this 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 are common when formability problems are discovered during the die production phase. The panel design needs to be analyzed for formability early because of the need to

provide early manufacturability feedback to the design, but when analyzed too early, the design may change and nullify the analytical results.

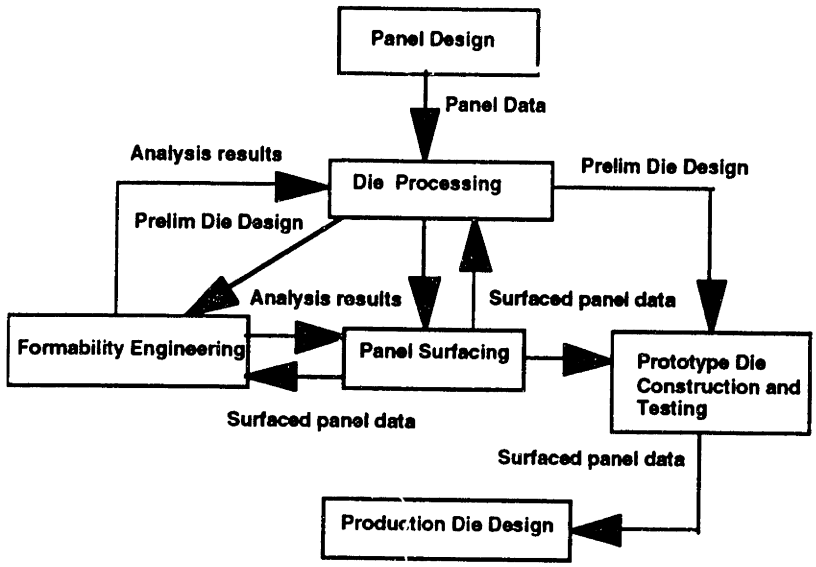


FIGURE 4. THE DIE DESIGN PROCESS

The die design group gets information from the panel designers,from which the panel design is evaluated from a manufacturing standpoint. This requires modeling of both the die and the panel forming process. Initial die processing involves receiving information from the upstream panel design activity, and detailing key die characteristics. The panel data and preliminary die design are then transferred to the formability engineers for 2-D analysis which can detect some potential manufacturing problems. The results are sent back to the die processors. At this point, the panel design is still in the form of line sections. The die processors in their second iteration, receive the 2-D analysis results and change the preliminary die design to avoid manufacturing problems.

The surfacing engineers then use the modified die design, the panel design and the 2-D analysis results to develop surfaces of the panels. The surfaced panels are then sent back to formability engineering for 3-D analysis. 3-D analysis tools simulate the process of stamping a metal blank into a panel and

detect associated problems. The surfaced panel is then analyzed, and the results forwarded to die processing and surfacing engineers. The die processors modify the die design to avoid manufacturing problems. The surfacing engineers also used the 3-D analysis results to improve the manufacturability characteristics of the panel. The 3-D analysts, die processors and die surfacing engineers iterate to create a final panel design. The final data from this stage is then used to build the prototype die and test it, after which the hard tooling is built for the factory floor.

The process is therefore highly iterative, requiring dense information flows between the product designers, die processors, formability engineers, and surface designers to create part and die designs optimized for manufacturability.

The signal flow graph of this process is shown in Figure 5. The times represented are in work days. The flow graph shows the tasks formability engineering and die processing replicated since the rework probabilities and task times change after the first iteration. Using the analytical method introduced above, the expected value of the lead time of the system shown in the figure is 48.4 days and the standard deviation of the lead time is 17.4 days.

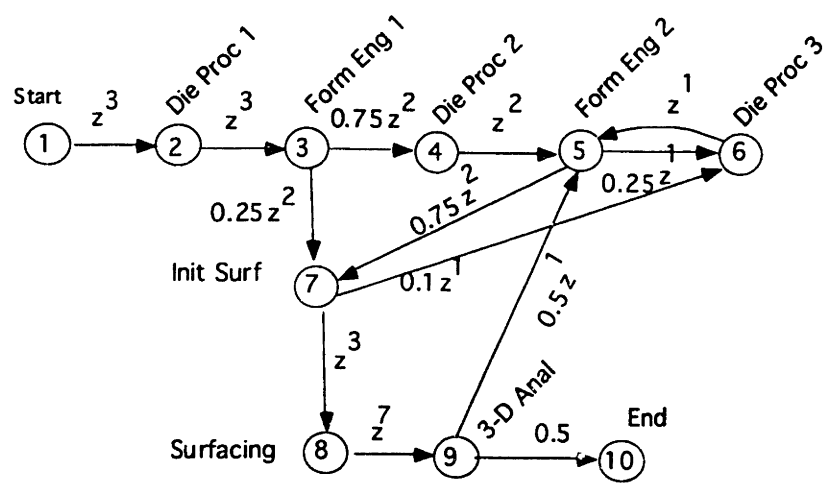


FIGURE 5. FLOW GRAPH REPRESENTATION OF DIE DESIGN VERIFICATION

Data were collected from the Die Management Group at General Motors regarding the lead times for panel design. Panels are coded according to complexity, with A being the simplest and F the most complex. Data were collected about a number of panel types and 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 19.75 days. The model predicts the process lead time and standard deviation with reasonable accuracy (22% low). Possible sources of error are modeling and data collection. There are several effects in real projects which are not handled in our model. Also, the probabilities of iteration had to be inferred from interviews with engineers working on the project. This is another potential source of error, since the project lead times predicted by the model are sensitive to variations in probabilities of iteration.

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

$$S_k^L = \frac{dL/L}{dk/k}$$

Calculation of the sensitivity of the expected value of lead time to changes in task times yields the lead time sensitivity matrix in Figure 6. The (i,j) term of the matrix is the sensitivity of the lead time to changes in duration of task i in interaction (i,j).

Task	2	3	4	5	6	7	8	9
2	0	0.75	0	0	0	0	0	0
3	0	0	0.06	0	0	0.02	0	0
4	0	0	0	0.06	0	0	0	0
5	0	0	0	0	0.04	0.26	0	0
6	0	0	0	0.06	0	0	0	0
7	0	0	0	0	0.01	0	0.36	0
8	0	0	0	0	0	0	0	0.87
9	0	0	0	0.09	0	0	0	0

FIGURE 6. LEAD TIME SENSITIVITIES TO VARIATIONS IN TASK TIMES

Calculating the sensitivity of the expected value of lead time to the branch probabilities yields the probability sensitivity matrix. The (i,j) term of the matrix is the sensitivity of the lead time to changes in the probability p(i,j).

Task	3	4	5	6	7	8	9
3	0	0.78	0	0	0.22	0	0
4	0	0	0	0	0	0	0
5	0	0	0	**2.1**	**7.18**	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0.62	0	0	0
8	0	0	0	0	0	0	0
9	0	0	**4.85**	0	0	0	1.00

FIGURE 7. LEAD TIME SENSITIVITIES TO VARIATIONS IN BRANCH PROBABILITIES

The expected value and variance of lead time were found to have low sensitivity to small changes in task times, and higher sensitivity to small changes in the probabilities of rework. This suggests that managing the interactions between tasks better (reducing repeat probabilities) has a greater impact on reduction of lead time than reducing individual task times. This also suggests that the model error of 22% could be largely due to slight errors in a few probabilities. For example, with a sensitivity of 7.18, an error of 3.06% in the probability of iteration would cause a model error of 22%. Further, since the lead time has low sensitivity to task times, the assumption of integral task times seems to be reasonable.

Figure 7 shows the sensitivity of process lead time to the probabilities in the flow graph. The interaction between tasks Formability Engineering 2 (task 5) and Initial Surfacing (task 7) has the greatest effect on lead times, followed by interaction between the 3-D analysis (task 9) and Formability Engineering 2(task 5). The reason for the importance of the interactions between tasks 5 and 9 could be that this interaction is part of two slow loops - the loop comprising tasks 5-6-7 and the loop consisting of tasks 5-7-8-9. The interaction between task 9 and task 5 is important because it is a long feedback loop. The sensitivity values indicate that even small changes in the probability of iteration have large effects on the lead time of the process.

4. CALCULATION OF PARTICIPATION FACTORS

Further analysis yields information 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 which 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 [13]. 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 mode.

Calculation of Participation Factors Assuming Unit Task Times

The concept of participation factors is well developed in linear system theory [11]. Let the signal flow matrix be P. If all the task times in the graph are set equal to unity, then the roots of the determinant of the system (which are the poles of the system) are the roots of {det[I - Pz] = 0}. The eigenvalues of the system are the roots of {det[sI - P]}. By inspection, the eigenvalues of the system determine the roots of the characteristic equation of the system (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. The largest participation factors correspond to the components of the system contributing to the slowest modes. In its present form, the analysis is similar to the eigenstructure analysis demonstrated by Smith and Eppinger [13] for a parallel iteration model.

The participation factors for the body surface design process are now computed. The signal flow matrix P for the system in Figure 5 assuming unit task times (without the start and finish nodes) is

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	1
9	0	0	0	0.5	0	0	0	0

FIGURE 8. SIGNAL FLOW MATRIX P OF THE DESIGN PROCESS

The eigenvalues of the matrix P, the roots of [det(sI - P) = 0], are {1.00, 1.00, 1.00, 0.1047, **1.8353**, **1.03 ± 0.7075i**, 1.00}

The corresponding participation factors (obtained by the termwise multiplication of the right and the left eigenvectors) are:

	1s	2nd	3rd	4th	5th	6th and 7th	8th
2	1	0	0	0	0	0	0
3	0	1	0	0	0	0	0
4	0	0	1	0	0	0	0
5	0	0	0	0.306	**0.214**	**-0.199∓0.028i**	0
6	0	0	0	0.127	0.067	-0.103 ± 0.028i	0
7	0	0	0	0.21	**0.188**	**0.3 ± 0.025i**	0
8	0	0	0	0.178	**0.226**	**-0.29 ∓ 0.042i**	0
9	0	0	0	0.178	**0.226**	**0.29 ± 0.0166i**	1

FIGURE 9. PARTICIPATION FACTOR MATRIX

Looking at the column of the participation factors matrix corresponding to the three larger eigenvalues, we see that the slowest modes of the system are affected by the tasks 5, 7, 8 and 9. This would indicate that the loop involving the 2-D formability analysis, die face surfacing, and the 3-D analysis dominates the convergence characteristics of the design process, and needs to be managed effectively to reduce the lead time of the process. Refer to Smith and Eppinger [13] for a richer explanation of eigen structure interpretation for design processes.

Extension for Participation Factors with Integral Task Times

The assumption of unit task times distorts the transient response of the system. This is because the lead time of a particular path assuming unit delays is equal to the number of branches traversed, and does not incorporate task times. Hence the probability density function of the lead times is incorrect. Since we are interested in computing the probability of reaching the finish state by various paths of different lengths, an extension of the model to incorporate task times is now presented.

We introduce (n-1) dummy states on each arc of the signal flow graph, where n is the integral task time associated with the branch. In the case of non-integral task times, they are all made integral. This 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 has unit delay and unit probability. In the expanded graph shown in Figure 10, each task is represented by several nodes.

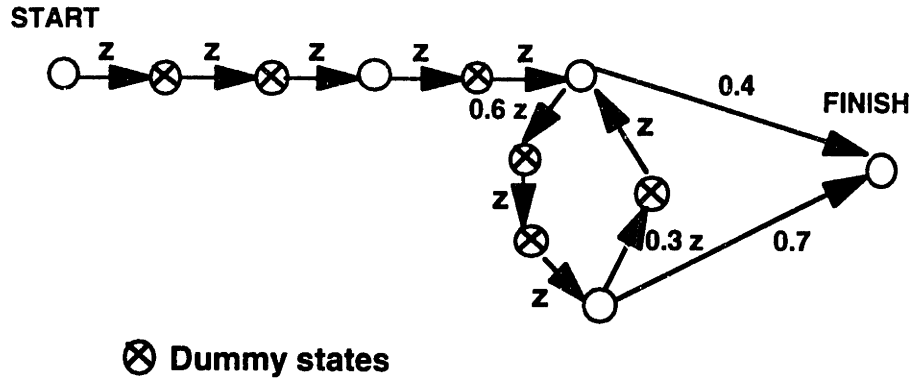


FIGURE 10. INCORPORATION OF TASK TIMES

5. DISCUSSION

Assumptions:

The models rely on several assumptions about the design process, the accuracy of which determine the fidelity of the analysis. It is assumed that the durations of tasks involved in the design process are deterministic, and that the interactions between tasks can be captured using a probabilistic rework rule. Extensions to the basic model are useful in depicting certain design situations accurately. Multiple tasks are 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 when conditions change with every iteration, as the number of states in the expanded graph representing the system blows up.

Further Extensions

The model allows flexibility along a number of dimensions, allowing probability distributions over rework, and incorporation of task times. For a discussion on the extension of the model to incorporate parallelism, see Appendix 2. The case where the magnitude of rework is not deterministic, but distributed according to some discrete probabilistic rule can be handled as in the Figure 11. B, B' and B" are not different tasks, just instances of different amounts of rework.

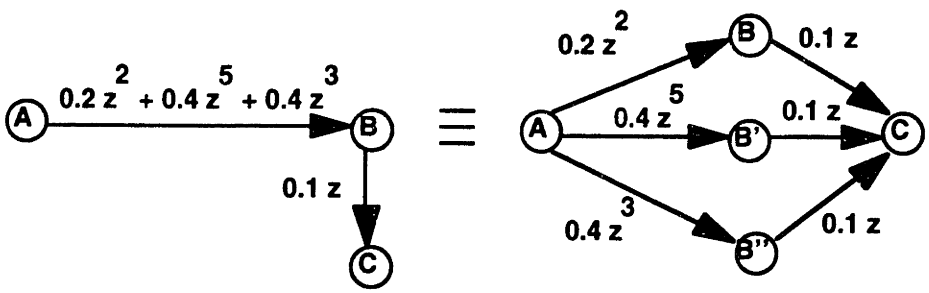


FIGURE 11. PROBABILITY DISTRIBUTION OVER REWORK

This phenomenon is typically seen in validation situations where the magnitude of rework depends on the type of error detected and there is a distribution over the types of errors detected.

Another possible extension is to handle cases where the rework probabilities and task durations change dynamically. This is a difficult and non-trivial problem to solve. Ahmadi and Wang [10] attack this problem 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 they remain constant. We believe this to be a good representation of reality in many design situations, because it is likely that the duration of a task

will not reduce infinitely with increase in iteration number, but settle at some constant rework time after a few iterations.

However, collection of project data about strength of interactions, modeled as probabilities, proved to be difficult, and were based upon estimates by design engineers and managers. Iteration process maps, a tool developed by Osborne [10], were used to collect information about the design process.

Engineering Management Insights

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

(a) Evaluation of alternate project structures

The model can be used to evaluate and analyze changes in project structures. Examples include evaluation of the benefits from a new 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 product development process

Examples of this scenario include complete changes in process, like eliminating the prototyping step, or doing prototyping before analysis, etc. This situation, though rare, can be very difficult to manage. The model helps identify the effects of these radical changes.

(c) Managerial Control

The model identifies the groups of closely coupled tasks which interact to elongate project lead times. This focuses managerial attention on these groups of tasks. Highly coupled processes can be accelerated by facilitating rich information exchanges, perhaps by co-locating individuals involved in

performing these tasks. The sensitivity analysis can further help identify the project linkages which have the most effect on lead time.

(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.

Computation of the expected value and variance of lead times involves differentiation of a polynomial. The computation was efficient for small problems, and took about 2 minutes for each computation for 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.

Scope for future work

The 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 [7]. Such a combination will provide a rich framework on which to extend the frontier of design process models, serving not only to better characterize these processes, but also to design and optimize them for the best use of time, effort and money. Further work can be attempted in the use of probabilistic techniques to calculate the makespan of the design process rather than the total effort when parallelism is modeled. Resource constraints are not considered here and need to be taken into account.

6. CONCLUSION

Signal flow graphs provide a powerful, flexible modeling tool for the purpose of analyzing product development processes. The modeling method is highly generalized, and allows the incorporation of several effects observed in industrial practice. It also 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 obtain information about project metrics like the moments of lead time. The model also provides information regarding the structure of the project, and about the sensitivity of the lead time to process parameters.

7. ACKNOWLEDGMENTS

Funding for this research has been provided by the MIT Leaders for Manufacturing Program. We appreciate the assistance of the General Motors Technical Center and the General Motors Die Management Group in developing the examples presented in this work.

8. REFERENCES

1. Ahmadi, R.H. and Wang, H. *Rationalizing Product Design Development Processes*, John E. Anderson Graduate School of Management, UCLA, 1994.
2. Adler, P., Mandelbaum, A., Schewerer, E., and Nguyen.V. *From Project to Process Management in Engineering: An Empirically-based Framework for the Analysis of Product Development,* WP #3503-92-MSA, MIT Sloan School Working Paper, 1992.
3. Clark, K. and T. Fujimoto. *Overlapping Problem Solving in Product Development*, Harvard Business School, Working paper # 87-048, 1987.

4. Clark, K. and T. Fujimoto. *Reducing the Time to Market: The Case of the World Auto Industry.* Design Management Journal, pp. 49-57, Vol. 1, No. 1, 1989.

5. Ha, A.Y., and Porteus, E.L. *Optimal Timing of Reviews in Concurrent Design for Manufacturability*, Research Paper #1184Rev, 1993.

6. Howard, R.A. *Dynamic Probabilistic Systems Vol. 1 and Vol. 2* , John Wiley and Sons, New York, 1971.

7. Krishnan, V. *Design Process Improvement: Sequencing and Overlapping Activities in Product Development*, ScD Thesis, MIT, 1993.

8. Mason, S.J., and Zimmermann, H.J. *Electronic Circuits, Signals, and Systems* , John Wiley & Sons, New York, 1960.

9. Nevins, J.L. and Whitney, D.E. *Concurrent Design of Products and Processes: A Strategy for the Next Generation in Manufacturing.* McGraw Hill Publishing Company, New York, 1989.

10. Osborne, S. *Product Development Cycle Time Characterization Through Modeling of Process Iteration*, Masters Thesis, MIT, 1993.

11. Perez-Arriaga, I.J., Verghese, G.C., Pagola, F.L., Sancha, J.L., and Schweppe, F.C. *Developments in Selective Modal Analysis of Small Signal Stability in Electric Power Systems*, Automatica, Vol. 26, No 2, pp. 215-231, 1990.

12. Smith, R.P. and Eppinger, S.D. *A Predictive Model of Sequential Iteration in Engineering Design*, Working Paper, MIT Sloan School of Management, WP # 3348-91-MS, 1991, Revised April 1994.

13. Smith, R.P. and Eppinger, S.D. *Identifying Controlling Features of Engineering Design Iteration*, Working Paper, MIT Sloan School of Management, WP # 3348-91-MS, 1991, Revised Dec 1994.

14. Sittler, R.W. *Analysis and Design of Simple Nonlinear Noise Filters*, ScD Thesis, MIT, 1960.

15. Ulrich, K.T. and Eppinger S.D. *Product Design and Development*, McGraw Hill Book Company, 1994

16. Truxal, J.G. *Automatic Feedback Control System Synthesis*, McGraw Hill Book Company, New York 1955.

17. Wheelwright, Stephen C., and Kim B. Clark. *Revolutionizing PRoduct Development: Quantum leaps in Speed, Efficiency and Quality*, The Free Press, New York, 1992.

18. Whitney, D.E. *Designing the Design Process.* Research in Engineering Design. pp. 3-13, Vol. 2, 1990.

APPENDIX 1. SIGNAL FLOW GRAPHS

Rules for the Signal Flow Diagram [16]

Rule 1 : Signals travel along branches only in the direction of the arrows.

Rule 2 : A signal traveling along any branch is multiplied by the transmission of that branch.

Rule 3 : The value of any node variable is the sum of all signals entering the node.

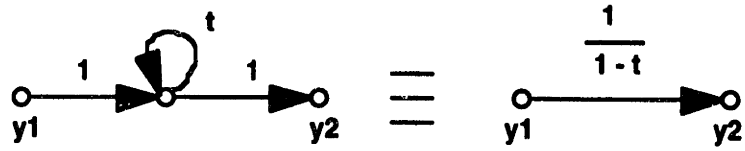
Rule 4 : The value of any node variable is transmitted on all branches leaving that node.

Basic Operations on Flow Graphs

Solution of the flow graphs requires knowledge of certain topological properties of flow graphs. 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 desired nodes. However, if the graph contains a closed loop of dependencies, as it will while modeling iteration, one or more self loops will eventually appear.

The Effect of a Self Loop

The effect of a self loop at some node on the transmission through that node is analyzed in Figure A1.1.



$$\frac{y2}{y1} = 1 + t + t^2 + t^3 + \ldots\ldots = \frac{1}{1-t}$$

FIGURE A1.1 EFFECT OF A SELF LOOP

The node signal at the first node is y1 and the signal returning around the self loop is y1*t. Now, since the node signal is the algebraic sum of the signals entering that node, the external signal arriving from the left must equal y2(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.

Absorption of a Node

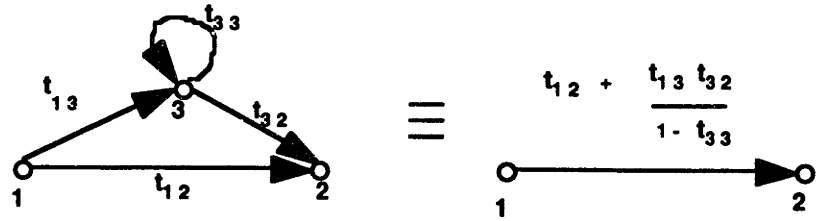


FIGURE A1.2 ABSORPTION OF A NODE

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 Figure A1.2.

Transmission of a Flow Graph

Before defining the transmission of a flow graph, the following definitions are in order.

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

Solution by Node Elimination

First, the two nodes between which the transmission is to be computed are identified. Then, all the remaining nodes are absorbed in turn, yielding the required transmission between the designated nodes. Reduction of the graphs is computationally intensive and solving graphs of even moderate sizes is computationally expensive.

Solution of Linear Equations

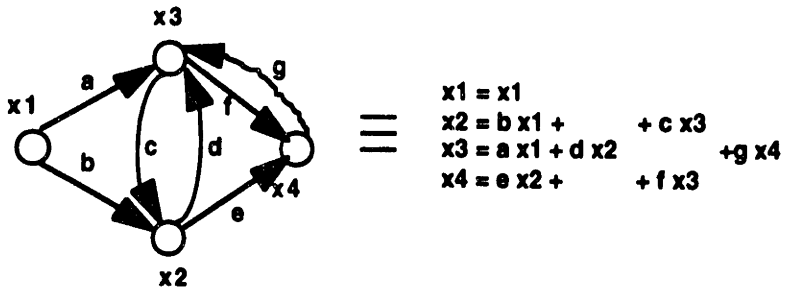


FIGURE A1.3 FLOW GRAPH AS A SYSTEM OF LINEAR EQUATIONS

In equation form, the flow graph is analogous to the set of linear equations {P} [y] = [A], with {P} and [A] determined from the structure of the flow graph as in Figure 6. The transmission of the graph from any node to any other is obtained by elimination of the variables corresponding to the other nodes from the system of linear equations.

The Geometric Transform

Consider a discrete function which can take on any real value, positive or negative, at any non-negative integer, n=0, 1, 2, ... It is also convenient to define the function to take on the value zero at all negative integers. The geometric transform of such a function is found by multiplying it termwise with a geometric sequence in the transform variable and summing. We shall use the transform variable z and denote the transform of a function f(n), n= 0, 1, 2, ... , by $f\mathcal{G}(z)$. The geometric transform is then defined by

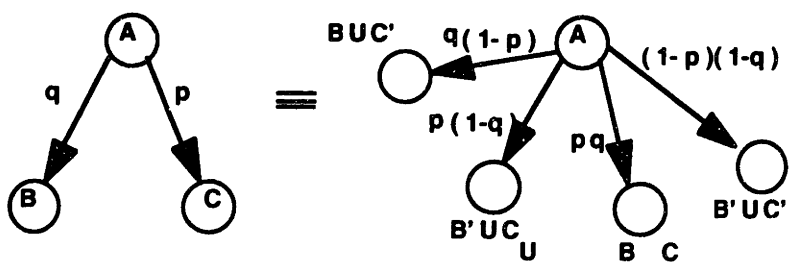
$$f\mathcal{G}(z) = f(0) + f(1)z + f(2)z^2 + f(3)z^3 + \ldots\ldots\ldots\ldots = \sum_{n=0}^{\infty} f(n)z^n$$

APPENDIX 2. PARALLEL EXECUTION OF TASKS

The models created have relied on assumptions about the order of execution sf tasks and the number of tasks working at any given point in time. These assumptions are not always satisfied in real projects. A modification of the

definition of a state in the probabilistic model enables the development of a more flexible model.

The definition of a state of the graph is altered in the following way. After a given state is reached (a task has been completed) all the tasks dependent on this task for information are attempted based on an individual probabilistic rule. The probability of repeating a task is governed only by the degree of 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 A2.1 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.



Legend :
B' - Complement of B
B U C - Both B and C work

(a) **(b)**

FIGURE A2.1. REDEFINITION OF A STATE

In Figure A2.1, the two systems represented by the two figures are analogous in an analytical sense. If the sum of probabilities p and q is restricted to 1, then the Figure A2.1 (a) represents a sequential iteration process. In this case, the analogous representation in Figure A2.1(b) would also yield the same results when subjected to analysis, 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 Figure A2.1 (a) does not represent sequential

iteration anymore, but the analogous representation in Figure A2.1 (b) can be used to model parallelism in a probabilistic sense.

In the signal flow graph, a path represents one instantiation of the design process. Due to the definition of a path, only one task is allowed to work at a given time along any path. 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 situation.

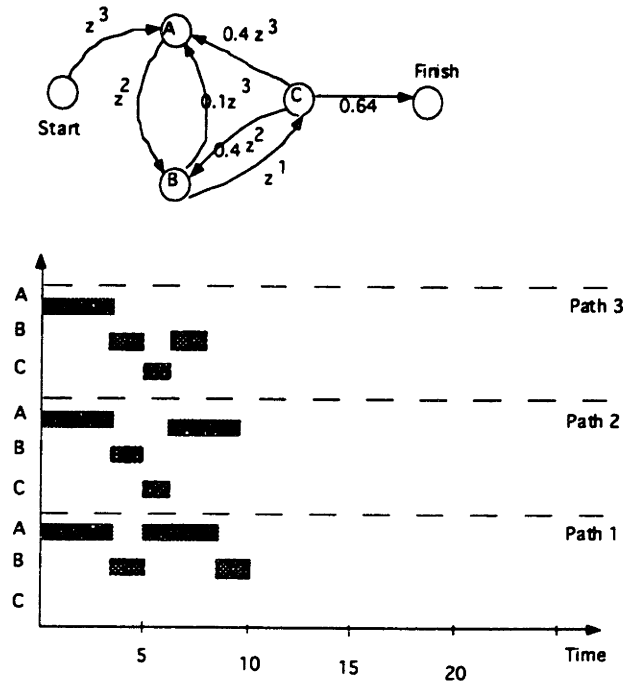


FIGURE A2.2. PARALLELISM IN DESIGN ACTIVITIES

Figure A2.2 illustrates the issue of parallelism. 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, and the sum of the path transmissions of all the possible paths gives the graph transmission. Hence the method of analysis outlined in the previous section still applies to the current 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 simulation 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.

Implications

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 which 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 one drawback of the model. However, the model represents reality in several situations, where preliminary information is transferred upon availability, and changes in this information and additional information are incorporated into the downstream tasks. Another implication is that in some situations, it might come to be that a task both works, and is 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 paths active at that point in time. This is again one of the

potential drawbacks of the model. However, it has been found in several realistic industrial situations that changes are requested of the task before the task completes working, so rework iterations and nominal iterations are not easily distinguishable. In such situations, we believe the model is reasonable.

APPENDIX 3 : DIE AND PART GEOMETRY

This section focuses on identifying and defining the elements of a die section The elements of a section describe the area outside the edge of the part. A **section** is a sectional view of a line drawn across the surface of the sheet metal beginning near the middle of the part and proceeding outward to the edge of the sheet metal. Inside the edge of the part is the punch face which forms the sheet metal into the product shape as it is determined by product styling and engineering. Beyond the edge of the part is the addendum, comprising of the Part Addendum and the Die Addendum Figure A3.1 shows the die cross section with Part and Die Addenda.

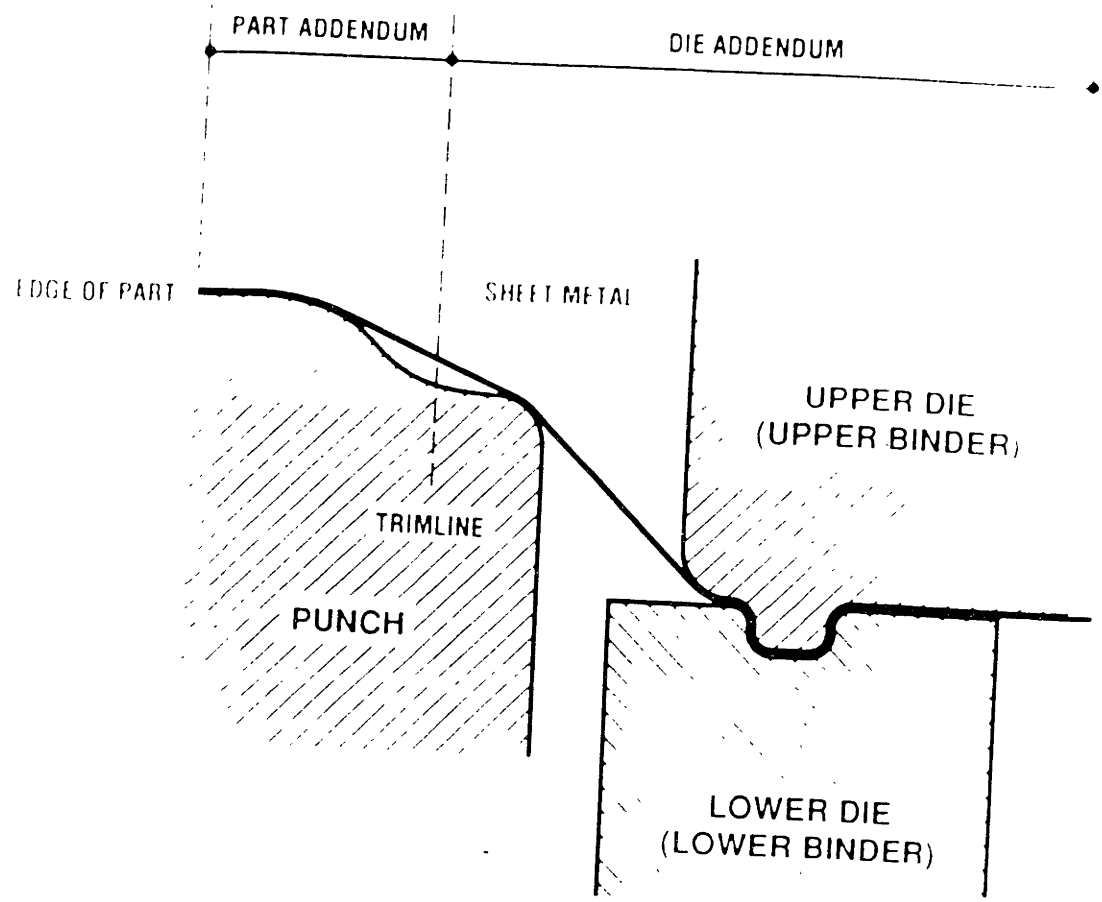


FIGURE A3.1 PART AND DIE ADDENDA

A3.1 Part Addendum

The part addendum includes the elements between the edge of the part and the trim line. There are two elements within the part addendum, plus region and the trim line. The plus region is the amount of sheet metal added to the tangent point of the final product shape to compensate for the amount of work hardening, bending and shrinkage which occurs during forming. Plusing provides more metal allowing for a larger radius and more favorable forming conditions without affecting the product surface. The trim line is the outline around the perimeter of the sheet metal part. Figure A3.2 depicts the part addendum.

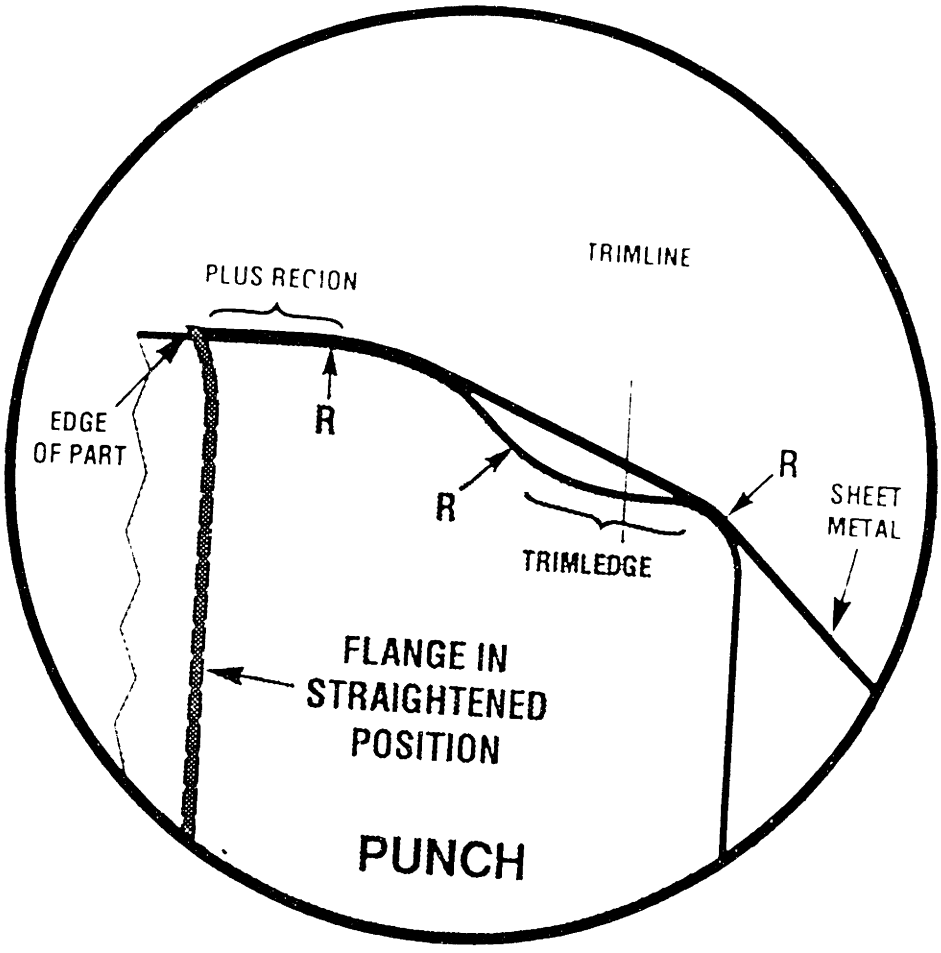


FIGURE A3.2 PART ADDENDUM

A3.2 Die Addendum

The die addendum is outward beyond the part addendum and includes all of the elements outside of the trim line. The die addendum includes the following elements. The trim ledge is a flat punch or die surface which accommodates the trim line. The punch break line is a line which identifies the outer most edge of the punch surface. The punch break radius is the radius associated with the punch break line and blends the punch surface with the stretch or draw wall The punch perimeter is the outermost surface of the punch The stretch or draw wall is the surface which is bounded by the punch perimeter line and the die perimeter line Figure A3.3 depicts the die addendum

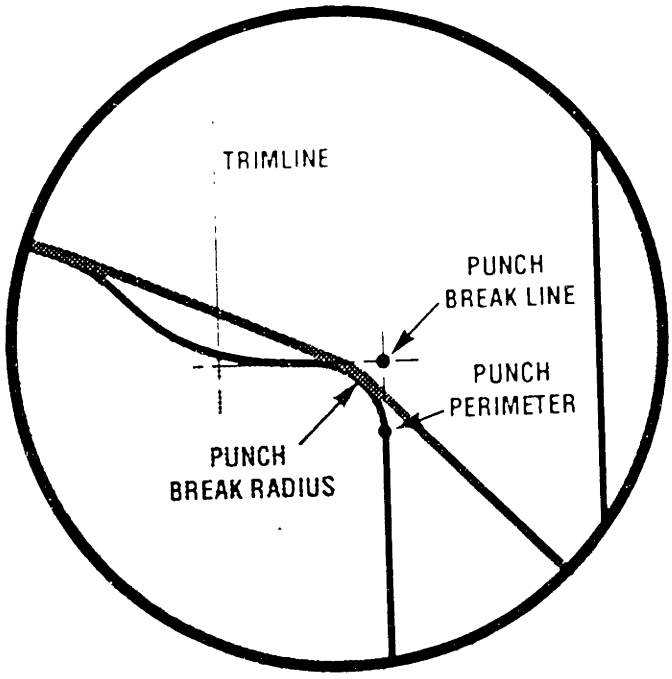


FIGURE A3.3 DIE ADDENDUM

THESIS PROCESSING SLIP

FIXED FIELD ill _____ name _____

index _____ biblio _____

▶ COPIES Archives Aero Dewey Eng Hum

Lindgren Music Rotch Science

TITLE VARIES ▶☐ _____

_____

_____

NAME VARIES ▶☐ _____

_____

IMPRINT (COPYRIGHT) _____

▶COLLATION 37l

_____

▶ADD DEGREE _____ ▶DEPT. _____

SUPERVISORS _____

_____

_____

_____

_____

_____

_____

_____

NOTES:

cat'r. date

page: 5125

▶DEPT [redacted] M.E.

▶YEAR 1995 ▶DEGREE M.S.

▶NAME NUKALA, Murthy

U.R.K.N.