DESIGN PROCESS IMPROVEMENT: SEQUENCING AND OVERLAPPING ACTIVITIES IN PRODUCT DEVELOPMENT

by

Viswanathan Krishnan

B. Tech., Indian Institute of Technology
(July 1988)

M. E., Carnegie-Mellon University
(August 1990)

Submitted to the
Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Science
at the
Massachusetts Institute of Technology
September 1993

Signature of Author _____
Department of Mechanical Engineering
July, 1993

Certified by _____
Professor Steven Eppinger
Thesis Supervisor

Accepted by _____
Professor Ain A. Sonin
Chairman, Department Graduate Committee

DESIGN PROCESS IMPROVEMENT: SEQUENCING AND OVERLAPPING ACTIVITIES IN PRODUCT DEVELOPMENT

by

Viswanathan Krishnan

Submitted to the Department of Mechanical Engineering

on July 31, 1993 in partial fulfillment of the

requirements for the degree of Doctor of Science in

Mechanical Engineering

ABSTRACT

This thesis focuses on the study of design processes in which the constituent steps and technologies are stable over time – with the aim of formulating methods to improve performance in terms of product quality, product development lead time, and development effort. Close examination of such processes in industry shows that the flow of information and the pattern of execution in these processes is largely sequential with information being generated by upstream development functions (such as styling) and transferred to downstream functions (such as engineering design or prototyping) for further processing. This thesis examines how the sequential execution affects development performance, and how it may be altered to improve performance.

It is seen that resequencing the design process can help improve product quality. The effect of the design decision sequence on the product quality is investigated. Also, the design activities can be profitably overlapped – thereby accelerating the product development process. A conceptual framework is presented which would help the designer/manager decide *when* and *how* to overlap the activities. The framework and the associated models are found to be useful to overlap activities and reduce product development lead time while ensuring that the adverse effects on product quality and development effort are minimized. Field application of the framework and models to the performance improvement of product development processes in the automotive and the electronics industry are also described.

Thesis Committee:
Dr. Daniel E. Whitney (Chairperson and Joint Supervisor)
Professor Steven D. Eppinger (Joint Supervisor)
Professor Warren P. Seering
Dr. Srikanth M. Kannapan (Xerox Design Research Institute)
Professor Gabriel R. Bitran

Preface

If there's a book you really want to read but it hasn't been written yet, then you must write it! – Toni Morrison

Traditionally, design has been viewed as a unique, craft activity. Many products in industry, however, are redesigned – to new specifications. In this thesis, I view design as a process – a collection of steps whose completion results in an end product – with the aim of identifying methods to model and improve product development performance. In Chapter 1 of the thesis, I present a summary of the methods and ideas contained in this thesis. In subsequent chapters, these ideas are developed in more detail.

The research described in this thesis was conducted under the auspices of the MIT Leaders for Manufacturing (LFM) program – a timely partnership between MIT's engineering and business schools and thirteen U. S. Manufacturing firms. The LFM program has generously supported my repeated visits to two of the partner companies, Chrysler Corporation, and Motorola Inc., for field study and data collection. These field visits have been an enlightening experience; I gained first-hand exposure to the technical and organizational challenges companies face in developing products.

Several people at Chrysler helped me gather and interpret data. I am most indebted to Alan Carlson who, despite his exacting task of managing the process of development of an entire automobile, met with me frequently (often beyond his office hours!), offered me office space, and gave me access to a wealth of data, and much valuable guidance and encouragement. Special thanks also go to: Andy Kuzdak, Dave Anderson, Mike St. Pierre, Steve Mitchell, Tom Kollar, Cliff Pacurari, and Andy Cleek.

Liz Altman coordinated my interactions with Motorola, and made the data collection effort possible through her energy, and cheerful promptness. I am grateful to her for following my research with interest, and for offering valuable feedback that helped me refine the underlying ideas.

By far the greatest influence on this thesis has been the guidance and critique of my advisors, Steven Eppinger and Daniel Whitney. I appreciate

their confidence in my ability, and encouragement at every stage of this thesis. Steve helped me make my stay at MIT memorable with his pleasant and friendly disposition. I am indebted to him for continually calling my attention to the applicability of the models I developed, and to the details of my presentation. Steve's commitment to students and teaching is admirable. I will try to emulate him in my interaction with students.

Early in my doctoral student days, I discovered the value of obtaining Dan Whitney's feedback to my ideas. Subsequently, I maintained a high bandwidth communication line with him, and benefited *immensely* from his experience and perceptive comments. Dan constantly challenged me to set my sights higher, and make my arguments sharper; Thesis chapter 1 underwent a major revision after he applied his scalpel (red pen) to it!

My numerous meetings with my doctoral committee have also helped shape the intellectual development of this work. Srikanth Kannapan offered detailed comments, and suggested generalizations of the models in Chapter 5 from an earlier draft of the thesis – which have been done. Despite their tight schedule, Gabriel Bitran and Warren Seering offered much valuable feedback.

(Captain) Mark Morelli, and my long time friends Sthanu and Bala enriched my life with diversions. Jim Rinderle, Prashant Rane, and Steve Hoover were very influential during my Carnegie Mellon days. I am grateful to Mr. E. N. Rao for providing me Indian classical music instruction – complementing my graduate study, and offering me great joy.

My cousins, aunts, and uncles have rooted for me since times immemorial; Ananthanarayanan has been a special inspiration. My brother Murali made me eagerly look forward to his letters from home filled with puns. Lakshmi, my wonderful sister, has shared with me the joys and sorrows of graduate student life. I thank her for lending me her ears and support, and wish her the best in her efforts at getting a graduate degree from MIT. Our parents, Sugantha and Viswanathan, faced many odds, and *dedicated* themselves to the well being of their children. I hope I have done a doctoral dissertation that would make them proud!

V. Krishnan July 15, 1993

"Tis the good reader that makes the good book." – Ralph Waldo Emerson

Table of Contents

1. Design Process Improvement

Like modern physicists, Buddhists see all objects as processes...

– Fritjof Capra in "The Tao of Physics"

In this chapter, I introduce the notion of design process improvement. After presenting the motivation for viewing design as a process, I explain what it means to view design as a process, and contrast it with the conventional project view of design. I also overview two specific techniques for design process improvement called sequencing and overlapping with applications to design problems from the industry and literature.

1.1 The Process Viewpoint

The dictionary defines a process as "a series of actions or operations conducting to an end" [1][1]. In this thesis, the design of electro-mechanical products is studied as a process with the mission of establishing and improving the approach towards the desired end.

The process notion is well established in manufacturing, and is becoming increasingly popular in software development. At the beginning of this century, manufacturing went from the craft or job shop mode to a more organized mode by a systematic division of larger tasks into smaller operations, division of labor, and automation [4]. In software development, the need to improve performance has resulted in the systematic study and classification of the development process, and has led to the transformation of an initial process with no orderly progress into one in which comprehensive measurements were available and continuous improvements possible [2]. Although it is not clear if the same path should be followed, it is indeed true that the same end point – a more organized and better performing process – is desired in electromechanical product design.

Design of complex products requires resolution of a multitude of conflicting considerations, thousands of decisions, and reasoning and search

[1]Other researchers have used similar definitions; see [2], and [3].

in multiple domains; the very scope and complexity necessitates that knowledge and data be distributed, and design be divided into a set of interacting tasks or information exchanging activities[2]. To view design as a process is to focus on this interaction and flow of information among activities (in this set) with the aim of integrating the output of the collection of activities.

1.2 Performance Orientation

Better design processes organize the timing and information relationships among the constituent activities more effectively. Measures of effectiveness include "design quality", calendar time to complete the design, total person-hours, and cost of design efforts (such as prototype material and machining cost). With manufacturing performance differences among product development firms shrinking, the pressure is greater today to gain competitive advantage through product design and development.

Several researchers have emphasized the importance of improving performance. A major fraction of a product's manufacturing costs – anywhere between 60-80% – are determined at the product design stage and it has been suggested that the decisions made at this stage be of higher quality [6]. Clark et. al [7] estimate that each day saved in introducing a small car to market represents a million dollars in profits. A late entrant to the market loses revenue not only due to lost sales, but also due to lost margins that early entrants receive by charging higher prices. Millson et al. [8] note that higher performance companies can reap huge cost benefits by more creative and efficient utilization of resources, smaller work-in-process and reduced development time.

Although there exists a large body of work on *why* performance should be improved, there is not sufficient research on *how* performance may be improved. In this thesis, the notion of design process is advanced as a means to improve development performance. In the following section, I briefly overview conventional perspectives on design (details are offered in the next

---

[2]After recognizing the fine distinction between the terms task and activity made by some authors such as McGrath et. al [5], I use these terms interchangeably in this thesis.

chapter). Next, I discuss how the design process perspective departs from the conventional view. Finally, I present some specific ways of representing and classifying the design process, and introduce two methodologies for design process improvement called sequencing and overlapping – illustrated with application to design problems from the literature and industry.

1.3 The Conventional "Project" Viewpoint and Its Weaknesses

Conventional ways of conceptualizing design lack both a context and a mission orientation [9]. Traditionally, design is viewed as a completely new, craft activity or a project performed by a single designer with the aid of a computer and/or other tools. Simplistic assumptions are made about the time and information relationships: a) that activities occur in a pre-determined sequence, b) that activities are points in time when viewed as information sources, c) that design information is "released" or exchanged in a finalized form– if the exchange is modeled at all (with the single designer view the exchange does not arise), and d) that activities are done exactly once.

In practice however, the following (contrasting) situation prevails:

- Designers, being a part of the product development enterprise, interact frequently with other development functions such as marketing, manufacturing, suppliers, and purchasing.
- The product developed may be unique in the different projects, but *the process of designing is a sequence of individual activities that may not vary much across projects* (described in more detail in Chapter 2 with reference to studies reported in [5, 10]).
- Activities need not occur in a predetermined sequence, and may be profitably resequenced or even overlapped.
- Activities are not always points in time, and indeed may produce or absorb information at distributed times or even continuously.
- Activities may be iterated – sometimes due to the circular information dependencies inherent in the product physics, and other times due to the need to start with preliminary information, and to incorporate changes.

1.4 Focusing on Information Exchange

To model this alternative scenario, it becomes necessary to consider the context – the existence of other design activities and product development functions, and the interactions among them. The interactions, manifested in information exchanges, impact how the product development process is conducted (sequentially or concurrently), and indirectly determine the development performance. The interactions are, however, often complex – the right information requirements, and their utility, availability and influence, are not easily understood in many product development organizations. In order to study design as a process it is necessary, therefore, to focus on the existing and desired interactions among the constituent activities – the information that development activities require from each other, the utility, availability and impact of such required information, and the timing and content of information exchanged in an existing process.

A major reason for studying design as a process is to clarify the existing interactions and modify them into desired interactions, thus improving product development performance. To see how performance may be improved, consider two different cases where the current process impedes performance. In the first case, certain design decisions made early in the design process (such as styling decisions) impact the downstream design activities (such as manufacturing) adversely, and the quality of the designed product suffers from the downstream perspective. Proper understanding of the design interactions helps ensure that the decision is made in accordance with the downstream activity, thereby improving quality of the product developed. In the second case, a downstream activity waits for a long time for some design information which is not essential to begin the activity. Again, recognition of the necessity for and influence of design information can ensure that the activity starts with a preliminary version of the information, leading to a process with improved time performance.

In light of the above, we see that process viewpoint leads to a totally different outlook, and brings up several new process management issues:

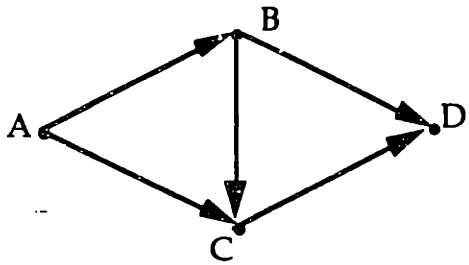
- What information is "time-critical" to a design activity – without which the activity will be inevitably delayed?

- What information is "content-critical" to a particular design activity – changes in whose content can cause substantial rework? [11]
- How can the design process be shortened and quality be improved?
- What is the best sequence in which the activities may be performed?
- How may activities be overlapped, and what risks are involved?

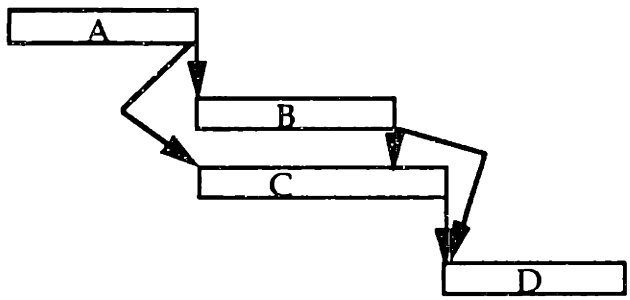
As will be seen in this thesis, some interesting design process management options emerge: (i) Alter the time availability of content-critical information causing some loss of flexibility for information generating upstream activities, but saving precious time and effort for the downstream activities; (ii) Alter the content of time critical information by disaggregating the exchanged information, "releasing" preliminary versions to the downstream activities earlier, and updating it later with the finalized version. This may require added downstream iterations and development effort, but may reduce downstream wait time and development calendar (lead) time.

Representations where Activities are Explicit

To explore these options, it is necessary to represent properties of exchanged information, such as its availability and its influence on design activities. Traditionally, activities have been the basic units of the design process and design information is implicitly defined as what activities exchange. Due to this implicitness, the above mentioned properties of the information exchanged go unrepresented. Examples of activity-based representations include (see Figure 1.1): (a) Activity on Node (AON) project management description in which the edges are labeled with precedence (time) constraints among activities, (b) Gantt Charts which show relative position of activities on a time scale and (c) Binary Design Structure Matrix (DSM) in which the rows and columns of the matrix are activities and the matrix entries denote presence or absence of information exchange among activities. Note that the common feature of the above examples is that they make the activities explicit, and denote the information exchanged, such as drawings or mockup, implicitly using marks or arrows. (It is noted though, that the above mentioned examples differ amongst themselves in the type and quantity of information they represent.)



(a) AON Diagram



(b) Gantt Chart

	A	B	C	D
A	•			
B	x	•		
C	x	x	•	
D		x	x	•

(c) Binary DSM

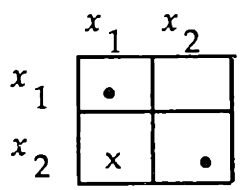
Figure 1.1: Activity-Based Representations of the Design Process

Examination of product development practices in industry indicates that (i) activity-based representations are intuitive, easily understood, and widely used (ii) the definition of activities is nonunique (different groups define activities at different levels of detail based on their understanding of the development process), and (iii) the level of detail impacts whether the interactions are hidden or visible. To manage the development process, it is desirable that the interactions be visible which often, but not always, requires larger level of detail of activities. But the greater the level of detail, larger the effort required to represent and understand the process.

Making Information Exchange Explicit

Alternatively, the information exchanged among the development activities can be made explicit. Figure 1.2 provides an example of this situation where the mark in the matrix could represent properties of

information exchanged such as coupling, availability of information, certainty, time constraints etc. This representation is equivalent to viewing the activities as information processors and the development itself as a process of generation and transformation of information [12]. Information exchanged is less subjective than activities because it is a snapshot in time, and can therefore be defined as the event that occurred at a particular point in time. It should be noted, however, that granularity still plays a role; if for example, in an organization the styling and engineering design functions are lumped together as a "design" function, then the information exchanged between styling and engineering design will not appear in the representation of the development process.



Example:

x_1 = corner radius

x_2 = maximum stress

x = time taken to run the finite element program that computes maximum stress from corner radius

Figure 1.2: Information-based Representation

The activity-based representations tell us which activity interfaces need to be focused. Once this is identified, it becomes necessary to make the information exchange between these activities explicit in order to identify improvements in the design process. This thesis focuses on modeling the properties of the information exchanged between time-critical activities.

1.5 Patterns of Execution and Process Performance

Using both activities and information transfers, it is possible to classify the pattern of execution of design processes into sequential, parallel and overlapped processes. As seen in Figure 1.3A for a process with two activities, sequential processes involve downstream activities following their upstream counterparts in a phased fashion. In parallel processes the coupling between

activities is removed, and the activities can occur simultaneously in time. In overlapped processes, the coupling is preserved, but the activities are carried out simultaneously by frequent information exchange.

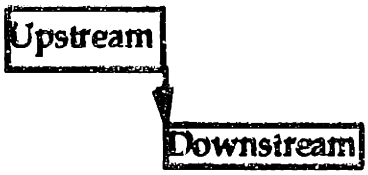


Figure 1.3A
Sequential Process

The downstream activity begins only after receiving finalized information upon completion of the upstream activity.

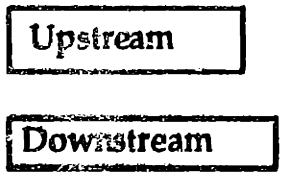


Figure 1.3B
Parallel Process

The coupling between upstream and downstream activities is removed resulting in a parallel process.

Figure 1.3C
Overlapped Process

The coupling is preserved and the activities are overlapped by improved information exchange.

It is noteworthy that the patterns of the execution are *choices that firms make to conduct the process while satisfying the couplings among activities.* The couplings may obviate some choices and may require tradeoffs among the performance parameters. Although executing a process in a parallel or overlapped fashion would help in reducing development time, decoupling activities to parallelize them may involve a fundamental redefinition of the product architecture resulting in a less integrated product. Overlapping activities may sometimes cause considerable rework in the downstream activity or worsen product quality. In the interest of managerial decision making about the pattern of execution, it is necessary to understand the connection between the pattern of execution and development performance.

1.5.1 Objectives of This Thesis

In this thesis, I investigate the relationship between the patterns of execution and the process performance – towards developing methods to alter the pattern of execution and improving development performance. The focus of my attention in this thesis is on product development processes which are nominally sequential (in which the flow of information is

predominantly sequential, as in Figure 1.3A; for a detailed description see Chapter 2). Previous study of industrial product development processes by several researchers including Eppinger et. al [13], Osborne [14], and Black et al [15], suggests that many processes in practice are nominally sequential -- justifying and warranting this focus. The question this thesis addresses may be posed as: How can such nominally sequential processes be analyzed and their performance improved by either resequencing the activities or overlapping them? As we will see in the next section, there may be several ways in which design processes may be sequenced and the different sequences result in products of different quality . By resequencing, quality of the product developed may be improved. To accelerate such processes, the sequential activities need to be overlapped – which may require ways to relax the precedence constraints among the nominally sequential activities. I present a framework and a methodology to overlap design activities in the subsequent sections of this chapter.

In the remainder of this chapter, I overview the sequencing and overlapping problem – which are dealt with in detail in the subsequent chapters.

1.6 The Sequencing Problem

If a process consists entirely of decoupled activities, then the constituent activities can be done in parallel. However, technical and organizational considerations impose couplings among activities. Although such couplings often translate into (directional) precedence constraints among the activities, there are many problems in which the couplings may not be directional. As an example, consider a cross-functional design decision–making problem where an artifact's size parameters (say $x_1, x_2, ..., x_m$) need to be decided while considering several product characteristics, such as functional performance, space and cost. Let the size decision–making based on each of these considerations be referred to as tasks, and let there be n cross-functional decision making tasks ($T_1, T_2, T_3, \ldots, T_n$). In this case, tasks $T_1, T_2, T_3, \ldots, T_n$ are coupled – they affect each other in that if the design decision is made based on functional performance, it impacts the amount of space the product occupies – but the tasks do not necessarily have to precede each other.

Although there exist no precedence constraints among the activities, sequential decision making (such that predecessors in a sequence decide all decisions that are of interest to them and which are yet undecided) is a way of ensuring that the constraints imposed by the couplings are satisfied – which simultaneous decision making by all tasks cannot satisfy without iterations. In this case, the sequential order ensures that each of the design parameters takes on a unique value. It is possible to perform the design tasks in several possible sequences. There exist efficient methods to resequence tasks in design problems where the tasks are related by strict precedence relations (see [16, 17]) but, to the best of my knowledge, there are no methods to sequence activities in design problems, as described above, where the tasks are coupled but not strictly constrained by precedence relations.

In each of these sequences, downstream tasks lose certain degrees of freedom to their predecessors resulting in a "loss in quality" for the design solution from the downstream perspective. The primary difference among the different sequences is in the design degrees of freedom decided in the upstream stages of the decision process. Decision sequences that result in a good quality product typically have the upstream decisions not causing too much of a quality loss for the subsequent decision makers.

In bad decision sequences, the decision-making capability of the downstream stages is severely impaired due to the *choices* imposed by the prior decision makers. For example, in a camera design project, an early decision to save cost by integrating several parts of a complex assembly may lead to increased complexity, lead time, and cost later in molding the integrated component. I have developed (i) a metric to quantify the quality loss suffered by downstream decision makers due to the decisions made in the upstream stages and (ii) a procedure to determine the optimal sequence that involves the minimum quality loss.

I define *quality loss* as the loss in the design solution due to the sequence constraints imposed by the design process. In the absence of sequence constraints, let a task T_i result in the "independent" output metric J_i^*. In a sequence φ, executing task T_i results in the sequential output metric J_i^{φ}. Quality Loss of the task T_i in the sequence φ is then defined as,

$$QL_i^{\varphi} = |J_i^{\varphi} - J_i^*|$$

The optimal sequence is defined as the one with the lowest total quality loss among all sequences (the total is weighted if some tasks are more important than the others). As an example, consider the following adapted parametric design problem taken from [18], which involves the design of a dc motor.

1.6.1 DC Motor Example

The example involves determining the parameters of a dc motor while maximizing the torque generated by the dc motor (performance), minimizing the area occupied by the stator (size) and minimizing the cost of materials (sum of the area occupied by the steel portion of the rotor and area of copper). The variables to be decided and their bounds are given in Table 1. The parametric design relations based on the physics of the dc motor are shown in Table 2 (formulated using constraints from [18]. As can be seen, each lifecycle issue (such as function, space, materials) has its associated design objective.

Decision Variable	Symbol	Bounds
Armature diameter	ad	$10 \le ad \le 12$ (inches)
Motor inner diameter	id	$0.1 \le id \le 3.0$ (inches)
Motor outer diameter	od	$20 \le od \le 24$ (inches)
Diameter of windings	dw	$0.01 \le dw \le 0.2$ (inches)
Current density	cd	$0.1 \le cd \le 50.0$ (amp / in^2)
No. of armature windings	nw	$1 \le nw \le 1500$ (turns)
Thickness of magnet used	tm	$0.05 \le tm \le 1.0$ (inches)

Table 1. Design Variables for a dc motor

Task	Task Description	Analytical Forms (Minimizations)
T_1	Maximize Torque	$J_1 = -1.57\ cd\ dw^2$
T_2	Minimize Space	$J_2 = 0.785\ (od^2 - ad^2) - 0.26\ nw\ dw^2$
T_3	Minimize Material Costs	$J_3 = 0.785\ (ad^2 - id^2) - 2.1\ ad\ tm + 0.785\ dw^2$

Table 2. Objectives as Functions of Design Variables

Table 3 shows the results if the design decisions were made based on each lifecycle criterion independently; the outputs under these circumstances are called the independent outputs and the decisions, independent decisions. Note that the independent decisions do not have a unique value; for example, while the designers executing tasks T_1 and T_2 independently drive the variable *dw* to the value of 0.2, the designer entrusted with T_3 drives *dw* to the value, 0.01. The product designed can have only one value for *dw* .

Task	Independent Decisions and Outputs
T_1	$dw_1^* = 0.2$; $cd_1^* = 50$; $J_1^* = -3.14$
T_2	$ad_2^* = 12$; $od_2^* = 20$; $dw_2^* = 0.2$; $nw_2^* = 1500$; $J_2^* = 185.3$
T_3	$ad_3^* = 10$; $id_3^* = 3$; $dw_3^* = 0.01$; $tm_3^* = 1.0$; $J_3^* = 50.4$

Table 3. Independent Decisions and Outputs

Making the decisions in a sequence is one way of satisfying the couplings. However, when the decisions are made in a sequence, only one of the designers gets to decide the value of *dw* ; subsequent decision makers are constrained by this value of the variable *dw*. An example of one sequence is given below (where the weights for the total quality loss are taken to be the inverse of the magnitude of their independent solutions, i. e. $w_i = 1/ J_i^*$).

The sequence $\{T_1, T_2, T_3\}$ produces the following results:

Stage 1 Minimize $J_1 = -1.57\ cd\ dw^2$

Decision: $cd = 50$; $dw = 0.2$; $J_1 = -3.14$; $w_1 QL_1 = 0.0$

Stage 2 Minimize $J_2 = 0.785\ (od^2 - ad^2) - 0.26\ nw\ dw^2$

subject to $dw = 0.2$;

Decision: $ad = 12$; $od = 20$; $nw = 1500$; $J_2 = 185.3$; $w_2 QL_2 = 0.0$

Stage 3 Minimize $J_3 = 0.785\ (ad^2 - id^2) - 2.1\ ad\ tm + 1.57\ dw^2$

subject to $ad = 12$; $dw = 0.2$;

Decision: $id = 3$; $tm = 1$; $J_3 = 80.75$; $w_3 QL_3 = 0.603$

Total Quality Loss = $w_1 QL_1 + w_2 QL_2 + w_3 QL_3 = 0.603$

It would be useful to resequence the existing process and reduce its quality loss as much as possible (preferably to the level of optimal sequence). To illustrate the utility of quality loss for design process improvement, I consider design problems where each individual design task can be modeled as a nonlinear program (with continuous objectives and design constraints). I introduce the notion of sequence and task invariance which correspond to the cases of the design decision variable value being independent of the sequence and the task position in a sequence, respectively. I develop sufficient conditions to identify sequence and task invariant variables, which help simplify the optimal sequence determination problem. Further simplification is achieved by partitioning the design variables into exclusive groups, which are groups of design variables decided by the same combination of tasks. When all the variables in a design problem are sequence or task invariant, I show that the optimal sequence corresponds to the shortest path in a network of quality losses. The details are in Chapter 3, where I also offer a step-by-step approach to identify the optimal sequence without the likelihood of exhaustive enumeration of all sequences.

1.7 The Overlapping Problem

In the previous section, I considered the problem of resequencing and improving performance (specifically, product quality) when activities are coupled but not constrained by precedence relations. Often, due to technical and organizational considerations, there exist precedence or information exchange requirements among activities. Because of these requirements, activities have been executed in a nominally-sequential fashion, as in Figure 1.3A. To accelerate such processes, we are interested in determining how to overlap the activities, as in Figure 1.3C, amidst precedence/information flow constraints.

To overlap activities, upstream capabilities and downstream information needs must be well understood; it is likely that upstream information may have to be frozen early if the information happens to be content critical to the downstream activity. In other cases, the downstream resources may be committed based on preliminary upstream information. However, if downstream iterations are started prematurely, and if the impact of subsequent changes in the exchanged information on the downstream

activity is substantial, then overlapping runs the risk of increasing the number of downstream iterations and even the development lead time. If, on the other hand, the exchanged information is not used until it is finalized, then the process is only as fast as a sequential process. The problem of increasing the overlap in an existing process while shortening its duration is called the *overlapping problem*.

To aid in overlapping the nominally-sequential activities, I develop a framework in which the exchanged information is classified in two key dimensions: upstream evolution and downstream sensitivity. Below, I first briefly discuss these dimensions and then present the framework that helps determine how to overlap the activities.

1.7.1 Upstream Information Evolution

It would be risky to start downstream activities with preliminary upstream information or to freeze the upstream information early without knowing how close the upstream information is to its final form. The rate of evolution (or evolution, in short) of the upstream design information is indicative of how easy it is to finalize the design information at various points in time during the upstream activity's progress. The evolution of the upstream generated information is said to be *rapid or fast* if the information gets close to its final form rapidly, and is capable of being frozen and passed downstream early in the upstream process without much quality penalty for the upstream activity. The evolution is said to be slow, however, if finalizing upstream information early in the upstream process is either impossible or entails a huge quality penalty for the upstream activity. We will limit ourselves to conceptual definitions of evolution in this chapter; more detailed, operational definitions of evolution are offered in Chapter 4.

It is noteworthy that evolution refers to the time availability of the upstream information – information that evolves faster is available at a given level of completeness earlier to the downstream activity than information that evolves to the same level slowly. The amount of change the exchanged information undergoes is also a function of its evolution; faster the evolution, smaller is the amount of change the exchanged information undergoes at the end of the upstream activity. Overlapping downstream activities is easier when the evolution of the upstream design

information is fast than when it is slow. The upstream information evolves slowly for example, when (i) the information pertains to a component which houses or interfaces with several other components each of which is likely to change, (ii) it is impossible to obtain some related information provided by an extraneous factor until late in the upstream process, or (iii) generating the upstream information involves solving a complex and coupled problem. It is clear that when the evolution is slow, overlapping activities entails considerable risk.

1.7.2 Downstream Sensitivity

Evolution of the upstream information describes whether the exchanged parameter undergoes small or large changes during the upstream activity. The consequence of the upstream changes to the downstream activity is captured by downstream sensitivity, which measures the duration of downstream work required to accommodate changes in the upstream information. Downstream sensitivity is said to be high (low) if the downstream work required to incorporate small (large) changes in the upstream information is large (small). The lower the downstream sensitivity, the easier it is to incorporate upstream design changes. The reader may observe that downstream sensitivity is indirectly a measure of the "content-criticality" of information referred to in [11].

It is noteworthy that evolution and sensitivity, which are properties of the upstream and downstream activities respectively, can to some extent be altered by better communication and computer tools. As I discuss later with reference to the development of a pager at Motorola, communication between the upstream and downstream activities in the studied process helped reduce the downstream sensitivity (effect of design changes on the downstream activity). Also, computer tools can ensure that the upstream design information evolves faster and the effects of changes on the downstream activity are reduced.

Earlier, I observed that while offering the potential of reduced development time, overlapping runs the risk of increased downstream iteration if downstream acts on the received information prematurely. When should downstream act on upstream information? How should the activities be overlapped when the downstream activity cannot work on preliminary

information? When is it better to freeze the upstream information rather than transfer it in a preliminary form to the downstream activity? The combination of the upstream evolution and downstream sensitivity values offers the answers to these questions as explained below.

1.7.3 The Evolution–Sensitivity Framework

In Figure 1.4, I show the four extreme situations likely to arise – when the upstream evolution is rapid or slow and when the downstream sensitivity is high or low. The activities are to be overlapped differently for each combination of evolution and sensitivity.

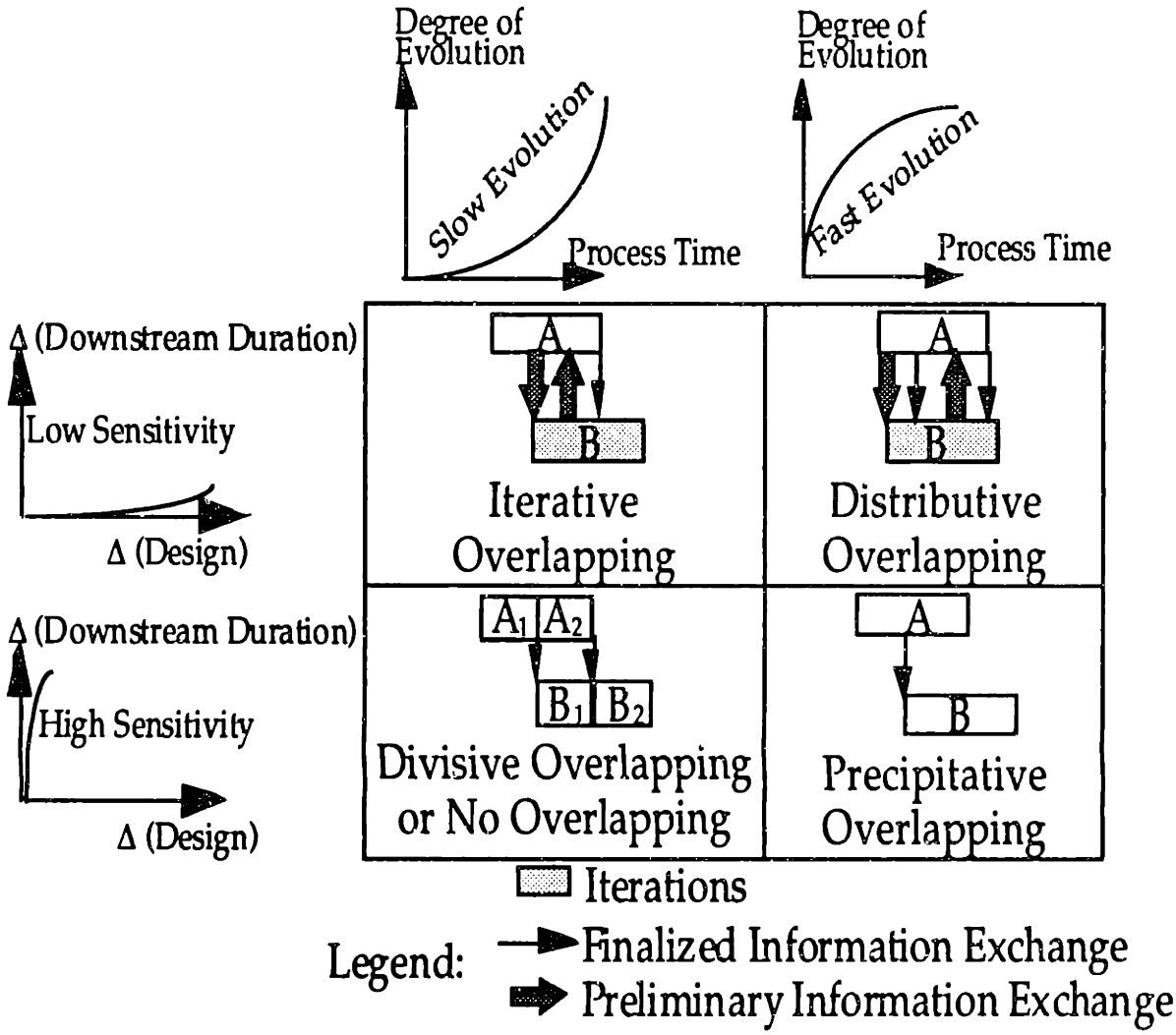


Figure 1.4: Type of Overlapping based on Evolution and Sensitivity

• When the sensitivity is low, it is possible to commit downstream resources based on advance upstream information. Because even if the changes are big, their effects on the downstream activity are not. If the upstream information

evolves slowly – it cannot be finalized until late into the upstream process – then the overlapping is said to be *iterative*, under which the downstream activity is started early with preliminary information and design changes are incorporated in future downstream iterations. In this case, the upstream design information is not finalized until the nominal completion of the upstream activity, because doing so may result in a large quality penalty for the upstream activity due to the slow evolution. Because of the low sensitivity of the downstream activity, the future iterations are not likely to be rework intensive. The models presented in Chapter 5 would help find when preliminary information can be committed for downstream iterations, how many iterations would result and with what lead time.

- The opposite case is when the downstream sensitivity is *high*, but the upstream information evolves *rapidly* (information is capable of being finalized early in the upstream activity). In such a case the exchanged information is to be precipitated to its final value/form at an earlier point in time. In other words, the upstream problem solving is accelerated and information frozen ahead of the normal time of freeze. This is called precipitative overlapping and would help reduce development time by starting the downstream activity earlier- but with *finalized* upstream information. Note that there are no subsequent downstream iterations. It may result in some quality loss to the upstream activity because it loses the opportunity to make changes until its original completion time.
- Consider the case when the downstream sensitivity is *high* and the upstream evolution is *slow*. Here, it is neither desirable to start downstream activity with preliminary information nor feasible to precipitate the exchanged information to its final form at an earlier point in time. In such a case, the exchanged information is disaggregated into components to see if any of the components evolve fast or if transferring any of the components in their advance form to the downstream activity is practical. Often the evolution and sensitivity of the components may be different from the whole information. Because the disaggregation is also based on physical or functional division of the upstream and downstream activity, this approach is called divisive overlapping. If neither of the parts evolve fast, nor can they be used by the downstream activity in an advance form, then no overlapping is recommended with the current evolution and sensitivities.

• The last scenario occurs when both the upstream information evolves *rapidly* and the downstream sensitivity is *low*. In such a case, it is possible to both start downstream activity with advance information and precipitate the exchanged upstream information to its final form. Because the impact of overlapping is distributed between the upstream and downstream activities (unlike in other cases), this situation is called distributive overlapping.

It is noteworthy that the different types of overlapping result in different trade-offs among the performance parameters. In iterative overlapping for instance, downstream effort is traded-off against lead time, while in precipitative overlapping, upstream quality is traded-off against lead time. The trade-offs, discussed in more detail in Chapters 4 and 5, help unify the different types of overlapping. Also, the reader might observe that the project management paradigm, which assumes the one-shot transfer of finalized information, corresponds to only one of the four possible cases; that of slow evolution and high sensitivity requiring finalized information release. The framework presented above expands the domain of project management by considering three other combinations of evolution and sensitivity.

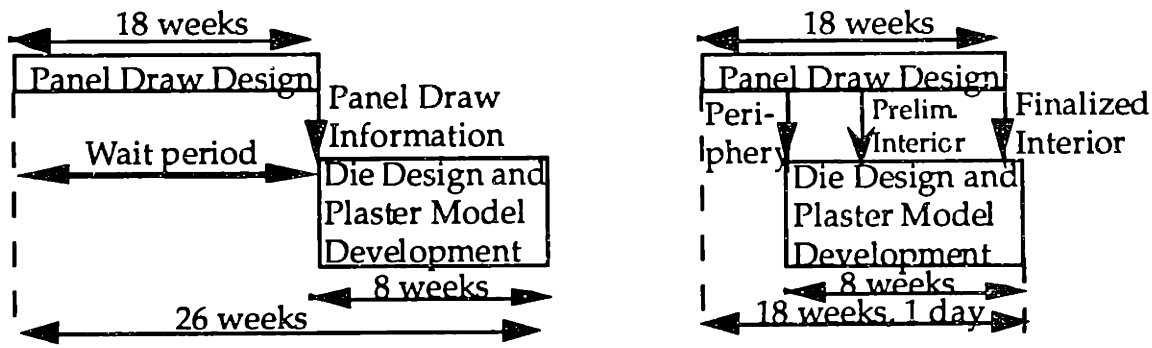
In the next section, I illustrate applications of the framework to three examples from the industry – involving the development of automotive doors and instrument panels at Chrysler, and pagers at Motorola. The door and instrument panel applications illustrate how a process may be improved by overlapping, while the pager example is an interpretation of an already overlapped process with the above framework.

1.7.4 Overlapping the Door Development Process

Door development at Chrysler is a complex process lasting over a year and involving several functions (styling, engineering, and manufacturing, to name a few). The details of door development are given in Appendix A2. Analysis of the existing process indicates that two long, adjacent phases in the critical door development path involve door panel draw design (lasting 18 weeks) and the plaster model development of dies (8 weeks) (see Figure 1.5).

As shown in Figure 1.5 (a), the process engineers wait for about 18 weeks to receive the panel draw information (shown in detail in Figure 1.6) which contains details of all surface formations both on the outer periphery and the interior of the door panels. Interviews with product designers indicates that

the panel draw information, especially in the panel interior, is preliminary until the panel draw design (upstream activity) is complete, so the evolution of the exchanged information can be called slow. Also, process engineers observe that changes in the panel draw information, especially at the periphery, can have a huge impact on the plaster model development activity. (As explained in Appendix A2, changes in the panel periphery may require that the die design be repeated and the plaster model rebuilt). So the sensitivity of the downstream activity (plaster model development) to the draw information is high. The draw information exchanged falls under slow evolution and high sensitivity category (lower left quadrant in Figure 1.4). According to the framework I disaggregate the information exchanged into components with different evolutions and sensitivities.



(a) Process without Overlap (b) Process with Overlap

Figure 1.5: Interaction between Door Panel Design and Die Development

As I hinted above, the sensitivity of the downstream activity to the panel periphery draw design information is different from its sensitivity to the interior draw design information (as the appendix A2 describes, changes in the interior such as changes in formations can be incorporated by mere touch-ups). Due to this difference in sensitivities, I examine the nature of evolution of the panel periphery and interior to see if they are different. It turns out that there is indeed a difference: while the panel periphery evolves quite fast, the panel interior evolves rather slowly; the door inner panel interior involves the packaging of the various components such as speakers, wiring, door lock etc. while the outer panel interior involves the door handle design – a slow process owing to the need to differentiate the visible door handle from other competitive products, and also due to the complex problem solving (described in Chapter 5).

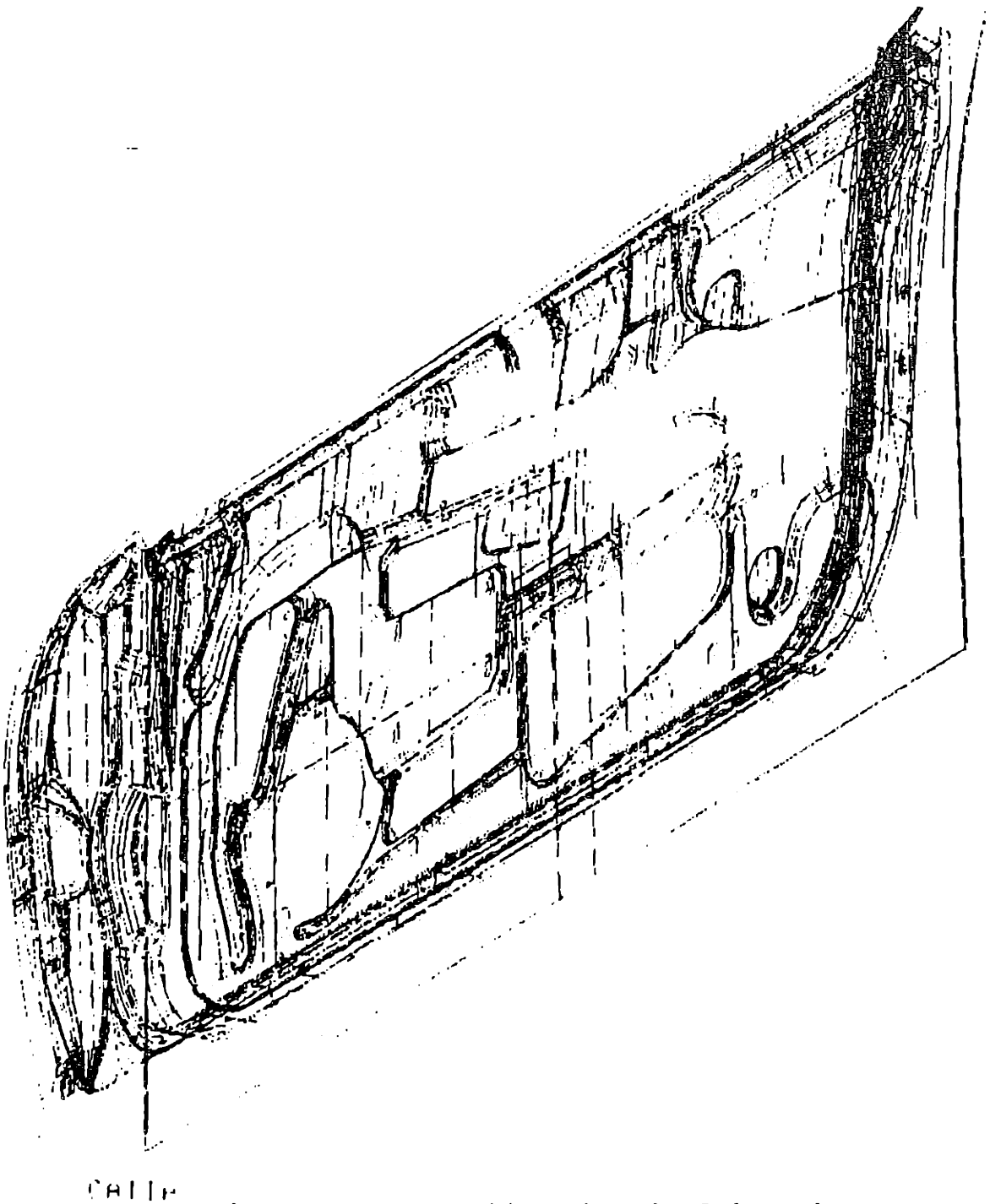


Figure 1.6: Door Panel (Draw) Design Information

The panel periphery, on the other hand, evolves relatively fast as it does not interface with many change-prone components. So the panel draw information can be disaggregated into the panel periphery draw information (whose evolution is fast and sensitivity is high) and the panel interior draw information (whose evolution is slow and sensitivity is low). Such a

disaggregation also suggests the division of the upstream and downstream activity: panel periphery draw design and plaster die model development and, panel interior draw design and model development.

Using the framework, it is seen that the panel periphery draw design and model development activities, coupled by the fast evolving, highly sensitive periphery draw information, should be precipitatively overlapped: the problem solving should be accelerated and the panel periphery should be frozen early. On the other hand, the panel interior draw design and die model development activities, coupled by the slow evolving, low sensitivity interior draw information, should be iteratively overlapped: model development should start with preliminary information about the panel interior and incorporate changes in subsequent iterations. In Chapter 5, I model this situation to determine when the panel interior information is trustworthy enough for downstream action and how many iterations would result. The application of the framework and models suggests a reduction in lead time of eight weeks (from 26 to 18 weeks).

1.7.5 Pager Development Process at Motorola

In this section, I will describe how the pager development process at Motorola can be analyzed using the above framework. Due to confidentiality reasons, I cannot disclose the exact geometric form of the pager studied until the end of 1993; for our purposes the product studied can be thought of as a rectangular block with length l, width w, thickness t, and corner radius, r. Figure 1.7 shows a representative pager and a pager cross section. The process of development of a pager is described in greater detail in Chapter 4. Here I focus on overlapping the two adjacent phases in pager development. industrial design (the upstream activity) and engineering design (the downstream activity).

Engineering design of the pager involves the design of the mechanical and electrical components (receiver and decoder boards, walls, ribs etc.) and requires pager external dimensions and shape details determined by the industrial designers. From experience with designing pagers, it is known that the pager dimensions evolve fast, constrained by the competition, target market and technology and determined by volume studies and human factor studies done by industrial designers. On the other hand, the shape details of

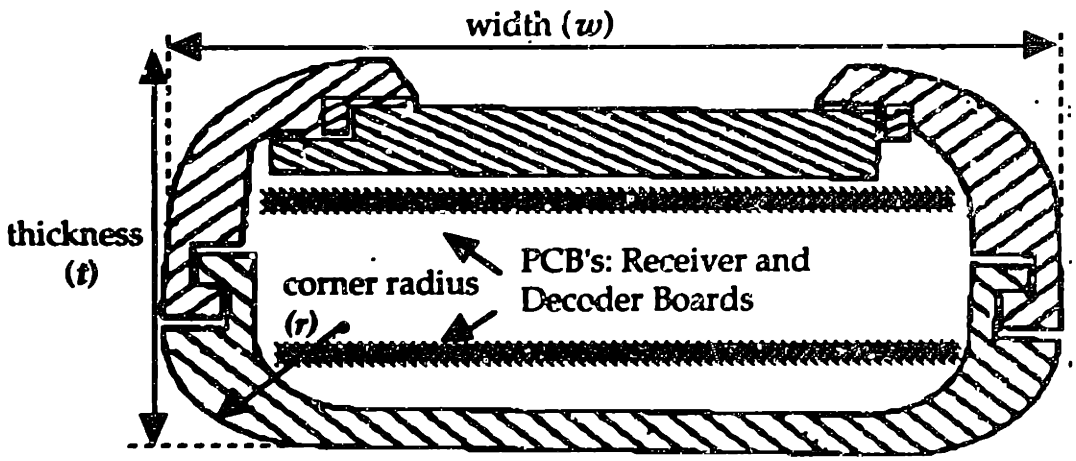


Figure 1.7: Representative Pager and Pager Cross-section

the pager, such as the corner radii, are known to evolve slowly; if any feature of the product changes, the shape details need to be changed to create an "integrated design". This is confirmed by data from a recently completed process which shows that the radius had changed as much as 30% near the end of the design process.

It is noteworthy that the engineering activity is very sensitive to changes in dimension. Not only is the layout of the components affected but also the choices of components, as smaller components (manufactured with more recent or not yet available technology) are needed to meet the shrinkage in size. Changes in the shape are of a different kind. With poor communication, changes in radii may affect the layout of components (as the wall would interfere with some components) but in the studied process communication was good. The engineers anticipated changes in the radii and placed all the tall components in the middle. Since changes in the corner radii do not affect the center, the effect of changes on the component layout was reduced and the shape details had low sensitivity.

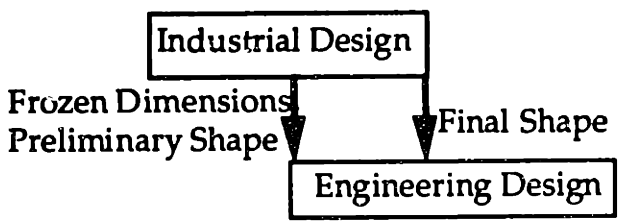


Figure 1.8: Overlapping Industrial Design and Engineering Design

Thus engineering requires two types of information from industrial design: pager dimensions whose evolution is fast and sensitivity is high, and shape details whose evolution is slow and sensitivity is low (owing to good communication). To overlap the engineering and industrial design functions in the studied process, engineering started with preliminary values of corner radius, and (close to) frozen values of pager dimensions (see Figure 1.8). Overlapping engineering and industrial design by early freeze of the highly sensitive, fast evolving pager dimension falls under precipitative overlapping in which the pager dimensions were precipitated to their final value to prevent enormous engineering rework. Changes in the shape details, on the other hand, were incorporated in future iterations due to their lower impact. This exemplifies iterative overlapping.

1.7.6 Instrument Panel Application

Instrument panel development is another example of a process which can be overlapped by classifying the information into parts with different evolution and sensitivity. Details of the development process are given in Appendix A3.

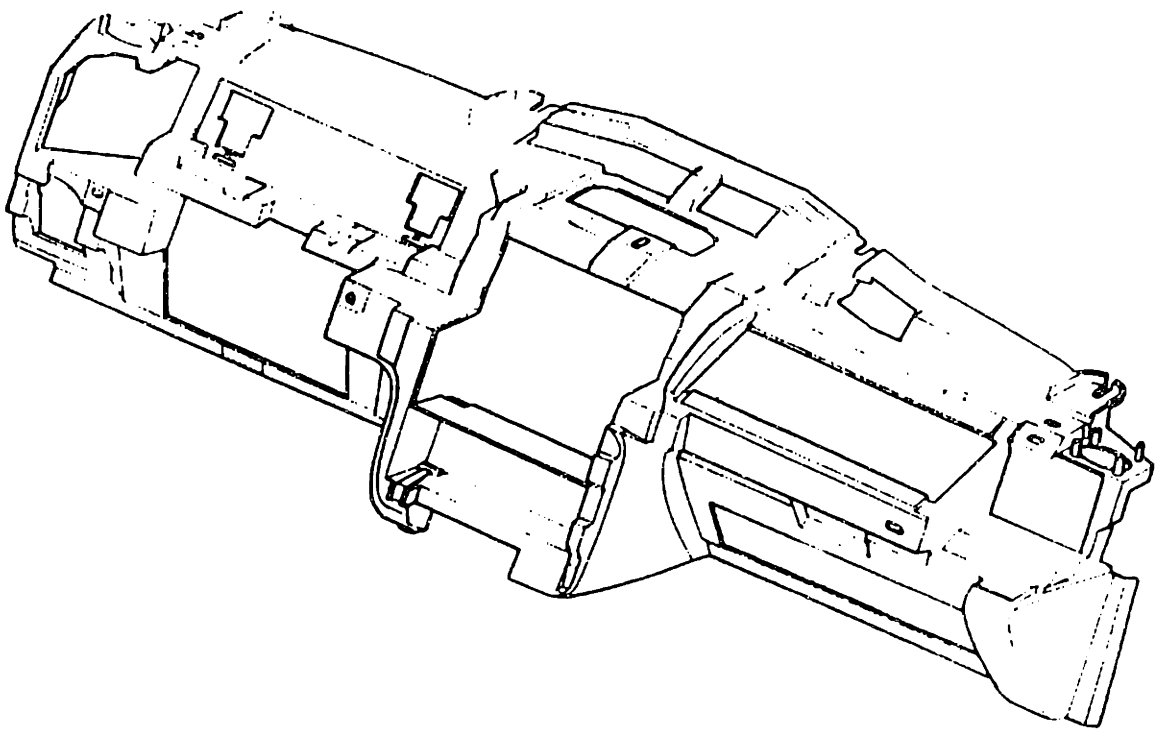
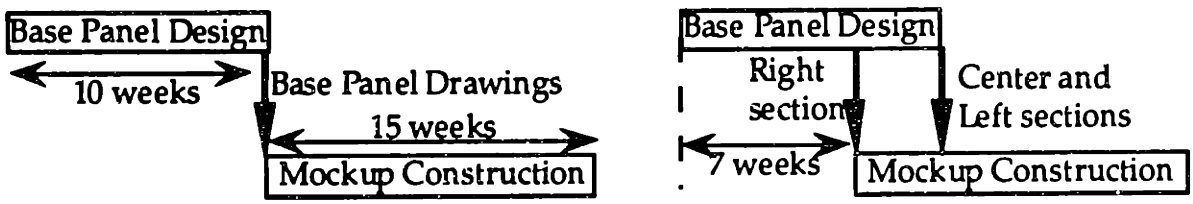


Figure 1.9: An Automobile Base Panel

Figure 1.9 shows the base panel which is the most complex component in the instrument panel system because most other components, such as the steering column, instrument cluster, air bags, trim panel, and the glove box, mount onto the base panel. Changes in any of these components causes changes in the base panel. Unlike the door panels which are made of sheet metal, the base panel is an injection molded part. Study of the development process of a base panel reveals that there are two lengthy phases in the critical path of the process, called the "design phase" and the "mockup construction phase". During the design phase, which lasts 10 weeks, the spaces, clearances

and the attachment schemes for each of the instrument panel components are defined. Mockup construction, which takes 15 weeks, involves the development and assembly of fiber glass parts from a wooden mold of the base panel. Being sequential, these two stages take 25 weeks. In the interest of overlapping the two stages, I examine the evolution and sensitivity of the information required by the downstream activity (mockup construction) from the upstream activity (design).

Interviews with the base panel designers suggests that the base panel information changes until the very end of the design phase, so the base panel design evolves slowly. The craftsmen who construct the mockups indicate that changes made in the base panel will have an enormous effect, often requiring that the wooden prototype be started all over again. Thus the exchanged information is of the slow evolution/high sensitivity category. The lower left quadrant of the framework suggests that the information be disaggregated. We examine if parts of the base panel differ in evolution or sensitivity.



(a) Sequential Process (b) Overlapped Process

Figure 1.10: Overlapping Adjacent Phases in Instrument Panel Development

Indeed, the passenger section (right side) of the base panel evolves faster than the driver (left) and center sections. The steering and instrument clusters, which are change-prone during the design process, drive the changes in the driver section to which they mount. The changes in the location of the ducting for heat, ventilation and a/c causes changes in the center section. In comparison to the other two sections, the passenger section of the base panel is relatively free of change, and can afford to be frozen early. The sensitivity of mockup construction to the design information about all three sections (left, center and right) is high.

With disaggregation, we find that the design and mockup stages exchange the fast evolving base panel passenger section, and the slow

evolving center and driver sections (all of which are of high sensitivity). The framework suggests that the right section be frozen early and the design and mockup stages be precipitatively overlapped (see Figure 1.10b). As I will show in Chapter 5, this can potentially lead to three weeks savings in development time. Further disaggregation of the left and center sections may be tried to see if parts of these sections evolve faster or are of low sensitivity .

In subsequent chapters, I develop the above concepts in more detail.

1.8 A Step-by-Step Methodology for Overlapping

Based on the framework presented above, I present a step-by-step methodology to overlap nominally-sequential activities.

Step 1: Map the "As-is process". Any of the tools, such as networks, flow charts, matrices etc. can be used.

Step 2: Reduce the aggregation of activities to see if any of the exchanged information is available earlier and, if available, see if it is useful to downstream activities. Aggregate those activities whose completion does not make any information available to later activities earlier.

Step 3: Model the interactions among activities as time constraints. Identify the critical path activities (time critical information exchanges).

Step 4: Classify each of time critical information exchanges based on their evolution and sensitivity (into slow/fast evolution and low/high sensitivity)

Step 5: Use the framework to determine the appropriate overlapping strategy

Step 6: Apply detailed models to determine how to overlap the activities.

Step 7: Update process and repeat steps 3 to 6.

This methodology will be illustrated in detail in Chapters 4 and 5. The reader will note that the first three steps are similar in spirit to the conventional project management based analysis. The next four steps differ from the convention in that instead of requiring that more resources be allocated to the critical path, the critical path activities are overlapped based on information needs.

1.9 Summary

In this chapter, I argued that the process viewpoint leads to an altogether different approach to managing product development compared to the conventional project viewpoint. A research agenda that emerges from the process viewpoint can be summarized as follows:

- Model design processes with emphasis on interactions – such that the constituent activities and information transfers are explicit.
- Develop a methodology to determine the key information exchanges that keep activities waiting, and/or are likely to cause iterations in activities.
- Identify ways to express the availability and impact of information exchanged
- Examine the relationship between patterns of execution and process performance parameters.

In the forthcoming chapters, I will develop these steps in detail. In Chapter 3, the focus is on the sequencing problem. Chapters 4 and 5 involve the detailed development of the overlapping problem. First, I position this work with respect to the rest of the literature in Chapter 2.

2. Previous Research in Product Development

...What we shall find is an exemplification, an encouragement, and a refinement of old wisdom.

– J. R. Oppenheimer in "Science and the Common Understanding"

In this chapter, existing research on design and product development are surveyed. (Work related specifically to the sequencing and overlapping problems is reviewed in Chapters 3 and 4.) The survey in this chapter is based on the taxonomy of existing research in product development presented in Figure 2.1. Before I delve into the individual branches in later sections, I present an overview of the taxonomy.

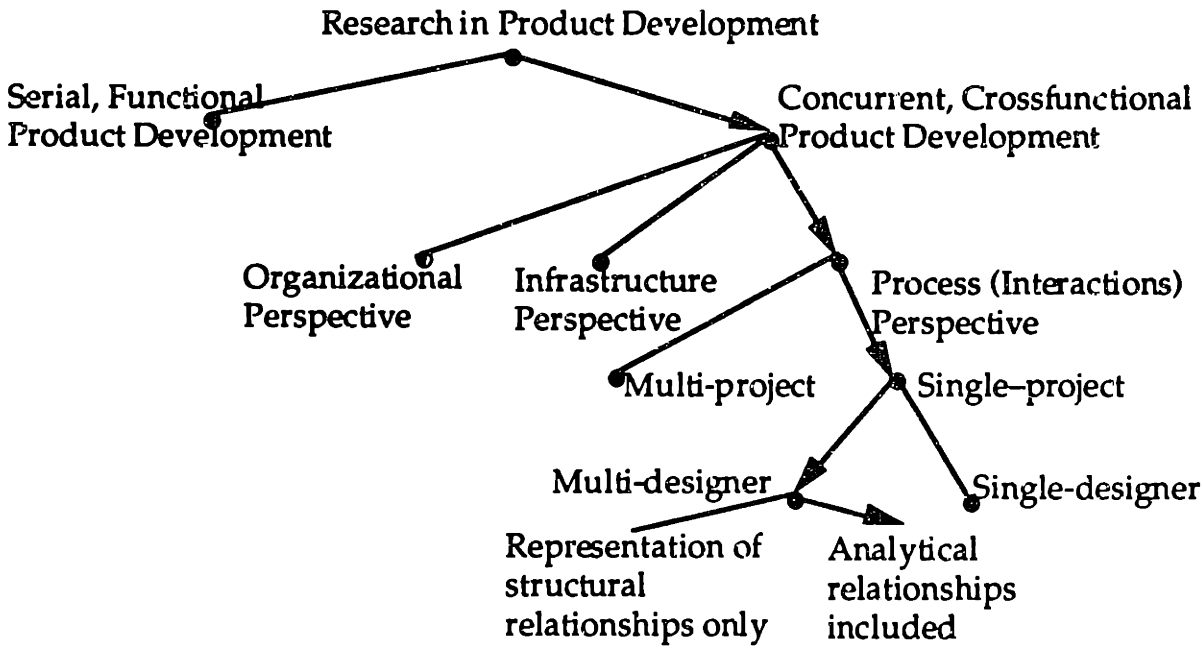


Figure 2.1: Taxonomy of Existing Research on Product Development
(The components dealt with in this thesis are indicated by arrows)

2.1 The Taxonomy

Based on its scope, research in product development can be classified into serial, functional research in which the focus is on individual functional disciplines (with subsequent serial assembly of the functional outputs), and concurrent, crossfunctional product development research in which the emphasis is on accomplishing the "integration" of the functional issues.

Research in crossfunctional product development could be further classified into (i) organization focused where the organizational factors that will be conducive to crossfunctional product development are considered, (ii) infrastructure focused where the goal is to build computer tools that will facilitate crossfunctional product development ("give-the-designers-the facilities-and-they-will-know-how-to-use-it"), and (iii) process interaction focused where the objective is to understand and manage the interactions among different functions and thereby improve development performance. The process perspective, which this thesis adopts, can further be classified into single-project, and multi-project research, based on whether interactions due to resource sharing among multiple projects are considered or not. Single-project research can be further classified into research that assumes a single designer (or a centralized source of design knowledge without information sharing), and multidesigner research which assumes design knowledge is distributed, and focuses on modeling the information exchanges among the multiple designers in the product development organizations. The arrows in Figure 2.1 indicates the branch which this thesis follows (single-project, multi-designer view).

Function	Viewpoint	Limitations
Marketing	The major challenge to effective product development is getting the fit between the product and the customer needs right.	Ignores the technical complexity of product development.
Styling	Competitive advantage is created by the product appearance, and so styling is the greatest challenge.	Ignores issues related to function and fabrication of the product.
Engineering	The technical complexity of the product (miniaturization, improving performance) makes engineering it a big challenge.	Underestimates the importance of making the product salable.

Figure 2.2: Exclusive Viewpoints of Different Functions

2.2 Serial and Concurrent Product Development

The process of development of a product is as much a business activity as it is a technical activity. As observed in Chapter 1, several different functions are involved including, Marketing, Styling (Industrial Design), Engineering Design, and Manufacturing. Traditionally, product development has been studied by researchers from each of these functional disciplines to the exclusion of others presumably with the aim of assembling the functional outputs in a serial fashion while creating the product. The emphasis of the various functional perspectives is summarized in Figure 2.2 along with some of the limitations. Lately, there have been topics which have merged two of the disciplines such as House of Quality (which has linked Marketing and Engineering concerns), and Design for Manufacturing.

"Concurrent Design", which advocates the more or less simultaneous design of the product and process, is a more recent approach [19]. I use the word "concurrent product development" to mean that all the product lifecycle issues such as customer needs, design details, manufacturing processes, and service factors are included in making decisions at every stage of the design process (not all lifecycle issues may be decided simultaneously due to technical constraints requiring that some decisions be made before others). The primary benefits of concurrent product development are, (i) a faster development process due to the simultaneous action, and (ii) better products because the decisions made by involving all the disciplines are more likely to be closer to the system-wide optimal solution.

The challenges facing concurrent product development are several-fold. First is the institutional challenge of implementing new practices in large-sized companies, and changing established habits and mindsets. Second, the existing infrastructure needs to be upgraded to facilitate concurrent product development. Third is the problem of integration in which the actions of multiple disciplines need to be coordinated by proper management of the interdependencies among the different development functions. Research approaches to these three problems are called the organizational perspective, infrastructural perspective, and the process perspective.

2.3 Organizational Perspective

Researchers subscribing to this view focus on the relationship between the patterns of organizational structure and behavior, and the effectiveness of product development.

Imai, Takeuchi and Nonaka [20] have argued that well-knit teams be formed that pursue product development like a rugby game (and not like a relay race). Although it is a powerful analogy, it implies that all product development activities can be carried out simultaneously, and ignores the interactions among development activities. In considering multiple lifecycle issues concurrently, one cannot pretend that interdependencies and precedence constraints among functions don't exist. The designers/design managers have to come to grips with the interactions by developing ways to relax, negotiate or even violate the coupling constraints among development activities.

Clark and Fujimoto [12], in their pioneering six year study of twenty automobile manufacturers in Japan, Europe and North America, examined the impact of strategy, organization and management on automotive product development. They used the "information perspective" in which product development is viewed as a process of transformation of data on market opportunities into a manufacturable design. As parameters of development performance, they quantified the engineering man hours, lead time, and product quality (a total product quality index constructed from various published quality data) of the different companies and observe significant differences among these companies. Indeed, their comparative study should serve as a benchmark and a powerful motivation for companies to *initiate* changes in their product development practice. However, their study does not consider the interdependencies among the development activities. I elaborate on this aspect using the following detailed observation.

One of the practices advocated in the above work to improve performance is integrated problem solving (which includes stage overlapping and intensive communication). Clark and Fujimoto argue that in order to improve integration, organizations should pay attention to five key dimensions:

- Timing of upstream and downstream activities (should be overlapped, not phased)
- Richness of information exchanged (should be face-to-face)
- Frequency of information transmission (should be more frequent)
- Direction of communication (should be bilateral)
- Timing of information release (should involve early release of preliminary information and not late release of complete information)

Although these dimensions may be useful to effect attitude changes in an organization, the generic reference made to information flow (without recognition of the differences among the different pieces of information exchanged) makes them insufficient to aid in the overlapping of adjacent phases. My own study of industrial product development shows that not all information exchanges are equal. Some product information exchanged can be more readily used in its preliminary form than others because of its lower impact on the downstream activities. Also, some product information can be frozen early, some need to be frozen early and some others cannot be or should not be frozen early. Overlapping of adjacent phases requires a careful understanding of the properties of the information exchanged apart from good, intensive and timely communication. I develop these properties in greater detail in Chapter 4.

Other researchers, notably McGrath et. al [5], Blackburn [21] and Reinertsen and Smith [22] have underscored the importance of development performance. McGrath et. al [5] offer a methodology, called PACE (Product and Cycle-time Excellence), to structure product development and improve performance. PACE contains seven interrelated elements: (i) Phase Review (ii) Core Teams (iii) Structured Development (iv) Product Strategy (v) Technology Management (vi) Design Techniques (such as DFM and QFD), and (vii) Automated tools. Their work is noteworthy for presenting several anecdotes in which an inefficient, and misdirected project was restructured. They also make a strong argument for studying product development as a process. However, most of this work is anecdotal; they do not treat the subject rigorously or give detailed examples on how to manage product development as a process.

2.4 Infrastructure Perspective

Research on infrastructure support for concurrent design emphasizes networks, databases and (blackboard) computer architecture. This work is motivated by the view that concurrent design is simultaneous search with many agents searching in multiple spaces at the same time [23, 24]). Using such a blackboard architecture, approaches have been proposed for dialogue between "intelligent agents" to resolve conflicts on the lifecycle issues. Alternatively, Kannapan et. al [25] have proposed a "concurrent engineering schema" in which the multiple lifecycle issues are modeled as intelligent design agents who negotiate to resolve conflicts on the design parameters using negotiation strategies derived from negotiation theory. Although these methods will be useful in design problems with a codified mathematical description, many problems in practice defy such a description and do not appear to be left to the control or decision making authority of "computer agents" in the near future.

2.5 Process Concepts in Product Development

The notion of process in electromechanical product development is relatively new [26]. Traditionally, product development has been studied as either a craft activity which has little or no resemblance to previous development efforts (in engineering departments) or as a strategic activity (in business schools). However, from their study of industrial development processes, McGrath et. al [5] note that about 72% of the work in a development project is repeated from the previous version of the project ("only 28% of the work is truly new"). 85% of GE's product development work is supposed to be purchase order reengineering or redesign [9]. Albano and Keska [10] did a "postmortem analysis" of three recently completed lightwave projects, reviewing available documentation, interviewing development personnel, documenting the intermediate steps and generating network diagrams. From their study, they conclude that while the product developed is unique in the different projects, "*the process of designing is a sequence of individual design steps that does not vary across the projects*". These and other studies serve as useful motivations for studying product development as a process. Apart from performance improvement that would result from such a viewpoint, it is also possible to leverage improvements in the methodology that arise in a

particular version of the process to other products, and to other generations of the same product.

Research on the (electromechanical) product development process can be classified into work focusing on single projects and that considering issues related to multiple projects. Multiproject related studies focus on processes that are not restricted to fully dedicated, platform teams. Hence, resource sharing is a major concern. Single project studies ignore the resource sharing effects and focus on modeling a single project in all its complexity.

2.6 Multi-Project Management

Adler et. al [27] apply the concepts of queueing theory to product development by modeling the development organization as a stochastic processing network in which engineering resources are "workstations" and projects are "jobs" that flow among workstations. They use these models to support decision making regarding resource allocation, and to predict project completion times. Watkins and Clark [28] develop a framework for resource allocation among multiple projects using three dimensions, (i) project sequencing (the number and content of projects, and the timing of different phases in different projects), (ii) resource dedication (the extent to which resources are dedicated) and (iii) resource specialization. Using this framework, they develop strategies for managing a portfolio of projects. Nobeoka and Cusumano [29] collect data from 223 new car projects to examine the influence of interproject linkage and development performance. *Lately however, firms have been decoupling the effect of multiple projects by adopting dedicated platform teams.*

2.7 Single-Project Description

Single-project research can be further classified into work considering multiple designers at work, and others focusing on the individual designer.

Single-designer work

This body of work assumes that the design knowledge is centralized (concentrated in a single designer) with no need for information sharing, and the goal of this work is to identify new tools and methodologies to alleviate the plight of the individual designer. It is based on the view that design is an intellectual activity performed by a single designer sitting with a note pad or

before a computer screen. This view is captured by the following definition from a renowned design theorist. Yoshikawa defines, design as a "typical intellectual activity which the human performs" [30]. Most of design theory is concerned with modeling the problem solving process of the individual designer. Tomiyama and Yoshikawa [31] have studied the design problem solving by individual designers in great detail and have described it as a sequential "evolution of metamodels" (without iterative refinement). Bell et. al [32] adopt a different tack by using dynamic systems as a metaphor to develop formalisms of problem solving by individual designers (which they call design process modeling). They offer a framework in which the "design process" is viewed at the structural, behavioral, and functional levels. (By structural description, they mean the connections between the methods and models used in the design process. By behavior they mean the interactions between models and methods. By function, they refer to the attainment of the design process goals measured in terms of quality, time etc.) Their work is noteworthy for its sophisticated conceptualization of design problem solving by the individual designer which they suggest will be useful in the development of future process tools and more effective usage of existing tools. Although it might help improve the productivity of the single designer, it still views design as individual problem solving and assumes that the interactions/information exchanges among multiple entities in product development should be easy to manage once the individual problem solving is well understood. In practice however, the interactions among multiple designers seem to be a major source of complexity.

2.8 Design as a Multi-Designer Activity

In this view, the design activity involves more than one individual; product architecture, physics and the organization structure require that the individuals involved in the design process interact. The emphasis is on modeling the interactions among the individual tasks, because it is argued that the complexity is not due to the individual tasks themselves, but in the way they interact each other [9]. The interactions manifest themselves in the exchange of product-related information among the individuals.

A strong motivation for this view (as well as this dissertation) is the study of industrial and academic approaches to electromechanical product

development in Japan and Europe by Whitney [11, 33]. Whitney concludes this study noting that "the main challenge industry people face is not in the quality of the individual designer's output", but in the organization and management of the design process – finding out its true structure, and information content. Most design research, however, focuses on improving the quality and productivity of the individual designer's output. He further observes that companies are trying to figure out ways to implement concurrent engineering (CE), and there are four stages of maturity in implementing CE:

Stage 1: A crossfunctional team is formed and the various conflicting product lifecycle issues are brought together. Confusion and disorder often results.
Stage 2: The team realizes its inability of the to solve the CE problem and feels the need for structuring the design process.
Stage 3: The entire team (along with facilitators) participates in mapping the design process, its decision sequence, and information flows. A detailed, collective, and comprehensive description of the design process emerges. Key design drivers that cause rework and delays are also identified.
Stage 4: The time-critical and content-critical design drivers identified in the previous step are addressed using computer tools or design methodologies.

Whitney notes that research community seems to be unaware of these stages; firms go through these stages with hardly any tools, and with no formal, scientific understanding of the steps/calculations needed to identify the key design drivers that are the sources of delays and iterations. I believe that the notion of evolution and sensitivity developed in this thesis will help make a beginning in formalizing the process of identification of the key design drivers. Evolution refers to the time availability of the upstream design information in its preliminary form, and sensitivity is equivalent to what Whitney calls content-criticality of information. Evolution data can help in reducing the wait time for the design information by transferring information or finalized form (by freezing design information in advance). In Chapter 6, I will discuss how the notion of evolution and sensitivity can help detect the key design drivers that a firm needs to address.

Whitney also outlines the steps that firms use in improving their design process. In one of the steps he says, "find precedence chains that can be broken so that the tasks can be resequenced (this requires classifying constraints,

much as Nippondenso does, into "must have", "would like"...)". The reader will note that this is similar to the role of the evolution-sensitivity framework in this thesis (relax precedence constraints by categorizing information based on its evolution and sensitivity).

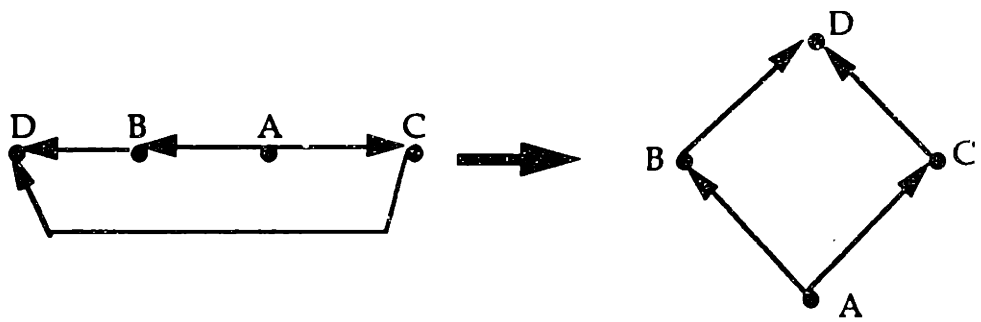
2.9 Analytical and Structural Descriptions

To identify improvements in a process, two characteristics of the process interactions need to be represented: (i) structural description which corresponds to the topological connectivity of the interactions, and (ii) analytical description which includes other properties of the interaction such as strength, frequency etc. These two aspects are considered in greater detail in the next section.

2.9.1 Structural Description of Interactions

The structural description of the interactions captures the *topological connectivity* (presence or absence) of the interactions: who is connected to who else (who depends on whom, who exchanges information with whom etc). Conventionally, the structure of interactions has been represented using graphs and binary matrices by researchers in systems engineering, including Warfield [34] and Steward [17]. Some examples are Interpretative Structural Modeling (ISM), and the Design Structure System (DSS). I discuss these tools briefly:

Interpretative Structural Modeling (ISM)



(a) Input **(b) Output**

Figure 2.3: A Directed Graph converted into a Hierarchical Form in ISM

ISM uses a directed graph representation of the connectivity of the interactions [35]. It manipulates or restructures the input directed graph in a sequence of steps; each step involves the identification of terminal nodes, and

placing them at the top level of a newly formed hierarchical representation. As an output, it produces a hierarchical representation of the input problem in which the higher levels of the hierarchy depend on the lower levels. For example, the directed graph representation showing the task couplings in Figure 2.3 (a) is transformed to the hierarchical structure shown in Figure 2.3 (b). The design parameters at a particular level are decoupled from other parameters at the same level. The design problem is solved bottom up (lower levels first followed by higher levels).

Design Structure System (DSS)

For large sized design problems, the DSS may be more convenient because it uses an adjacency matrix representation of the directed graph to map the structure of interactions [36]. As an example, the directed graph of Figure 2.3 (a) showing the task couplings is represented using the DSS in Figure 2.4. The coupling between tasks A and task B is indicated with a mark in the corresponding row and column of the matrix. It is seen that unlike ISM, the DSS uses a flat structure. The rows and columns of the matrix can be rearranged (or partitioned) such that the information generating tasks are at the top and information receiving tasks at the bottom[3]. The rearranged matrix is called the Design Structure Matrix (DSM). Note that the DSM{B, C} entry does not have a mark because tasks B and C are decoupled.

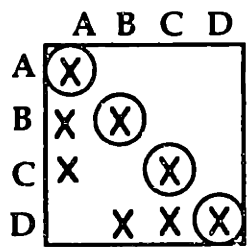


Figure 2.4: A Design Structure Matrix of the Graph in Figure 2.3 (a)

---

[3]In 1962, Steward developed the techniques of partitioning and tearing to solve a set of simultaneous equations [37, 38]. It is noteworthy that partitioning is equivalent to a graph theoretical technique called reverse topological sort. At the end of the 60's Steward advanced his matrix as a tool to represent interactions in a process. He termed the partitioned precedence matrix, Design Structure Matrix. Several schemes for partitioning are also reviewed by Gebala and Eppinger [16].

A process which can be partitioned to an entirely below-diagonal matrix (as in Figure 2.4) is said to be *completely sequential*. Partitioning of an adjacency matrix however, does not always lead to a DSM with only below diagonal elements. In Figure 2.5 for example, the activities A, C, D are mutually coupled perhaps due to the underlying product physics. These tasks are said to constitute a block. To solve such a block, iteration is required to ensure that the mutual couplings are satisfied. If the tasks involve making decisions about design variables, making a guess about one variable decouples or tears the problem. It would be desirable that the minimum number of guesses or "tears" are made. Steward has developed a procedure called shunt diagram (now available in a software program called TERABL) to determine the minimum number of tears [17].

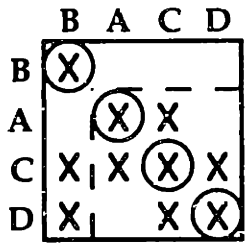


Figure 2.5: A DSM with Above Diagonal Entries

The DSM has been used quite extensively to study the patterns of information flow in product development processes [13, 15, 39-41][4]. In their work, Eppinger et. al [13] use the DSM as a tool to *document* the technical

---

[4]There are some subtle differences in the matrix representation proposed and used by Steward and the version used by Eppinger and colleagues. In Steward's representation, the marks in the matrix represent precedence relations among activities (Page 28 of Steward's book [17] states that the Design Structure Matrix results from doing partitioning and tearing of a precedence matrix), while Eppinger et. al [13] use the mark to denote information exchange among tasks. The most important difference among these versions arises from transitivity: while precedence is a transitive relation, information exchange is not. Thus in Steward's DSM, if task A precedes B and B precedes C, then A precedes C, and a transitive closure will contain a mark in the entry corresponding to DSM(A, C). In the representation used by Eppinger et. al, if B receives information from A, and C receives information from B, it is not necessary that C recieve information from A. Further Steward uses the DSM to primarily describe the interactions among parameters.

structure of a design project. They augment the DSM with numerical entries that may denote degree of dependence, task completion times etc. Further, they propose several different strategies to improve the design process including Parallelization, Artificial Decoupling, and Increased Coupling. Of these, the last deserves a special mention, because it relates to one of the approaches prescribed for design process improvement in this thesis, called overlapping, in which the process is improved by increasing the volume of information transferred.

Black et. al [15] mapped the interactions in an automotive brake system design problem as a Design Structure Matrix. It is noteworthy that a major portion of the brake system DSM is sequential although there exists a portion which is not (one hundred and three design parameters were identified out of which thirty four parameters form a block; see Figure 2.6). Such processes, which may not be completely sequential but are predominantly so, are said to be *nominally sequential.*

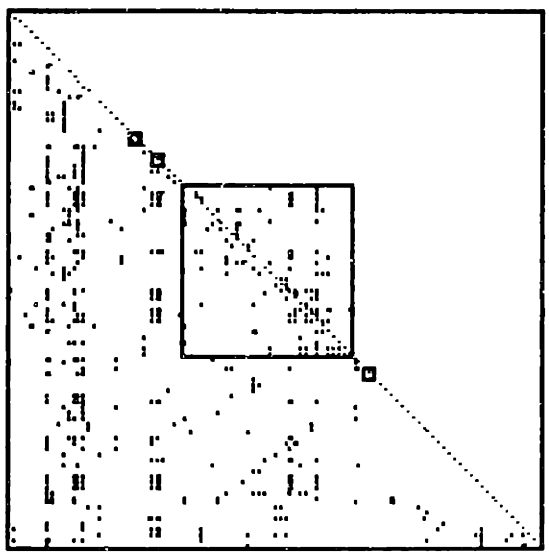


Figure 2.6: The Brake System Design Structure Matrix

Sequeira [40] used the DSM to suggest improvements in the automobile development process. She modeled the entire automobile development process (with greater detail on the upstream design phases) by interviewing the development personnel. Her work is noteworthy for pointing out the confusion over the "as is" and "to be" process in practice. She also classifies the dependency among tasks into self, primary, secondary and tertiary. "Self"

is just the diagonal element, but the other three types are related to this thesis, so I will consider them in greater detail.

In her work, the dependency between tasks A and B is termed:

- primary, if B cannot start until information from A arrives.
- secondary, if B requires information from A for completion
- tertiary, if information provided by A is helpful, but not crucial.

Although this classification is somewhat artificial, it implies that there is some information to which the downstream activity is more sensitive than others, and the downstream activities may start with preliminary values of some other information. These ideas, although not made in the context of overlapping activities, are useful in overlapping activities as we will observe in the following chapters.

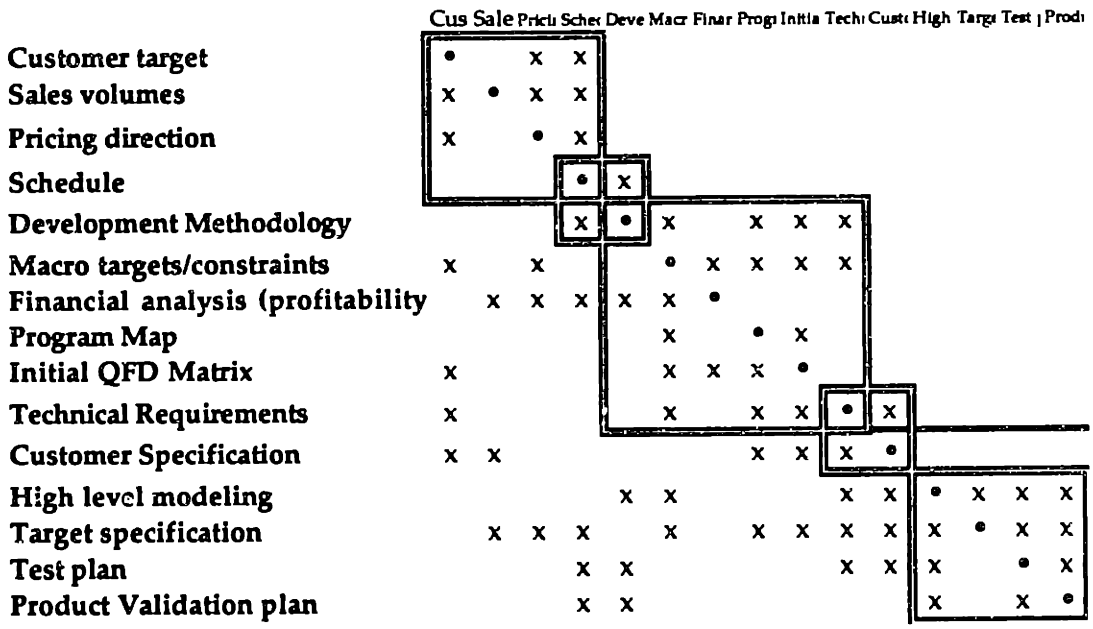


Figure 2.7: Portion of the Intel DSM from [14]

Osborne [14] used the DSS to map the product development process at Intel, and characterize cycle time. He has obtained some interesting results: for instance, he estimates from collected empirical data that iterations represent as much as two-thirds of a project effort . A portion of the binary DSM from one project studied by Osborne is shown in Figure 2.7. This DSM shows that this process (like the brake system) is also not completely sequential, but there are many blocks arranged in a sequential fashion. (In other words, this process is also nominally sequential.) This diagram is, however, more coupled than the brake system problem but the blocks are still

local. I will come back to this DSM in Chapter 6, where I will advocate that future work on overlapping should consider that the upstream and downstream design activities are blocks (individually), and focus on increasing the overlap between such blocks which are arranged sequentially. Such an effort may be one way of integrating previous work on iteration with this work on overlapping.

Design interactions have also attracted attention from several researchers associated with NASA. Rogers [42] has developed an expert system to partition and tear a DSM with the aim of analyzing and improving the design process. For solving large scale systems optimization problems, Sobieszczanski-Sobieski [43] uses a matrix representation similar to the DSM, (called the N^2 diagram). By reshuffling the rows and columns of the N^2 diagram, he attempts to decompose a large system into a hierarchic pyramid of modules.

2.9.2 Analytical Information about Interactions

The methods presented in the previous section use only the topological connectivity of the interactions. Several researchers [13, 44] have observed that analytical information about the interactions, such as the algebraic relationship among parameters, certainty of planning estimates, volume of information exchange, and the strength of dependency should also be used along with the connectivity information. In this thesis, I find that the sensitivity of downstream activities to upstream information, and the rate of evolution of the upstream information are two other useful properties in overlapping an existing process.

Rogan [44] argues that the directed graph based representations impose a precedence relationship in some problems where one may not already exist. He further shows that representations that exclude analytical information (contain only structural information) may not always lead to a feasible design. To obtain the analytical information however, he assumes that the design problem is formulated as a mathematical optimization problem. This compares with my approach in Chapter 3, but only some problems in practice may possess such a description, so it is necessary to obtain the analytical information about the interactions in other cases using models or estimates.

Smith and Eppinger [45, 46] replace the binary form of task dependency in a conventional DSM with numerical entries which are a quantitative measure of the strength of dependency between tasks. They provide two extensions to the DSM which helps estimate the time it takes to execute a coupled block – a probabilistic model (also called the sequential iteration model), where the number of iterations and the amount of rework per iteration is a random variable, and a deterministic model (also called the work transformation model) where the amount of rework caused by each iteration is a linear function of the amount of work done in the previous iteration. These models are explained in greater detail below.

The *sequential iteration* model [45] assumes that the tasks in a coupled block are done one at a time (sequentially), and there is a finite probability that coupled tasks will have to be iterated. Figure 2.8 shows a "Sequential Iteration DSM". The diagonal elements of this DSM indicate the length of time that the task would require if it were done in isolation, with all input information available. Each off-diagonal value element a_{ij} indicates the probability that another iteration of task i will be necessary given that task i was performed without the knowledge of the latest results from task j. The authors show in their work that the sequential iteration DSM describes a decision Markov chain each state of which corresponds to completing one task at one time, and the transition probabilities correspond to the probabilities of repeating a previous task, or of attempting a new task. They use this model to compute the expected time of each ordering, and to identify an initial ordering of the design tasks that minimizes the expected time it takes to execute the set of coupled design tasks.

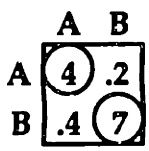


Figure 2.8: Sequential Iteration Design Structure Matrix (from [45])

(In Figure 2.8 above, task A takes 4 units of time and task B takes 7 units if done in isolation. Tasks A and B are coupled such that if A is done before B, then there is a probability of 0.2 that A will have to be repeated because the results of B are incompatible with the previous results of A. If B is initially

done before A, then there is a 0.4 probability that B will have to be repeated later.)

The sequential iteration model is restrictive because it makes several assumptions. The duration of the tasks is assumed to be the same from one iteration to another. It is also not clear how the off-diagonal elements, which are assumed to be constant and known, may be estimated in practice. Further, the fact that each of the tasks are done in a sequence in each iteration seems to represent an extreme case.

The work transformation model [46] relaxes the assumption that the duration of the tasks is the same from one iteration to another and assumes instead that the duration of the iterations decreases with time in a linear fashion. Further, the off-diagonal elements are not probabilities in this model but deterministic strengths of dependencies. All tasks are executed in each stage, and the duration of a task in the first iteration is given in the diagonal element of the work transformation matrix (WTM).

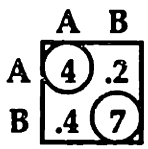


Figure 2.9: Work Transformation Matrix (from [41])

(In Figure 2.9 above, task A takes 4 units of time and task B takes 7 units in the first iteration. In subsequent iterations, the duration of task A is one-fifth the duration of task B in the previous iteration; similarly the duration of a subsequent iteration of task B is two-fifth the duration of task A in the previous iteration)

The work transformation model is interesting because, (i) it helps predict which portions of the design problem consume the largest amount of time, and (ii) the eigen structure of the work transformation matrix helps obtain the nature and rate of convergence of the design process. However, the linearity assumption – that the duration of a task in a particular iteration is a linear sum of the duration of each of the coupled tasks in the previous iteration – is central to this model (it makes all the results from linear algebra accessible), and it needs to be investigated if this assumption holds in real design processes.

Project Management

Project Management methods fall between structural and analytical representations in that, apart from representing the connectivity information, they also represent the timing constraints among activities. These methods represent the structural and timing information as a precedence network, and can further be classified into (i) Activity on Arrow (AOA) representation used by PERT and CPM in which arrows in the network are activities and nodes represent start and finish events, (ii) Activity on Node (AON) representation, followed by the Precedence Diagramming Method (PDM), in which arrows represent precedence constraints [47]. The latter is more flexible as it offers ways to model the interrelationships, such as start-to-start and finish-to-finish apart from the traditional finish-to-start constraints used by PERT and CPM. However, both approaches are activity-based and assume that durations of activities are predetermined constants. This assumption prevents them from allowing for dual and cyclic relations among activities which are common in product development. In Figure 2.10, I give an example of a dual relationship between two activities A and B: a start-to-start constraint, and a start-to-finish (overlap) constraint. If the activity durations are assumed constants, then the relationship in Figure 2.10 introduces a cycle in the network which will break down the network analysis technique (network cannot be ordered and further calculations cannot be performed). Such a relationship is physically feasible (for instance, one can think of A being design, and B being prototyping in which case one could conceive of a situation where prototyping cannot start until a few days after design starts, and design not finish until prototyping has been underway for a few days – to correct any infeasibilities). There is no reason, however, to require that the activity durations be predetermined constants. If this assumption is relaxed, the dual relationship can be allowed and the project management techniques can be extended.

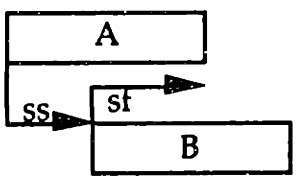


Figure 2.10: Dual Relationship in a Project Which Cause Cycles

In practice, project management techniques tend to be widely in use (as I observed in my field visits), and it would be worthwhile if these techniques can be augmented for usage in product development. So in Table 2.1, I list some of the limitations in project management that need to be addressed to make them useful for product development.

Project Management Model	How It Needs to be Modified
Information exchange is implicit	Exchange should be made explicit
Activity durations are constants and dual constraints as in Figure 2.10 cannot be represented.	Activity durations should be let to float. Activities must finish when process interactions are satisfied.
Activities are continuous (with no wait times), and happen only once.	Should be allowed to iterate, and to wait to process information.
Information is available for exchange only once and at the finish of the generating activity.	Should allow for information exchange more than once and at intermediate points of activities.
Only time-criticality is considered. Ignores effect of information content.	Should include the content of the information.
Process is improved only by crashing (expending resources faster).	Should include overlapping activities in the critical path.

Table 2.1: How Project Management Methods need to be Modified

2.10 Where This Thesis Fits

This thesis can be interpreted in several ways: (i) as seeking to solve some of the open questions in improving the design process posed by Whitney in [11, 33] (quoted in Sections 2.8 and 4.3), (ii) as addressing the issues not modeled by project management (see Table 2.1), (iii) as an extension of previous work on documenting the structure of the design process by Eppinger et. al reported in [13], and (iv) as complementing the previous work done by Smith on design iterations [41].

As indicated in Figure 2.1, this thesis limits itself to multi-designer, single project processes (which do not share resources with other projects). Within this domain, I focus on processes in which the activities are nominally sequential. Both structural representations (such as ISM, DSS etc.) and project management methods would have been helpful to organize the process into such a nominally sequential pattern of execution, as was exemplified in Figures 2.6 and 2.7. I am now interested in beneficially altering the sequence (Chapter 3) or determining operational ways to relax the constraints requiring nominally sequential execution (Chapters 4 and 5).

In Chapter 3, I consider design problems which can be described as a mathematical optimization problem (similar to the ones considered by Rogan [44] and Sobieszczanski-Sobieski [43]). These authors offer methods to decompose centralized design problems into smaller modules using structure and sensitivity. In Chapter 3, I consider the situation where knowledge about the design problem is already decentralized or distributed. I develop a procedure to order the solution of the distributed modules, without iteration, that leads as close as possible to the system-wide optimal solution.

In Chapter 4, the focus is on processes that do not necessarily possess a mathematical description, and in which the activities are coupled by information exchanges. I introduce two analytical properties of the information exchange, called evolution and sensitivity, that help relax the precedence constraints among the activities (modeled in project management) and thereby overlap the activities. Such overlapping may be accomplished by freezing upstream information ahead of time or by doing more downstream iterations. The combination of evolution and sensitivity values determine how to overlap activities and with what performance trade-offs – as explained in Chapters 4 and 5.

3. Sequencing Coupled Design Activities

Strategic product design is a total approach to doing business. It can mean changes in the pace of design, the identity of the participants, and the *sequence of decisions.* – Daniel Whitney in [6]

3.1 The Sequencing Problem

Technical and organizational considerations impose couplings among product development activities. If the couplings involve (directional) precedence constraints among the activities, they can be represented using any of the tools presented in Chapter 2 such as ISM or the DSS. Further, the process can be organized by restructuring the directed graph or partitioning the precedence matrix.

However, there are some design problems in which the couplings may not be directional. In Chapter 1, I described the dc motor design decision problem in which the motor parameters (such as diameter) were to be decided while considering product lifecycle characteristics (function, space and material cost). The problem exemplified the situation where all the design tasks are coupled, but not related by any precedence constraints. In such problems, there are multiple ways in which the design tasks can be ordered or sequenced (tasks are done once without any iteration), and it is not possible to distinguish among these different orders using information about the structure of interactions alone. The primary difference among the different orders is in the loss in quality of the design solution for the subsequent decision makers due to the decisions made by their predecessors. In this chapter, a methodology will be presented to identify the optimal ordering of tasks (with the lowest loss of quality for the set of all lifecycle issues) in design problems where each design task can be posed as a nonlinear program. I will use the dc motor example to illustrate the various concepts introduced in the optimal ordering identification.

It would be difficult to understand the characteristics of cross-functional decision making in the absence of any specific interpretations for design tasks. To focus on the problem, an individual team member's task is interpreted as the optimization of a particular design criterion. This provides a unified basis

to model the behavior of the multiple decision makers in cross-functional product design. Although in routine life, people may be satisficers rather than optimizers [48], in many commercially competitive activities product developers are required to obtain optimal results. In the design of complex and novel technologies such as hypersonic aircraft, using the optimal results may make the difference between "flying and staying on the ground" [43]. In this chapter, I further assume every functional design task has the form of a nonlinear program, as is the case with some parametric design activities. First, I will discuss the relationship of this chapter to other work in literature.

3.2 Ordering and Decomposition

It is important to compare and contrast this work from other work on decomposition and design process planning [43, 44]. The similarity is that both of these works focus on parametric design problems which involve the decision about product variables for a *given* concept. In decomposition and design process planning however, the researchers seek to decompose *centralized* knowledge about a design problem into loosely coupled subproblems. Because the problem is not already decomposed, there is the flexibility to decide what the objectives, decision variables, and constraints of each of the modules should be in order to get as close as possible to the multiobjective optimal solution.

My work in this chapter, on the other hand, considers problems where the knowledge about the designed artifact is decentralized – distributed among multiple designers or design groups (as in crossfunctional teams). The objectives and constraints of each of the modules is predetermined by the functional disciplines. If the decisions made by the individual groups were to be assembled in a serial fashion, as in a conventional sequential process, without considering the interactions among the decisions, then the product is likely to be sub-optimal because upstream decisions are likely to affect the downstream activities adversely. If the decisions were made simultaneously, on the other hand, they are likely to be in conflicts, and several iterations may be needed to arrive at a solution that satisfies all considerations. So I consider the interactions among tasks and determine an ordering of the tasks that

results in the least quality penalty for the sum total of all tasks[5]. Because no task is done more than once, sequencing is a faster and less complex way to ensure that the couplings are satisfied than iterating or negotiating - although less likely to be globally optimal because of the lack of subsequent iterations.

Further, a reader familiar with decomposition literature will note that many more combinations of parameters can be passed between modules in decomposition than in ordering problem (where the decomposition is treated as a given item); the added flexibility in decomposition complicates the computation of the quality penalty for parameter passing. In this chapter the quality penalty is calculated simply by taking the difference between the sequential solution and the solution without sequence constraints – as described in the next section after introducing some terminology.

3.3 Terminology

For this analysis, I use the following definitions.

• P is a cross-functional product development process with n decision makers, entrusted with executing n cross-functional decision making tasks $T_1, T_2, T_3, \ldots, T_n$ and thereby choosing the values of m parametric design variables $x_1, x_2, x_3, \ldots, x_m$. Let X be a set comprised of all the design variables $x_1, x_2, x_3, \ldots, x_m$.

• Each task or function T_i, upon execution produces a functional output J_i. Let Z_i be the set (and z_i a vector) consisting of all the design variables that could possibly be decided by T_i. ($Z_i \subset X$). The variables belonging to Z_i, $\{z_{i1}, z_{i2}, \cdots, z_{ir}\}$, are said to occur in task T_i.

• The decision process in which the task T_i is not subject to any order constraints is called the independent decision process. Let J_i^* represent the functional output of task T_i in the independent decision process; the value set by T_i for the design variables in the independent decision process will be called independent decisions.

---

[5]Because the term "sequence" brings back bad memories of a conventional sequential process (which does not consider interactions) to some readers, I will often use the term "ordering" to remind the reader that interactions and multiple orders are being considered in this work.

- A *decision order* φ is a sequence of the n tasks, $T_1, T_2, T_3, \ldots, T_n$, such that:

1) The tasks in the design process are executed in the order in which they occur in φ.

2) Each task T_i upon execution decides on a value for all undecided variables belonging to Z_i. The output of task T_i in the order φ will be denoted by J_i^{φ}.

Let $\overline{Z_i}$ denote the complement of Z_i. Consider the order $\varphi = \{T_1, T_2, T_3, \ldots, T_n\}$. In this order, T_i decides the values of all variables in $Z_i \cap \overline{\{Z_1 \cup Z_2 \cup \cdots \cup Z_{i-1}\}}$. It faces order constraints of the form, $z_{iq} - z_{iq}^{\varphi} = 0$ for $z_{iq} \in Z_i \cap \{Z_1 \cup \cdots \cup Z_{i-1}\}$. In other words, T_i loses freedom in $Z_i \cap \{Z_1 \cup \cdots \cup Z_{i-1}\}$.

- A design variable x_k's value in the ordered design process is *decided* by the first task in the order in which it occurs.

- A task T_i is said to lose the design freedom x_k in the order φ, if $x_k \in Z_i$ and the value of x_k in the order φ is decided by a preceding task.

Defn: Quality Loss incurred by task T_i in a decision order φ, QL_i^{φ}, is defined to be the nonnegative offset of the output of task T_i in φ, J_i^{φ}, from J_i^*:

$$QL_i^{\varphi} = \left| J_i^{\varphi} - J_i^* \right|$$

Defn: Quality Loss of an order φ, QL^{φ}, is defined to be the weighted sum of quality losses incurred by each task T_i in φ.

$$QL^{\varphi} = \sum_{i=1}^{n} w_i \, QL_i^{\varphi}$$

Defn: A decision order φ^* is defined to be the *optimal decision order* if its quality loss $QL^{\varphi^*} \leq QL^{\varphi}$ for all possible φ.

In the next few subsections I relate the optimal order (for execution of the cross-functional decision tasks) to the underlying structure of the design interactions, illustrated through the example of the design of a dc (direct current) motor introduced in Chapter 1, and repeated here for convenience.

3.4 DC Motor Design Problem

To decide on the order in which decisions should be made by a cross-functional team, whose members are concerned respectively with maximizing the torque generated by the dc motor (performance), minimizing the area occupied by the stator (size) and minimizing the cost of materials (sum of the area occupied by the steel portion of the rotor and area of copper). The variables to be decided are given in Table 3.1 and the tasks are given in Table 3.2. The independent decisions and outputs are summarized in Table 3.3.

Decision Variable	Symbol	Bounds
Armature diameter	ad	$10 \le ad \le 12$ (inches)
Motor inner diameter	id	$0.1 \le id \le 3.0$ (inches)
Motor outer diameter	od	$20 \le od \le 24$ (inches)
Diameter of windings	dw	$0.01 \le dw \le 0.2$ (inches)
Current density	cd	$0.1 \le cd \le 50.0$ (amp / in^2)
No. of armature windings	nw	$1 \le nw \le 1500$ (turns)
Thickness of magnet used	tm	$0.05 \le tm \le 1.0$ (inches)

Table 3.1. Design Variables for a dc motor

Task	Task Description	Analytical Forms (Minimizations)
T_1	Maximize Torque	$J_1 = -1.57\ cd\ dw^2$
T_2	Minimize Space	$J_2 = 0.785\ (od^2 - ad^2) - 0.26\ nw\ dw^2$
T_3	Minimize Material Costs	$J_3 = 0.785\ (ad^2 - id^2) - 2.1\ ad\ tm + 0.785\ dw^2$

Table 3.2 Objectives as Functions of Variables

Task	Independent Decisions and Outputs
T_1	$dw_1^* = 0.2;\ cd_1^* = 50;\ J_1^* = -3.14$
T_2	$ad_2^* = 12;\ od_2^* = 20;\ dw_2^* = 0.2;\ nw_2^* = 1500;\ J_2^* = 185.3$
T_3	$ad_3^* = 10;\ id_3^* = 3;\ dw_3^* = 0.01;\ tm_3^* = 1.0;\ J_3^* = 50.4$

Table 3.3 Independent Decisions and Outputs

It is seen (as was observed in Chapter 1) that the designers executing the different tasks independently drive the variables to different values; ordering the tasks is one way of ensuring that the design variables assume a unique value (because subsequent decision makers lose their degrees of freedom and are constrained by the value of the variables set by predecessors). An example of a decision order was given in Chapter 1.

The straightforward method to determine the optimal order, by explicitly considering all orders and evaluating their quality losses, is shown in Table 3. 4. Evaluating the quality loss of each decision order requires the execution of every task because their results depend on their position in the decision order. So in this case a total of 3! x 3 = 18 nonlinear programs need to be solved to identify that $\{T_1, T_3, T_2\}$ is the optimal decision order with the lowest quality loss. In large designs, determining the optimal order becomes tedious requiring *Order*($n!$ n) optimizations. Evidently, even if the tasks may not be nonlinear programs, exhaustive enumeration is expensive.

Order	Design Decisions made during the order	QL
$\{T_1, T_2, T_3\}$	*ad*=12; id=3; *od* = 20; *dw*=0.2; *cd*= 50; *nw*= 1500; *tm* =1.0	0.603
$\{T_1, T_3, T_2\}$	*ad*=10; id=3; *od* = 20; *dw*=0.2; *cd*=50; *nw*= 1500; *tm* =1.0	0.189
$\{T_2, T_1, T_3\}$	*ad*=12; id=3; *od* = 20; *dw*=0.2; *cd*=50; *nw*= 1500; *tm* =1.0	0.603
$\{T_2, T_3, T_1\}$	*ad*=12; id=3; *od* = 20; *dw*=0.2; *cd*=50; *nw*= 1500; *tm* =1.0	0.603
$\{T_3, T_1, T_2\}$	*ad*=10; id=3; *od* = 20; *dw*=0.01; *cd*=50; *nw*= 1500; *tm* =1.0	1.268
$\{T_3, T_2, T_1\}$	*ad*=10; id=3; *od* = 20;*dw*=0.01; *cd*=50; *nw*= 1500; *tm* =1.0	1.268

Table 3.4: Results of All Decision Orders

Inspection of Table 3.4 shows that the values of several design decisions (for example, *cd*) are invariant over all the orders while some design variable values are invariant over a subset of all orders (for example the value of *ad* is the same in three of the orders). These invariances can be systematically utilized to relate the quality loss of a decision order to the structure and strength of the design interactions as shown below.

Quality loss of the individual tasks varies with the decision order because different degrees of freedom are lost in different decision orders. However there exists a subset of decision orders over which a particular task loses a particular design freedom. For instance, in four of the six orders T_3 loses the design freedom, *dw*. In all these orders the design variable assumes the same value, $dw = 0.2$, and T_3 faces the same order constraint, perhaps resulting in the same quality loss. How could we exploit invariant design variable values which translate into invariant order constraints? If we can decompose the quality loss incurred by a task into components due to loss of individual design degrees of freedom, then we can compute the order variant quality loss from order invariant terms. Then the following questions arise: What is the general structure in such problems that leads to order invariance? Can the quality loss of every order be decomposed into a certain small number of order invariant quantities, calculated a priori?

3.5 Exclusive Groups

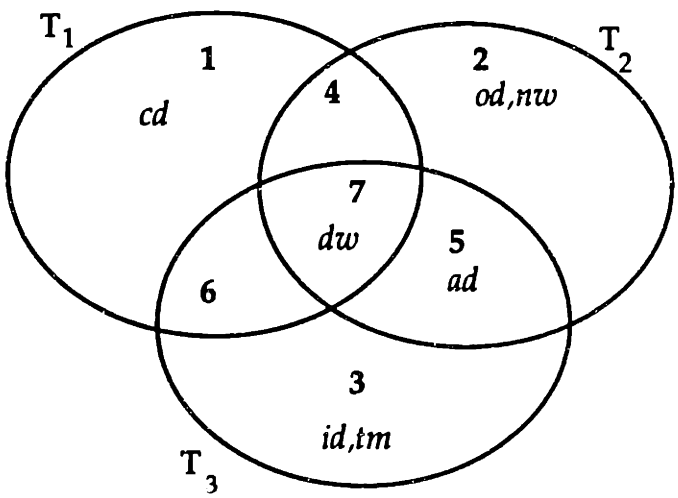


Figure 3.1: Exclusive Groups in the DC Motor example

It is interesting to observe that in ordering cross-functional decision making, if two design variables x and y occur in the same combination of design tasks, then the decision about the values of both x and y will *always* be made by the same team person. Any subsequent member who loses the design freedom x will also lose design freedom y to the same member that decided x. Whenever we compute the quality loss due to loss of design degree of freedom (d. o. f.) x, we will also compute the quality loss due to loss of (d. o.

f.) y. So all variables that occur in the same combination of tasks can be grouped together as shown in Figure 3.1. The quality loss incurred by a task needs only to be decomposed into components over losses of such groups of design freedom. Any cross-functional team member possesses or loses design freedom in groups. In the three task dc motor problem, design variables get partitioned into seven groups. Because the combinations are exclusive to each other, the groups are called exclusive groups.

An exclusive group consists of variables that appear in the same combination of tasks. For a n task design process there are $M = 2^n-1$ groups, which correspond to the boolean combination of the complements given below[6].

$$\begin{Bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_i \\ \vdots \\ Y_M \end{Bmatrix} = \begin{Bmatrix} (Z_1 \cap \overline{Z_2} \cap \overline{Z_3} \cap \cdots\cdots \cap \overline{Z_n}) \\ (\overline{Z_1} \cap Z_2 \cap \overline{Z_3} \cap \cdots\cdots \cap \overline{Z_n}) \\ \vdots \\ (Z_1 \cap Z_2 \cap \overline{Z_3} \cap \cdots\cdots \cap \overline{Z_n}) \\ \vdots \\ (Z_1 \cap Z_2 \cap Z_3 \cap \cdots\cdots \cap Z_n) \end{Bmatrix}$$

The tasks in which the variables belonging to any exclusive group Y_i occur will be referred to as tasks *forming* Y_i. For example, the tasks forming Y_7 are T_1, T_2 and T_3. The tasks themselves are said to be *spanned* by the various exclusive groups.

In Figure 3.1, T_3 is spanned by Y_3, Y_5, Y_6 and Y_7. Let y_j be the vector of variables belonging to Y_j, and t_j be the set of tasks forming Y_j. For the dc motor problem, we have:

$$z_1 = \begin{Bmatrix} cd \\ dw \end{Bmatrix}; \quad z_2 = \begin{Bmatrix} ad \\ od \\ dw \\ nw \end{Bmatrix}; \quad z_3 = \begin{Bmatrix} ad \\ id \\ dw \\ tm \end{Bmatrix}$$

---

[6]In a typical product design problem, n is about 5, while m is about 1000; n << m, making the grouping of variables useful.

$$y_1 = \{cd\};\ y_2 = \begin{Bmatrix} ad \\ nw \end{Bmatrix};\ y_3 = \begin{Bmatrix} id \\ tm \end{Bmatrix};\ y_4 = \{\varnothing\};\ y_5 = \{ad\};\ y_6 = \{\varnothing\};\ y_7 = \{dw\}$$

$$t_1 = \{T_1\};\ t_2 = \{T_2\};\ t_3 = \{T_3\};\ t_4 = \begin{Bmatrix} T_1 \\ T_2 \end{Bmatrix};\ t_5 = \begin{Bmatrix} T_2 \\ T_3 \end{Bmatrix};\ t_6 = \begin{Bmatrix} T_1 \\ T_3 \end{Bmatrix};\ t_7 = \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \end{Bmatrix}$$

The decision made by a task T_j for the values of the variables in a particular exclusive group Y_i is referred to as the exclusive group Y_i decision by task T_j. The loss of design freedom in deciding the values of all variables belonging to an exclusive group is referred to as loss of exclusive group freedom. Notice that in the dc motor case, the values of all the design variables in certain exclusive groups, such as Y_3, do not vary from one order to another (Table 3.4). Such groups are called *order-invariant exclusive groups*.

Defn: An exclusive group Y_j is said to be order invariant if the value of each design variable belonging to Y_j is the same under every order.

Why are some exclusive groups not order invariant? There are two reasons why a certain exclusive group decision may vary from one order to another. *First*, different designers may decide the values of design variables in an exclusive group in different orders. For example, in the order $\{T_3, T_1, T_2\}$, the team member responsible for the first task (T_3) makes the exclusive group Y_7 decision and sets the value of variable *dw* to 0.01, while in the order $\{T_1, T_3, T_2\}$, the designer responsible for T_1 makes the exclusive group Y_7 decision (sets *dw* to 0.2). *Secondly*, even when the same task makes an exclusive group decision, the value decided may vary from one order to another, because the task may face different constraints in different orders (because of different predecessors). If however, the order constraints do not affect the exclusive group Y_i decision made by T_j, then the exclusive group Y_i decision by task T_j is the same as its independent decision. What this means is that the exclusive group Y_i is invariant over a subset of all orders, the subset in which it is decided by the forming task T_j.

Defn: An exclusive group Y_j is said to be *forming task T_i invariant* if the value decided by T_i for the design variables belonging to Y_j is the same in every order in which T_i makes Y_j decision.

Notice that if an exclusive group is order invariant, its value does not vary under any order, and so it is also forming task invariant. However, an exclusive group that is forming task invariant is not necessarily order invariant. The motive behind pursuing these two invariances is to use them to facilitate the identification of the optimal order. For instance, if all the exclusive groups spanning a certain design task are forming task invariant, then the designer entrusted with the task need not repeat the decision making process (after having identified the independent decisions). Identifying order invariance impacts (by reducing the size of) every task in the design process. The question arises as to how one may identify these two invariances.There are several situations under which exclusive groups may be order invariant or forming task invariant. Towards characterizing these properties, we focus in our research on identifying increasingly stronger sufficient conditions. As a first step, I state the following propositions (proved in appendix A1) for design problems where all design objectives are continuously differentiable and explicitly expressible as functions of design variables (through serial constraint sets) and all inequality constraints are range constraints.

Proposition 1 An exclusive group Y_j is order invariant if C1 (C1.1 or C1.2) is satisfied.

C1.1 $\dfrac{\partial}{\partial y_k}\dfrac{\partial J_p}{\partial y_j} = 0 \;\forall\; y_k, (j \neq k)$ spanning every $T_p \in t_j$.

(i. e. the exclusive group is insensitive to other exclusive groups spanning every forming task)

and the independent decisions of all forming tasks equal the same value y_j^*.

C1.2 In the range of the design problem in question, each design variable in Y_j is monotonic in every forming task (partial of the forming task with respect to the design variable is sign invariant in the range of the design problem); further, the monotonicity of a variable is of the same type (increasing or decreasing) in all tasks:

$$\forall\; X_i \in Y_k, \text{ either } \frac{\partial J_p}{\partial X_i} > 0 \;\forall T_p \in t_j \text{ or } \frac{\partial J_p}{\partial X_i} < 0 \;\forall T_p \in t_j$$

Proposition 2 Exclusive group Y_j is forming task T_i invariant if it satisfies C2.1 or C2.2

C2.1 $\frac{\partial}{\partial y_k}\frac{\partial J_i}{\partial y_j} = 0 \; \forall \; y_k, (j \neq k)$ spanning T_i and not satisfying C1.

(i. e. y_j decision by T_i is sensitive only to other provenly order invariant exclusive groups)

C2.2 $\forall \; X_i \in Y_k$ either $\frac{\partial J_p}{\partial X_i} > 0$ or $\frac{\partial J_p}{\partial X_i} < 0$ (i. e. every Variable in y_j is monotonic in T_i in the range of the given design problem).

$$\frac{\partial J_3}{\partial y_3} = \begin{Bmatrix} -2.1\, ad \\ -1.57\, id \end{Bmatrix} < 0;$$

$$\frac{\partial J_3}{\partial y_7} = \{1.57\, dw\} > 0;$$

$$\frac{\partial J_3}{\partial y_5} = \{1.57\, ad - 2.1\, tm\} = \frac{\partial J_3}{\partial y_5}(y_3, y_5)$$

Table 3.5: Satisfaction of C2 by all groups spanning T_3

Table 3.5 lists the variables in the spanning exclusive groups of T_3 and the partial of J_3 with respect to the exclusive group freedoms. Due to the positivity of design variables we can calculate the ranges of partial derivatives in the given domain and we find that T_3 is monotonic with respect to the variables in Y_3 and Y_7 and these variables are always driven to their limits by T_3. Hence, the exclusive group Y_3 and Y_7 decisions satisfy C2.2. We can also show that Y_3 is an order invariant exclusive group. The variables in Y_3 occur only in T_3 and so are driven to the same limit in every order because only T_3 can decide them. (Y_3 satisfies condition C1.2). Y_7 is not order invariant because the variable in Y_7 occurs in all tasks and with different monotonicities; in different orders it can be driven to different values by different tasks. We can also see that the exclusive group Y_5 decision is sensitive only to the order invariant exclusive group Y_3 (apart from itself) and thus Y_5 is also forming

task T_3 invariant (C2.1). All exclusive groups spanning T_3 are forming task T_3 invariant and thus there is no need to reexecute T_3 because, its decisions under any order will be the same as its independent decisions. Such tasks, all whose spanning exclusive groups are forming task invariant, will be referred to as *tasks with invariant spanning groups.*

Notice that identifying invariant exclusive groups using C1 or C2 is simple because it just requires mapping design variables into exclusive groups and checking the sign invariance of the range of first partial derivatives (using tools such as interval analysis) or evaluating the mixed partials with respect to the exclusive groups. It is also interesting to observe that if a particular exclusive group is sparse (or the number of forming tasks is minimal) then it is likely to be order invariant, because the number of deciding tasks is less. Such is the case with the order invariant dc motor exclusive group Y_3. If the exclusive group is monotonic in a task, then it is (forming) task invariant. If an exclusive group is both monotonic and sparse then it is a strong candidate for order invariance. Thus the topological notion of sparseness and the analytical sensitivity based idea of monotonicity come together in a synergistic fashion to enhance the two invariances. These ideas are useful in ordering decision making in cross-functional teams in the following incremental fashion:

- If a certain exclusive group is order invariant, then the size (and thereby the complexity) of all tasks are reduced. If it is forming task invariant, then the size of the particular forming task is reduced.

- All tasks with invariant spanning groups need not be required to make decisions after having been asked to make independent decisions.

- If every task is spanned by invariant groups, as in the case of dc motor, then the quality loss calculation and optimal order identification can be simplified as shown below.

Let us consider the invariance of spanning exclusive groups in more detail.

3.6 Composing Partial Quality Losses

The next step in the reasoning involves attributing the quality loss incurred by a task to losses of specific exclusive group freedoms. We introduce an intermediate construct called Partial Quality Loss. *Partial Quality Loss* (PQL) incurred by a task due to loss of a particular exclusive group freedom is the degradation in the task results due to the loss of only that particular exclusive group freedom. For instance, the partial quality loss incurred by T_3 due to loss of exclusive group Y_7 freedom is the loss in quality of the output of T_3 when design freedom in only the variables belonging to Y_7 is lost. I show in appendix A1.4 that, if all exclusive groups spanning a task satisfy C1 or C2.1, as is the case with T_3, then the total quality loss incurred by the task is the sum of partial quality losses due to loss of each individual exclusive group freedom[7]. This enables the decomposition of the quality loss of an order into partial quality losses over exclusive groups. However, the value of this partial quality loss depends on which task to which T_3 lost its exclusive group Y_7 freedom. Y_7 freedom could be lost to either T_1 or T_2. This implies that we have to calculate both terms, partial quality loss for loss of Y_7 freedom to T_1 and to T_2, and use the quantity that is appropriate to the order. In other words partial quality loss *incurred* is a function of three entities: 1) the deciding task 2) the freedom losing task and 3) the exclusive group under consideration.

Quality Loss Infliction and Stage Independence

Consider the case where instead of T_3 losing its exclusive group Y_7 freedom to other tasks, other tasks lost the exclusive group Y_7 freedom to T_3. In such cases T_3 "inflicts" a quality loss on other tasks by constraining them with its decision of Y_7. The partial quality loss inflicted by a task such as T_3 is defined to be the weighted sum of the partial quality losses incurred by all other tasks due to loss of exclusive group freedoms to T_3. (When all exclusive groups spanning a task satisfy C1 or C2.1, the partial quality loss incurred due to loss of a particular exclusive group freedom can be readily

---

[7] An interesting mnemonic for this result is the Dalton's law of Partial Pressure: Just as the total pressure in a mixture of is the sum of their partial pressures, the total quality loss is the sum of partial quality losses.

calculated with just the independent decisions, as shown in appendix A1.4.) Because, the tasks forming an exclusive group are known from the topological structure, *the partial quality loss inflicted is only a function of the deciding task and the exclusive group!* How can we use this to our advantage? In designs where every task is spanned by exclusive groups satisfying C1 or C2.1, the quality loss of an order, which equals the sum of partial losses incurred, is shown in Appendix A1 to be equal to the sum of the quality losses inflicted by each task in the order. We can represent these quality losses as a network, as illustrated in Figure 3.2 for the dc motor.

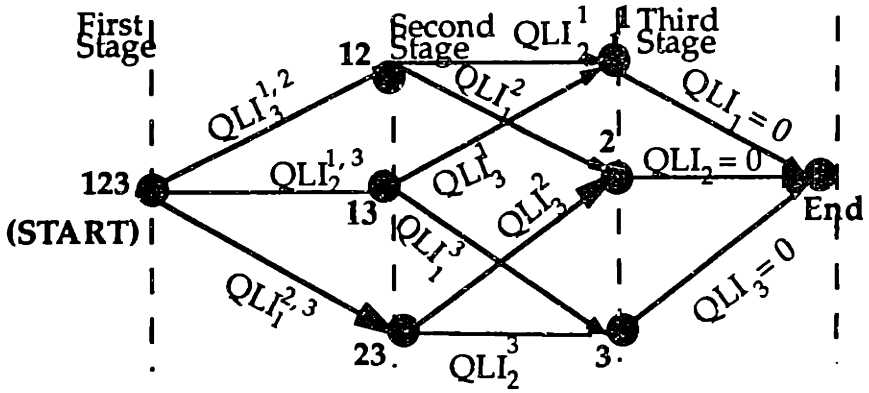


Figure 3.2: Network representing the Cross-Functional Decision Orders

The nodes of the network (in bold letters) represent the tasks remaining to be executed at a particular stage of the cross-functional decision process. For example the start node 123 indicates that tasks T_1, T_2 and T_3 are left to be executed and the node 23 indicates that tasks T_2 and T_3 are left to be executed. End node signifies that no more tasks are left to be executed. The edge which connects node 123 and node 23 corresponds to the execution of task T_1 in the first stage and the weight of this arc, $QLI_1^{2,3}$, is the quality loss inflicted by T_1 on T_2 and T_3 when it is the first task to make decisions. Notice that one task is executed at every stage and each directed path from the start node to the end node corresponds to one specific order and the length of the path equals the sum of quality losses inflicted by each of the tasks. All orders are represented in the network, the number of paths from start to end equals the number of orders. If every task is spanned by exclusive groups satisfying C1 or C2.1, the quality loss of the order is equal to the sum of quality losses inflicted, which is equal to the length of the path from the start to the end node. Hence the

optimal order (with the smallest quality loss) corresponds to the shortest path of the network given in Figure 3.2.

3.7 Identifying the optimal order

Following is the procedure used to identify the optimal order.

1. Each design task is executed separately for its "independent" output and decisions.

2. The design variables are partitioned into $M = 2^n - 1$ exclusive groups.

3. It is verified if a specific exclusive group is forming task invariant or order invariant. Any task all whose spanning exclusive groups are invariant need not be executed again, after having executed it once in Step 1. Its decisions at a stage of the order are the same as its independent decisions. Also the quality loss of such a task is a simple algebraic summation of the partial quality losses.

	T_1	T_2	T_3
Y_1	0	x	x
Y_2	x	0	x
Y_3	x	x	0
Y_4	0	0	x
Y_5	x	0.602	0.188
Y_6	0	x	0
Y_7	0.001	0.001	1.081

Table 3.6: Quality Loss (Inflicted) Matrix

4. When every task is spanned by exclusive groups satisfying C1 or C2.1,the quality losses inflicted by each task (by deciding the variables in each spanning exclusive group) are stored in a matrix (Table 3.6), called the Quality Loss Matrix (QLM). QLM(i,j) denotes the quality loss inflicted by T_j by deciding the variables in Y_i. Notice that a task cannot decide the values of design variables in exclusive groups which it does not form. For example T_1 cannot decide the value of variables in Y_2 and therefore cannot inflict a quality loss. QLM(i,j) is set to x if T_j cannot inflict a quality loss in Y_i. Notice that if a particular

exclusive group is order invariant, then the corresponding row in the QLM will be zero (except for the x's denoting the non-forming tasks). This is because the exclusive group does not contribute to any quality loss. Such an exclusive group may be removed from the design problem, reducing the size and complexity. In Table 3.6, this is the case with exclusive groups Y_1, Y_2, Y_3, Y_4 *and* Y_6. (It is noteworthy that Y_1, Y_2 and Y_3, are also the sparse exclusive groups, formed by a single design task).

5. The next step involves determination of the optimal order using the quality loss terms stored in QLM. We construct a network of quality losses, similar to Figure 3.2, and using the terms in QLM(see Figure 3.3). The arc connecting node 123 and node 23 corresponds to the execution of task T_1 in the first stage and the weight of this arc is the quality loss inflicted by T_1 when it is the first task to make decisions (= QLM(7,1) + QLM(4,1) + QLM(1,1)). Similarly, the arc connecting nodes 23 and node 3 corresponds to the execution of task T_2 in the second stage and its weight is the quality loss inflicted by T_2 when it is the second task to make decisions (= QLM(5, 2)). The weights of the other arcs can be filled in the same fashion. It is noteworthy that weights of several arcs (italicized) equal zero due to order invariant exclusive groups. Since the optimal order is the one with the lowest sum of the quality loss inflicted at all stages it is obtained by finding *the shortest path from the start node to the end node* (using an algorithm like dynamic programming [49]).

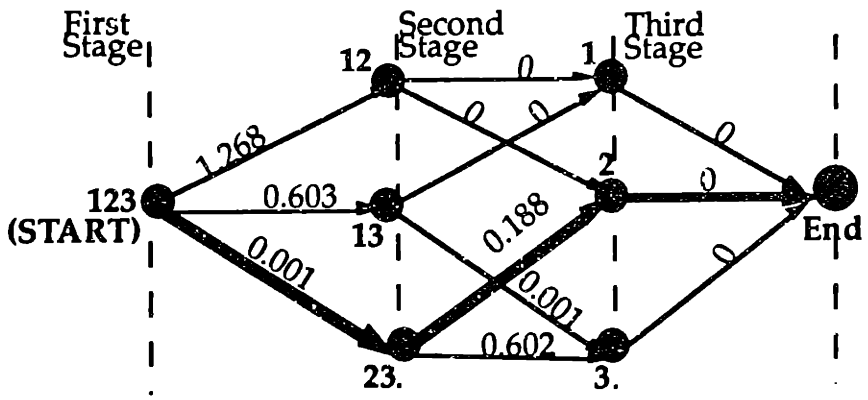


Figure 3.3: Network with Quality Losses inflicted computed from QLM (Bold lines show the shortest path which is also the optimal order)

3.8 Discussion

The analysis of decision ordering in cross-functional teams has served the following purposes:

1. Partitioning of design variables into exclusive groups has enabled the decomposition of quality loss incurred by a task into components due to loss of exclusive groups. In real design decision making processes, where the number of tasks or lifecycle considerations, n, is much less than the number of design variables, m, such a decomposition would result in substantial computational payoffs. It is noteworthy that we are interested in seeking special solutions to a nonlinear program, solutions that correspond to ordered decision making in a cross-functional group. Clearly, such solutions (to a more constrained problem) will be less optimal than the globally optimal solution to the nonlinear program.

2. In a three step approach, exclusive group structure has been exploited repeatedly to arrive at a series of sufficient conditions. First, conditions are identified under which the exclusive group is order invariant and forming task invariant. Such groups reduce the complexity of one or more design tasks. When all exclusive groups spanning a task satisfy C1 or C2.1, quality loss incurred by the task can be written as the summation of quality loss components due to loss of individual exclusive groups. In the third step, quality loss is interpreted as loss inflicted to induce stage independence in calculating the optimal order so that the shortest path of a directed network, equals the optimal decision order.

3. This analysis has interwoven connectivity and sensitivity issues together in identifying the optimal order. Partitioning into exclusive groups is done on the basis of topological connectivity. However, the concept of invariance (order invariance and forming task invariance) uses both connectivity and sensitivity. As observed earlier, both sparseness (topological notion) and monotonicity (sensitivity based idea) enhance invariance. The reasoning process reveals the utility of both these aspects and unearths the strong links between design structure and optimal decision strategies.

4. The results are incremental in their utility. If a certain exclusive group is forming task invariant, then the size of the particular forming task is reduced. If it is order invariant, then the sizes (and thereby the complexity) of all tasks

are reduced. If all exclusive groups spanning a task are invariant, then the task is not required to make decisions after having been made independent decisions. If all exclusive groups spanning every task are invariant, as in the case of dc motor, then the optimal order is obtained simply by identifying the shortest path in a network of quality losses.

5. Although the conditions stated to identify invariance apply only to specialized situations, the notion of quality loss, exclusive groups and invariance are broader and can be applied in more general cases, in particular, to future study of iterative decision strategies.

Given below is a (nonexhaustive) list of limitations of the approach:

1. The results hold for design processes with specially structured design tasks. Tasks are nonlinear programs with a continuously differentiable objective. Constraints could only be either range inequalities or equalities that could be explicitly eliminated by symbolic propagation (as in serial constraint networks). The possibility of including other type of constraints in the objective, other definitions for Quality Loss and methods for weight determination have to be investigated in future.

2. Definition of a task assumes that a task is able to decide design variables by itself, without assistance from other tasks. This model may be invalid in some design situations.

3. The conditions developed to recognize invariances are not strong in nonsparse, nonmonotonic situations.

In the next two chapters, I consider the complementary problem – to improve the performance of a design process in which activities are coupled by directed precedence constraints. I develop methods to relax the precedence constraints and thereby overlap the activities.

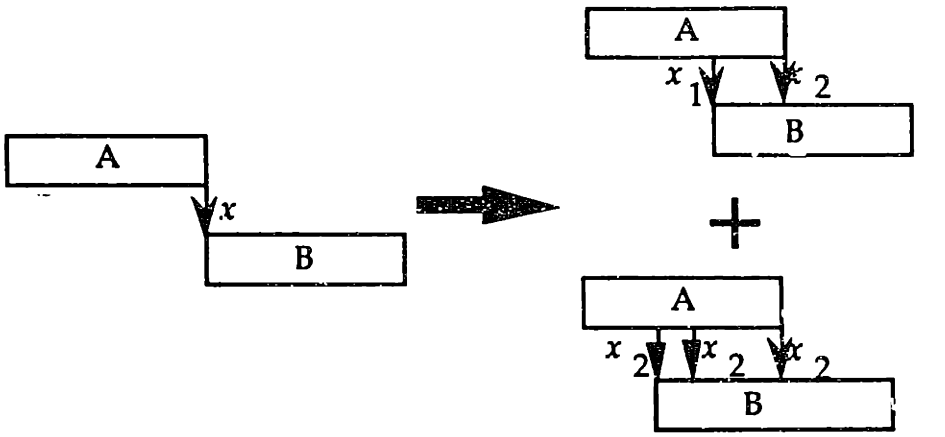
4. The Overlapping Problem

If a man will begin with certainties, he shall end in doubts; but if he will be content to begin with doubts, he shall end in certainties. - Francis Bacon

4.1 Introduction

As exemplified by the brake system DSM in Chapter 2, the flow of information in many processes in practice is largely sequential with information being generated by upstream activities, and transferred to downstream activities for further processing. In this chapter, I seek to accelerate such processes by *overlapping* the nominally sequential activities. Overlapping the activities, such that the downstream development activities start before their information–releasing upstream counterparts end, requires that the product information be made available early in a final or preliminary form. As shown in Figure 4.1, overlapping may involve both advance finalization of parts of the product information (denoted by x_1), and the preliminary exchange of other parts, x_2, that cannot be finalized early (future changes in such information should be incorporated in subsequent downstream iterations). It needs to be determined (i) which parts of the exchanged information can be frozen early and how early, (ii) which parts may be exchanged in preliminary form and with what consequences.

Overlapped processes represent the full spectrum between the two extreme cases: the purely sequential process (in which the downstream activity is phased with the upstream) and the ideal concurrent process (in which the downstream is completely overlapped with the upstream). This chapter offers an operational method to increase the overlap between activities and thereby move towards the ideal concurrent process. In the next section, I introduce the overlapping problem with a glimpse of the risk associated with overlapping. Following this, I review some related literature. In subsequent sections, I develop the notions of evolution and sensitivity, and then describe why together they determine how activities must be overlapped.



(a) Sequential Process (b) Overlapped Process

Legend
→ Preliminary Information Exchange
⇒ Finalized Information Exchange
x Information Exchanged

Figure 4.1 Transforming a Sequential Process into an Overlapped Process

4.2 The Overlapping Problem

The primary advantage that overlapping offers is the potential reduction in development time while preserving the coupling among the activities (and keeping the product integrated). The reduction in lead time (over the sequential process) may not, however, be equal to the duration during which the activities are overlapped (the overlap period). This is because the duration of the downstream activity may be different in the overlapped process than that in a sequential process: If upstream information is absorbed by the downstream activity too early, then future changes have to be incorporated in time consuming subsequent iterations – contributing to an increase in the downstream duration and development effort.

When parts of the upstream generated information are frozen early, the upstream activity loses the flexibility to make changes in these parts until its completion. This loss of flexibility to make changes and thereby to tailor the product to customer needs is interpreted as a quality loss (QL) for the upstream activity. Any attempts at overlapping the activities – by exchanging preliminary information or by freezing upstream information early – should ensure that the increase in downstream development effort and the upstream quality loss are not enormous. The problem of improving the lead time

performance, while minimizing the adverse effects on product quality and development effort, is called the overlapping problem.

Using the effect of overlapping on development performance as the basis for disaggregating the exchanged information (x into parts such as $x1$ and $x2$) can help solve the overlapping problem. Because freezing upstream information early may cause a quality loss, only those parts whose early freeze would produce very little quality loss should be frozen early. Similarly, the part of the upstream information, such as $x2$, exchanged in a preliminary form should be such that it does not cause a substantial increase in the downstream effort. Such disaggregation can ensure that development time is reduced without a large adverse impact on product quality or development effort. If the disaggregation is not done with care – the effect of overlapping on development performance is not considered – then excessive quality penalty may accrue to the upstream activity, and substantial downstream iterations may lead to an increase in the development effort, and even the development lead time. This potential for performance to worsen due to (careless) overlapping is referred to as the risk associated with overlapping.

To profit from overlapping activities, it needs to be determined freezing which parts early would not involve much quality loss, and which parts may be used in the preliminary form without much increase in the development effort. I develop a conceptual framework that would help managers decide when and how the development activities should be overlapped. The framework classifies exchanged product information on two key dimensions: (upstream information) evolution and (downstream) sensitivity. The conceptual framework and associated ideas are illustrated with application to pager development at Motorola and door development at Chrysler Corporation. Before presenting the ideas and applications, I discuss how this work relates to other research in the literature.

4.3 Related Work in Overlapping

The effect of overlapping on product innovation and noncommercial product development has been investigated by several researchers including Mansfield [50], and Eastman [51]. From their field studies, these researchers observed that overlapping stages led to an increase in development costs. Their work should help motivate the need for careful overlapping.

In the context of commercial product development, Imai, Takeuchi and Nonaka [20, 52] observed that faster development processes use a more holistic or overlapped approach. They likened the sequential process to a relay race with one group of functional specialists passing the baton to the next group and the overlapped approach to a rugby game in which a hand-picked, multi-disciplinary team travels the distance as a unit. Although the analogies are interesting, these authors (i) imply that all activities in the development process can be carried out concurrently and, (ii) do not indicate *how* a sequential or phased process may be overlapped. As noted earlier, careful consideration is required to minimize the risk associated with overlapping. Further, the technical complexity of the products developed imposes certain precedence relations among development activities due to which not all activities can be done concurrently. For instance, major resource commitments for the development of tooling and fixtures (for mass production) of an automobile cannot be made until the vehicle concept is available. The structure and strength of interactions needs to be well understood for activities to be overlapped successfully.

In their comparative research of product development practices around the world, Clark and Fujimoto [53] developed a measure called overlap ratio to show that faster development processes are more overlapped. They also recognized the organizational barriers to overlapping activities, such as hostile environment, poor communication, and the lack of consistency and balance in managing the critical linkages [54]. To facilitate overlapping, they recommended frequent, face-to-face, bilateral communication of preliminary information instead of late release of complete information [12].

These recommendations would be useful to initiate organizational changes, but the generic reference made to information exchange needs to be augmented with more specific understanding of the characteristics of the information exchanged to operationalize overlapping. Close examination of industrial product development suggests that not all information exchanges are equal. Some types of product information can be more readily used in their preliminary form than others because of their lower impact on the downstream activities. Also, some product information can be frozen early, some need to be frozen early and some others cannot be or should not be frozen early. This thesis builds on previous research by Clark and Fujimoto

in the following ways: It a) develops a methodology to overlap activities based on the specific characteristics of the information exchanged between them; b) examines the effect of overlapping on development performance with the aim of addressing the risk associated with overlapping; and c) offers specific methods to disaggregate (or "batch") exchanged product information.

Other researchers, notably Blackburn [21] and Smith and Reinertsen [22], have recognized the merits of overlapping, but not provided detailed information on how overlapping may be implemented in a coupled, nominally sequential process. Further, the effects of design interactions and the risks involved in overlapping strongly coupled activities have not been considered. Smith and Reinertsen [22] recommended that downstream designers to search for ways to start early with partial information but did not warn of considerable iterations that may result from hasty starts. Again, industrial reality suggests that overlapping involves not only the downstream activities starting early with preliminary information but also (i) the determination of the design parameters that are of high importance to the downstream activity and (ii) upstream freezing these parameters ahead of normal course of action.

Nihtila studied product development in five industries and notes the risk associated with overlapping – "... excessive overlapping of too many interdependent phases seemed to result in quite heavy iteration in determining the engineering parameters and constraints..." [55] – but offered no specific solutions. I believe this corresponds to the rework in downstream activities to incorporate changes in upstream information caused by the overlapping of activities. Proper understanding of the properties of the exchanged information would help prevent such rework-intensive iterations.

In his study of Japanese product development processes, Whitney [33] notes that the "main strategy" used by the Japanese to expedite product development is overlapping; He further adds:

> "Using incomplete information ... is risky... and requires carefully supervised release of partial design information... Companies want ways to categorize this information according to how important it is, when it is needed, and how the impact varies depending on how much the delayed information, once it arrives, deviates from what was assumed. Among the

possible difficulties are wasting time in extra design iterations or creating grounds for product liability if incorrect assumptions are not eliminated before the design is released".

This matches closely to my approach to the overlapping problem. I offer ways to categorize the information based on its availability and impact on the downstream activity. Further, I develop models of how much the delayed information deviates from what was assumed (when it arrives).

In the overlapping problem, iterations are used to incorporate upstream changes. This is only one of the many ways in which iterations can happen in product development. Smith and Eppinger [46] develop models of iterations for coupled sets of tasks, where the distinction between upstream and downstream tasks vanishes; the coupling is bi-directional and doing any task creates a certain amount of rework for all coupled tasks. Overlapping on the other hand is a method to relax the precedence constraints relating an upstream and downstream task– constraints that arise from the interactions between tasks, which are discussed in greater detail below.

4.4 The Information Processing View and Parametric Information

In developing the ideas in this chapter, I adopt the "information processing" view of product development used by several other researchers (see [53] for a summary). In this view, product development is thought of as a process of transformation of input information about customer needs and market opportunities into output information which corresponds to manufacturable designs, and functional tooling for mass production. Individual product development activities are themselves viewed as information processors – which receive input information from the preceding activities, and during the duration of the activity transform the input information into a form that would be suitable for subsequent activities. To enhance our understanding of information processing, I model the exchanged information and its transformation in greater detail below.

In practice, the information exchanged between activities takes various forms: customer needs, benchmark data, engineering specifications, part dimensions and tolerances, prototypes etc. Here, I focus on exchanged information that can be described as a collection of parameters. A parameter

is defined as a scalar piece of information that can be represented within a compact (closed and bounded) set. Examples of parametric information include part dimensions, tolerances, and customer specifications[8]. Examination of product development processes in practice suggests that most exchanged information in the engineering stages of product development can be represented as a collection of parameters. Examples of product information that cannot be represented as a collection of parameters include new product concepts whose domain is neither closed nor bounded (designers may come up with concepts whose bounds cannot be specified a priori). I denote an individual (scalar) parameter by x and a collection (vector) of parameters by $\mathbf{X}$.

Combining the information processing view and the assumption of parametric information, we obtain the following model. The information processing upstream activity A begins at t_{As}, and transforms the exchanged information until t_{Af} – when the exchanged information can be nominally finalized. This transformation in the exchanged information between t_{As} and t_{Af}, denoted by $\Delta x(t_{As}, t_{Af})$, ensures that the product developed is of appropriate quality (by meeting customer needs). At intermediate points in time, t_i, during the upstream activity, the transformation in the exchanged information is only partially complete and is denoted by $\Delta x(t_{As}, t_i)$. In general, the transformation or the change in the exchanged information between two points in time t_i and t_j, $t_{As} \leq t_i \leq t_j \leq t_{Af}$, is denoted by $\Delta x(t_i, t_j)$. (The nature of this transformation varies from one problem to another: for example, in some problems the transformation may involve narrowing of an interval corresponding to the exchanged parameter; in others, it may involve reducing the variance of the parameter and so on. Operational interpretations of the transformation of information are developed in a future section.)

In the nominally sequential process, downstream activity B begins with the finalized value of the exchanged information at time t_{Af}. When the activities are overlapped, however, the downstream activity B begins earlier at time t_i ($< t_{Af}$) – either with preliminary upstream information when the

---

[8]The value assumed by the parameter in this set need not necessarily be a point value; it could be a value within a range, but still in a closed and bounded set (eg. toleranced part dimensions).

upstream information is still undergoing changes or with frozen upstream information (finalized early). In this thesis, I argue that both the magnitude of the change in the exchanged information x and the quality loss incurred by the upstream activity (due to early freeze) depend on the *evolution* of the exchanged information x – the rate at which the transformation of the exchanged information occurs due to the upstream activity. The consequence of this change in x to the downstream activity depends on the *downstream sensitivity* to changes in x – which itself is a function of the downstream capability (how easily the downstream activity can absorb changes in x), and the tools and methodologies used in the downstream activity. Detailed understanding of upstream and downstream activities can help determine the evolution of the upstream generated parameter and the sensitivity of the downstream activity to changes in the parameter. The combination of evolution and sensitivity values determine how the pattern of information exchange needs to be modified and the activities must be overlapped[9]. In the forthcoming sections, I develop the notions of evolution and sensitivity in detail.

4.5 Evolution of the Upstream Generated Parameter

Freezing the upstream generated parameter x early results in a quality loss for the upstream activity because upstream designers forfeit the ability to modify the parameter x until t_{Af} (and thereby ensure that the product is of appropriate quality). When the parameter x is frozen at a particular point in time t_F, before t_{Af} ($t_F \leq t_{Af}$), the upstream designers are unable to make the

---

[9]In Chapters 1 and 2, we saw how design interactionscan be represented using PERT charts, Gantt Charts, or the Design Structure Matrix (DSM). Figure 1.1 shows a sample example for a process with four activities. Although this diagram indicates that activity *C* requires information from *B*, the information required by *C* may be available before *B* is finished. Even if such information is not available earlier in a finalized form, it may be available in a preliminary form. Further, it might be information to which *B* is more or less sensitive. These diagrams represent the availability of information at the completion of activities but need to be augmented to represent information in its preliminary form and the sensitivity of downstream activities to such information. It is in this regard that the notions of evolution and sensitivity are useful.

transformation or change that the parameter will undergo between t_F and t_{Af}, $\Delta x(t_F, t_{Af})$. Let the quality loss for freezing parameter x at t_F be denoted by $QL(t_F)$. I model that:

$$QL(t_F) = \phi\,(\Delta x(t_F, t_{Af}))$$

where ϕ is called the quality loss function. The quality loss function translates the change forfeited by the upstream activity into a quality loss value. It is evident that ϕ depends on a host of factors such as the market, the importance of the parameter x to the product etc. It appears reasonable to expect, however, that the larger the amount of change forfeited by the upstream activity, greater is likely to be the quality loss incurred by the upstream activity. In other words, the quality loss function is likely to be a monotonically increasing function of the amount of change forfeited by freezing the parameter x at t_F, $\Delta x\,(t_F, t_{Af})$. If this is the case, then to minimize the quality loss to the upstream activity due to early freeze of parameter x, the change in x forfeited by the upstream activity needs to be reduced as much as possible.

The change in parameter x between t_F and t_{Af}, $\Delta x\,(t_F, t_{Af})$, itself depends on how much of the transformation is complete by t_F. The term degree of evolution is introduced to measure how close the unfinalized parameter is to its final value. The degree of evolution, ε_i, at a given instance t_i (during the upstream activity's progress) is defined as the fraction of the overall transformation in the exchanged parameter x expected to occur by time t_i. Mathematically,

$$\varepsilon_i = \frac{\Delta x(t_{As}, t_i)}{\Delta x(t_{As}, t_{Af})} \qquad (4.1)$$

The above definition normalizes ε such that $\varepsilon = 0$ at the beginning of the upstream activity, and $\varepsilon = 1$ at the end of the upstream activity. Note that ε is a dimensionless quantity. As the upstream activity unfolds, the transformation is increasingly complete, and ε approaches the value of 1. A plot of ε (the degree of evolution of the parameter x) as a function of time (upstream progress), obtained from a detailed understanding of the upstream activity, is called the *evolution* of the parameter x.

Using definition (4.1), the total change in the exchanged product information between two points in time t_i and t_j for $t_j > t_i$, $\Delta x(t_i, t_j)$, is given by:

$$\Delta x(t_i, t_j) = \Delta x(t_{As}, t_j) - \Delta x(t_{As}, t_i)$$

$$= \Delta x(t_{As}, t_{Af}) (\varepsilon_j - \varepsilon_i)$$

In other words, the change in the exchanged parameter between any two points in time is proportional to the difference in degrees of evolution at these points in time. Larger this difference, greater is the change in the design parameter. If we choose one of these points as t_F, and the other as t_{Af}, then the change in the parameter represents the change $\Delta x(t_F, t_{Af})$ forfeited by the upstream activity. From the above result, $\Delta x(t_F, t_{Af}) = \Delta x(t_{As}, t_{Af}) (1 - \varepsilon_F)$ - since $\varepsilon_{Af} = 1$. This change is dependent on the value of ε_F at time t_F. The larger ε_F is at time t_F, smaller is the change forfeited by freezing the parameter at t_F and therefore the quality loss. The value of ε_F is at time t_F is however dependent on the evolution of x.

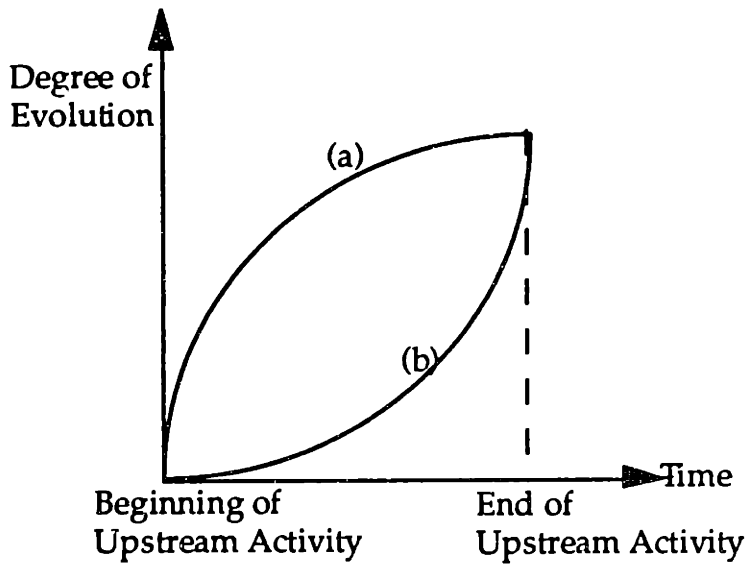


Figure 4.2: Evolution of Two Different Versions of the Upstream Activity

Figure 4.2 shows the evolution of x for two different versions of the upstream activity. It is seen that the at any point in time the degree of evolution of the parameter x in (a) is always higher than in (b). In this sense, the parameter x is said to evolve faster in (a) than in (b). It is evident that the quality loss incurred, due to freezing the parameter x at an earlier point in time, will be lower in version (a) than in version (b). The faster the

evolution of the exchanged parameter, the smaller would be the quality penalty for freezing it early. However, the evolution of the parameter does not by itself determine the magnitude of the quality loss to the upstream activity. Two other quantities, the quality loss function, ϕ, and the overall change ($\Delta x(t_{As}, t_{Af})$) impact the quality loss value as well:

$$QL(t_F) = \phi \left(\Delta x(t_{As}, t_{Af}) (1 - \varepsilon_F) \right)$$

The evolution of the parameter x is said to be rapid when the quality loss due to freezing the parameter early in the upstream process is low. The evolution of x is said to be slow when the quality loss for freezing the parameter even later in the upstream process is high. The evaluation of whether the quality loss is low or high depends on the product development process in question.

Examination of industrial product development processes indicates that the nature of the design information and the upstream process often determines if the evolution of the design information is slow or rapid. Consider for example, the pager design process at Motorola. The designers can predict, from prior experience in designing the product, that the evolution of the dimensional information (length, width and depth of the pager) is rapid; these dimensions are largely determined by competitive products, target market, volume studies and human factor studies, and can be frozen early without much quality loss for the product concept. On the other hand, the shape details of the pager, such as the corner radii evolve slowly; if any feature of the product changes, many of the shape details need to be modified to create an "integrated design".

The notion of evolution is useful not only to characterize the upstream quality loss, but also to quantify the amount of change in the parameter x that the downstream activity may expect to process in future iterations if it were to start early with preliminary upstream information. Whether this change in the parameter x will inordinately increase the downstream effort or not depends on the downstream sensitivity discussed below.

4.6 Downstream Sensitivity

The notion of downstream sensitivity is introduced to capture the consequence of the changes in parameter x to the downstream activity. *By*

downstream sensitivity, I mean the duration of downstream work required to incorporate changes in parameter x made by the upstream activity. Downstream sensitivity is said to be high if the duration of the downstream work required to incorporate changes in the parameter x is high. Lower downstream sensitivity describes the case when substantial changes in parameter x can be accommodated quickly by the downstream activity (see Figure 4.3). The function relating the increase in the downstream duration to the change in the upstream design parameter is called the *influence function.* The influence function is mathematically depicted as $\mathcal{F}$ while capturing the relation between the change and downstream iterations as, $d_i = \mathcal{F}(\Delta x(t_{i-1}, t_i))$ where d_i is the duration of the downstream work required at time t_i to incorporate the change in the design information from time t_{i-1} to time t_i.

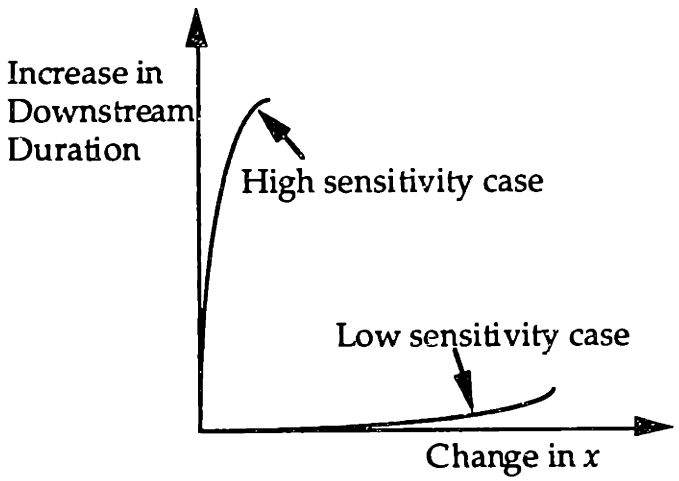


Figure 4.3: Low and High Sensitivity

Modeling downstream sensitivity may get complex. The same amount of change in the parameter x could have different effects on the downstream activity based on the design state. In such cases, the influence function is said to be state dependent. For example, consider the pager development example once again. A 2 mm reduction in the width of the pager made by industrial design (x) could have different effect on product engineering if the initial width was 200 mm instead of 400 mm. In a state-independent influence function, the increase in downstream duration is only a function of the change in the design parameter. In this thesis, for the sake of simplicity, we will confine ourselves to state-independent influence functions.

4.7 The Performance Tradeoffs due to Overlapping

In Chapter 1, I presented the framework that helps determine how to overlap activities based on the evolution and sensitivity of the information exchanged. The framework is summarized here (see Figure 4.4).

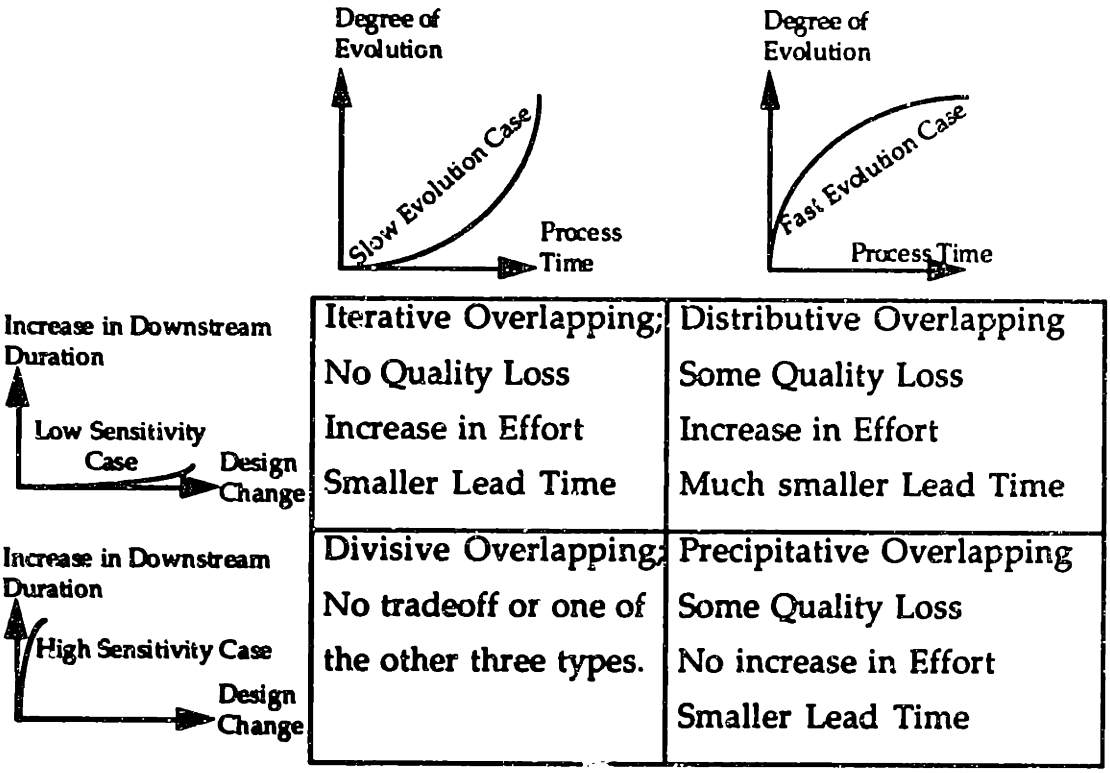


Figure 4.4: Type of Overlapping and Likely Performance Tradeoffs

- When the upstream information evolution is slow, and the downstream sensitivity is low, the overlapping is iterative; downstream activity begins with preliminary upstream information and incorporates future changes in subsequent iterations.

- When the upstream information evolution is rapid, and the downstream sensitivity is high, the overlapping is precipitative; upstream information is precipitated to its final form, and finalized before the completion of the upstream activity enabling downstream activity to start early with finalized upstream information.

- When the upstream information evolution is rapid, and the downstream sensitivity is low, the overlapping is distributive; both upstream information is frozen early, and downstream activity starts with preliminary upstream information.

- When the upstream information evolution is slow, and the downstream sensitivity is high, the overlapping is divisive; the exchanged information is decomposed into components to see if any of the components falls under the previous three categories.

In this section, I discuss the performance trade-offs that result for the different types of overlapping.

Under iterative overlapping, the upstream information is not finalized early, and so there is no loss of quality (flexibility) for the upstream activity. (In fact, there may even be an improvement in quality because the upstream activity can now obtain some feedback from the downstream activity which begins early with preliminary information.) On the other hand, the downstream effort is expected to be higher under iterative overlapping. Because the downstream wait time is reduced, savings in lead time is expected. Using the mathematical model developed in the next chapter, the beginning of downstream iterations can be chosen so as to reduce the amount of development effort, and to ensure minimum lead time.

Under precipitative overlapping, the downstream activity starts with finalized upstream information, and so there are no subsequent iterations (leading to an increase in downstream time or effort). From the upstream point of view, however, there may be a quality penalty for foregoing the ability to make design changes until later in time; there may not be an increase in effort to precipitate the information because the effort is only advanced. The lead time performance is expected to be better because the downstream activity starts early.

In distributive overlapping, the upstream activity may incur a quality loss for the finalizing the exchanged information early; an increase in downstream effort is expected because downstream begins with preliminary information. However, being the most aggressive of all the four types of overlapping, it can lead to a substantial reduction in lead time.

With divisive overlapping, there may be no effect on the performance parameters if all components of the exchanged information possess the same evolution and sensitivity patterns. However, if some components fall under

any of the previous three categories they would experience similar performance trade-offs.

4.8 Determinants of Evolution and Sensitivity

In this section, I use the pager development process at Motorola introduced in Chapter 1 as an example to illustrate what factors determine evolution and sensitivity, and how they can be altered. (Refer to Figure 1.7 for a representative pager and a pager cross section.) I describe the process in more detail before discussing evolution and sensitivity.

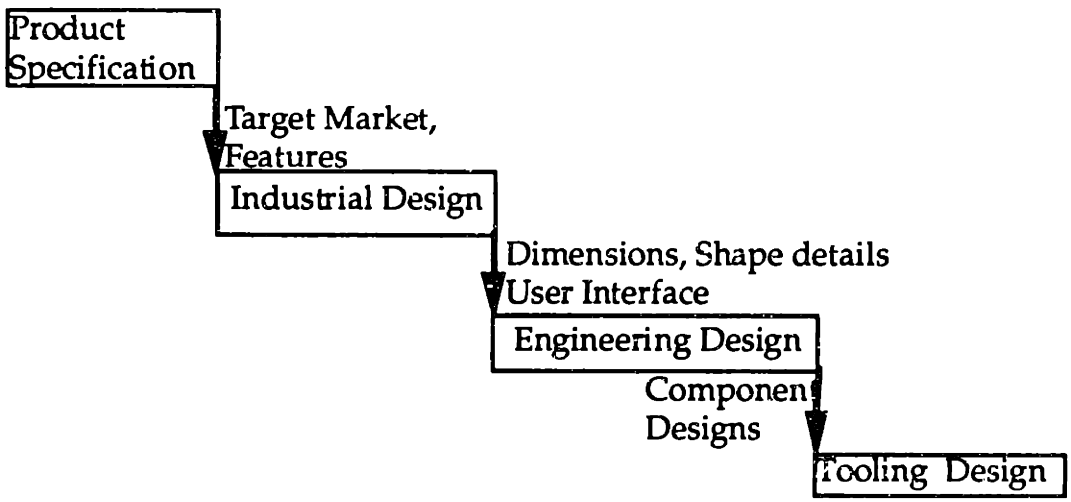


Figure 4.5: Pager Development Process Phases

As shown in Figure 4.5, the process of development of a pager involves four stages: product specification, industrial design (or styling), engineering design, and development of tooling and fixtures. During the product specification stage, the target market, pager geometric form (rectangular, hexagonal etc.), and carrying method (pocket, belt etc.) are defined. During the industrial design stage, the external dimensions, shape details and the user interface (location and size of display and buttons) are defined. The engineering design stage involves the design of the mechanical parts (walls, ribs etc.) and the electrical components (receiver and decoder design). It is noteworthy that the downstream phases are dependent on the upstream stages: industrial design of the pager requires knowledge about the target market, geometric form, and carrying method; engineering design requires additionally the external dimensions and shape details to layout the chips,

select components and design pager housing. Developing tooling and fixtures requires information about wall thicknesses and other surface formations.

In Chapter 1, I discussed the overlapping of the engineering design and industrial design phases. Specifically, I noted that the phases were overlapped by (i) early freeze of the pager dimensions (under precipitative overlapping), and (ii) preliminary exchange of the unfinalized shape details (iterative overlapping). Figure 4.6 depicts the overlapped pager development process.

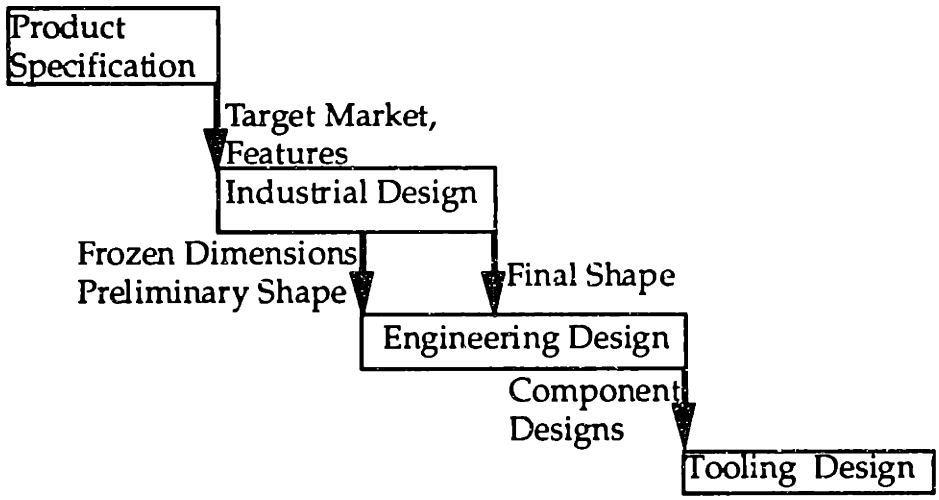


Figure 4.6: Overlapping of Engineering Design and Industrial Design

It is instructive to take a closer look at the evolution and sensitivity of the pager dimensions and the shape details. The dimensions evolve rather quickly during the industrial design phase, being determined largely by the competition, target market and technology. For instance, if the pager designed was to fit into a shirt pocket, its length (and to some extent width) are immediately constrained by standard pocket sizes. Further, aesthetics and competitive products necessitate that the width and thickness be in proportion to length. The evolution of the shape details is, however, slow as industrial designers change the corner radii to ensure consistency of shape with the rest of the features. This is confirmed by data from a recently completed process which shows that the radius had changed as much as 30% near the end of the industrial design phase, while the corresponding change in the pager dimensions was small enough to not require a subsequent engineering iteration (the pager dimensions grew slightly, but the increase in dimensions does not affect engineering adversely).

The sensitivity of engineering design activity to any decrease in dimensions is high as an altogether new layout and smaller components (manufactured with more recent or not yet available technology) may be needed to accommodate the shrinkage in the size of the case. As noted earlier, the sensitivity of engineering design to changes in the corner radii was kept low by placing the taller components in the middle. Thus communication between the functions and anticipation by the downstream function helped reduce the sensitivity of the downstream activity.

Figure 4.7 shows the evolution and sensitivity of the dimensions and shape details (enclosed in capsules) and how they may be altered (using arrows). First, better communication ensured that sensitivity of shape details were changed from high to low. With use of CAD tools the evolution of the shape details can further be accelerated. A major obstacle to the determination of the shape is the creation of the (corner) blends in the surfaces. Currently, industrial designers sketch the surface on paper, and engineers labor for weeks to input the sketched surface into the computer – in ensuring that it reflects the industrial designer's intent. New tools such as ALIAS can be used by the industrial designer to directly input the surface into the computer. This will help accelerate the evolution of shape details. Further changes can be easily made and prototypes directly fabricated, so the sensitivity further decreases.

The sensitivity of dimensions can be altered by better design practice and technology. If the engineering designers laid out the components with a clearance (the dimensions are assumed to a certain magnitude smaller than what industrial designers offer), the sensitivity of the shape details can be reduced to a certain extent. Also, designing below the current manufacturing capabilities (using chips with larger size than what is currently producible) can ensure that any surprise changes by industrial design can be taken care of. However, if market considerations dictate that the chips selected be at the frontier of available manufacturing technology (as they currently do), this suggestion cannot help reduce sensitivity. Improved communication about the likelihood of changes in dimensions can also be of use to some extent as it helps warn advanced development group of the need for new technology, and senior management of the need to identify suppliers with better manufacturing capabilities.

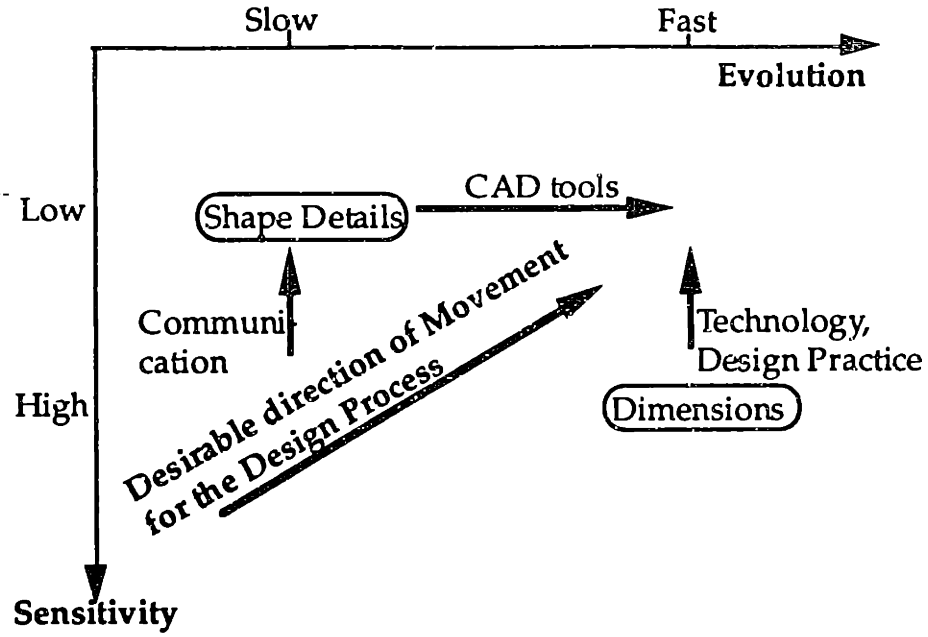


Figure 4.7: Evolution – Sensitivity Map

Figure 4.7, the Evolution–Sensitivity map, also shows the desirable direction of movement for a design process; from the point of view of overlapping, it is desirable that the evolution of the design information become faster, and sensitivity lower. I believe such a map will be useful in practice to both evaluate existing process capabilities and transform defective processes into effective processes.

4.9 Models of Transformation of Upstream Information

So far, I have referred generally to the transformation of upstream information. In this section, I offer an operational interpretation. In this interpretation, the exchanged parameter x is represented as a point value in its frozen form(e.g. $x = 7.6$). In its preliminary form the parameter x is modeled to be known within an interval (a, b), but its exact value within the interval is not known[10]. At the beginning of the upstream activity, x is known to lie within the interval (a_{in}, b_{in}), also called the initial interval. During the course of the upstream activity, as more calculations are performed and external inputs are received, the parameter x is gradually

[10]Such a model of preliminary product information has also been used by several other researchers; for example, see Ward [56] for instance.

narrowed to its final value – a point within the initial interval. In other words, we are modeling the transformation as the convergence of parameter x from an initial interval (a_{in}, b_{in}) at the beginning of the upstream activity to a point value at the completion of upstream activity (see Figure 4.8). The rate at which the interval narrows is modeled to be characteristic of the upstream activity, and should be obtained as an input from studying the upstream activity. As the upstream activity proceeds the interval within which the parameter lies changes until when the parameter attains its final value at the completion of the upstream activity.

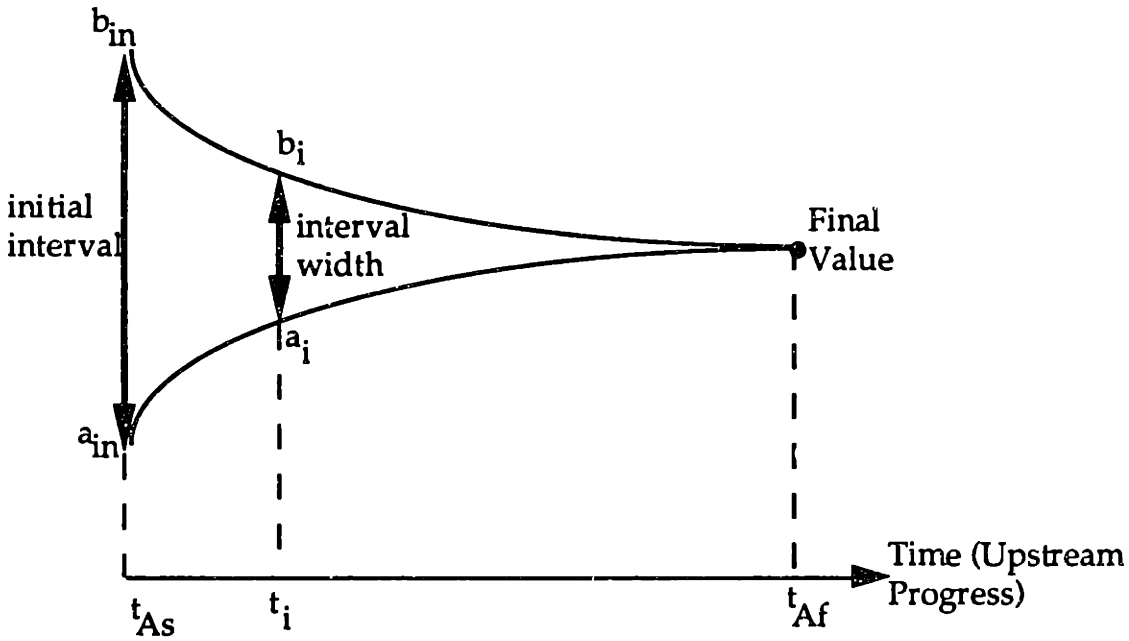


Figure 4.8: Upstream Activity Narrowing the Parameter to its Final Value

With this interval model of preliminary information, the interval width is a measure of how close the parameter is to its final value: smaller the interval width, closer is the parameter to its final value. The degree of evolution (ε) is now defined to be a linear function of the width of the design interval, with a negative slope. It can be verified that the following expression relates the degree of evolution ε_i, at time t_i, to the width of the design interval at t_i, (a_i, b_i):

$$\varepsilon_i = 1 - \frac{b_i - a_i}{b_{in} - a_{in}}$$

Once again, ε is a dimensionless quantity and is normalized such that $\varepsilon = 0$ at the beginning of the upstream activity, and $\varepsilon = 1$ at the end of the upstream activity. As before, the plot of ε as a function of time (upstream progress) gives the *evolution* of the parameter x. It is shown below that the magnitude of change in the upstream information between time t_i and t_j ($t_{Af} \geq t_j \geq t_i \geq t_{As}$), $\Delta x(t_i, t_j)$, equals: $\Delta x(t_i, t_j) = \beta\,(\varepsilon_j - \varepsilon_i)$ where β is a parameter of the design process and ε_i and ε_j are the degrees of evolution at times t_i and t_j.

Let at time t_i, when the evolution is ε_i, the design interval be $\{a_i, b_i\}$ and at time t_j, when the evolution is ε_j, the design interval be $\{a_j, b_j\}$. Using this model, it can be shown for monotonic evolution[11] (such that for $t_j > t_i$, the interval $\{a_j, b_j\}$ is a subset of $\{a_i, b_i\}$ i. e. $b_j \leq b_i$; $a_j \geq a_i$) that the maximum expected change in the design information between the two points in time t_i and t_j ($t_j > t_i$), is proportional to the difference in degree of evolution, $(\varepsilon_j - \varepsilon_i)$.

The expected value of the design information at t_i is $\frac{a_i + b_i}{2}$ and at t_j is $\frac{a_j + b_j}{2}$ (since using intervals implicitly assumes an uniform distribution). To maximize the change, I solve the optimization problem in Figure 4.9:

$$\text{Maximize} \left| \frac{a_j + b_j}{2} - \frac{a_i + b_i}{2} \right|$$

subject to:

$$b_j \leq b_i$$

$$a_j \geq a_i$$

$$(b_j - a_j) = \frac{1 - \varepsilon_j}{1 - \varepsilon_i}(b_i - a_i)$$

There are two solutions based on the sign of $\left(\frac{a_j + b_j}{2} - \frac{a_i + b_i}{2} \right)$:

(1) When $\left(\frac{a_j + b_j}{2} - \frac{a_i + b_i}{2} \right) \geq 0$, $b_j = b_i$; $a_j = b_i - \frac{b_{in} - a_{in}}{2}(1 - \varepsilon_j)$

(2) When $\left(\frac{a_j + b_j}{2} - \frac{a_i + b_i}{2} \right) \leq 0$, $a_j = a_i$; $b_j = a_i + \frac{b_{in} - a_{in}}{2}(1 - \varepsilon_j)$

Figure 4.9: Maximizing the Expected Change

---

[11] A similar result is derived for nonmonotonic evolution in the appendix A4 which shows that the change in the design information between t_i and t_j ($t_j > t_i$), is proportional to $(\varepsilon_j + \varepsilon_i)$.

In both cases of Figure 4.9, the maximum change = $\Delta x(t_i, t_j) = \frac{b_{in} - a_{in}}{2} (\varepsilon_j - \varepsilon_i)$

This result suggests that if the upstream activity was to attempt to freeze the design information at t_F, then it foregoes the ability or the need to make the expected design change, of magnitude $\frac{b_{in} - a_{in}}{2} (1 - \varepsilon_F)$, which is large for small ε_F. This confirms the intuition expressed in the previous sections that the smaller the degree of evolution, at any given instance, the harder is it to freeze the design information at that instance. The reader will also observe that this relation is very similar to (4.1) with $\Delta x(t_{As}, t_{Af})$ substituted by $\frac{b_{in} - a_{in}}{2}$. All the other ideas developed earlier in this chapter also extend directly to this interval narrowing model of the design process.

4.10 Evolution and Sensitivity as the Basis for Disaggregation

In the pager example, it was seen that disaggregating the exchanged product information into the fast evolving, high sensitivity dimensional information, and the slow evolving shape details helped overlap the activities. Thus evolution and sensitivity were the basis of disaggregation of the exchanged information to facilitate overlapping among the activities. The following guideline may be used in disaggregation: if two different parts of the exchanged information differ in evolution and/or sensitivity, separate them; if not, keep them together in the exchange of information. In the pager example, the dimensional information (length, width and thickness) were not disaggregated because they exhibited similar evolution and sensitivity patterns. The shape details which were much slower in evolution than the dimensions and were therefore disaggregated from the dimensions.

The role of disaggregation/batching of the exchanged information in overlapping activities has been discussed by Clark and Fujimoto [12]. However these authors do not discuss how or on what basis the product information should be disaggregated. Using evolution and sensitivity helps disaggregate the exchanged information into parts that may be frozen early without much quality loss, and parts which may be used in preliminary form without much increase in downstream effort. Such disaggregation is strictly motivated by performance considerations.

4.11 The Overlapping Methodology

In this section, I present a step-by-step methodology to overlap a nominally sequential process. This methodology requires a detailed understanding of the existing process, identification of time–critical information exchanges, determination of the evolution and sensitivity of the time–critical information exchanges and appropriate overlapping of the constituent activities.

Step 1: Map the nominal process. A variety of tools, such as networks, flow charts, matrices etc. can be used; here I use the DSM for illustration.

Step 2: Decompose (disaggregate) activities to see if any of the exchanged information is available earlier and if available, see if it is useful to downstream activities. Re-aggregate those activities whose completion does not make any useful information available earlier.

Step 3: Model the interactions among activities as time constraints. Identify the time critical information exchanges.

Step 4: Classify each of the time critical information exchanges based on their evolution and sensitivity (into slow/fast evolution and high/low sensitivity).

Step 5: Use the framework of Figure 4.4 to determine the appropriate overlapping strategy.

Step 6: (optional) Apply detailed models to determine how to overlap the activities – if detailed evolution, sensitivity data are available.

Step 7: Update process and repeat steps 3 to 6.

Below is an example illustrating the application of the methodology to the development of an automotive door shown in Appendix A2, Figure 1.

Step 1: Begin by mapping the nominal process. The DSM mapping of the door panel design and die development process is shown in Figure 4.10. Activities must be defined such that the information exchanges in the current process occur at the completion of the defined activities. If not defined correctly, the DSM might depict some information exchanges which do not occur in the current process. For example, if the Develop Wireframe and Surface activity is defined separately as two activities, Develop Wireframe and

Develop Surface, it might indicate transfer of wireframe information as soon as the design was complete, which did not occur in the current process.

Generate Theme	•							
Develop Wireframe and Surface	x	•						
Develop VERCAM		x	•					
Modify Surface			x	•				
Modify VERCAM				x	•			
Develop Panel Draw	x					•		
Develop Draw Dies						x	•	
Fabricate Soft Dies					x		x	•

Figure 4.10 DSM showing Mapping of the Nominal Door Process

Step 2: To overlap, we need to know if any of the exchanged information is available in a finalized form before any of the activities are complete. So I disaggregate the activities documented in Step 1. For example, Generate Theme is divided into Develop Theme and Digitize Theme and similarly, Develop Wireframe and Surface is disaggregated into four separate activities (see Figure 4.11) and the VERCAM building activity is divided into several activities. Such a division shows that VERCAM building can start with the theme information (the vendors can be identified) and can use wireframe for blocking and surface information for coding. Note however, that the division of Generate Theme into Develop Theme and Digitize Theme was not useful because the developed theme is not of use to any other activity other than Digitize Theme. I reaggregate those activities (within a function) which do not provide any information to other functions (see Figure 4.12). The DSM shown in Figure 4.12 is now of manageable size and at the same time ensures that all information is used as soon as it is available in a final form. I call it the information-based DSM or IDSM.

Step 3: Although we could overlap any pair of activities, it is most useful to overlap activities that are on the critical path. I find the subset of the IDSM activities that are on the critical path by conventional project management (see Appendix A2 for calculations). Each mark in Figure 4.13 represents a critical information exchange (no slack in them).

Develop Theme	•																
Digitize Theme	x	•															
Dev. Boundaries, P1&P2 lines			•														
Trim Surface & Dev. Flanges			x	•													
Develop Wireframe				x	•												
Develop Surface			x	x	x	•											
Write Quote Package		x					•										
Invite Quote							x	•									
Place Quote								x	•								
Identify vendors									x	•							
Assemble Stack ("Blocking")		x								x	•						
Develop Code					x	x					x	•					
Cut VERCAM												x	•				
Modify Surface													x	•			
Modify VERCAM														x	•		
Develop Panel Draw		x														•	
Develop Draw Dies																x	•
Fabricate Soft Dies															x		x	•

Figure 4.11: Less Aggregated DSM of the Door Development Process

Generate Theme	•											
Develop Wireframe	x	•										
Develop Surface	x	x	•									
Identify vendors	x			•								
Assemble Stack ("Blocking")		x		x	•							
Develop Code		x	x		x	•						
Cut VERCAM						x	•					
Modify Surface							x	•				
Modify VERCAM								x	•			
Develop Panel Draw	x									•		
Develop Draw Dies										x	•	
Fabricate Soft Dies									x		x	•

Figure 4.12: Re-aggregated DSM ("IDSM")

Generate Theme	•			
Develop Panel Draw	x	•		
Develop Draw Dies		x	•	
Fabricate Soft Dies			x	•

Figure 4.13: Door DSM with critical information exchanges

Step 4: I classify each critical information exchange on the basis of its evolution and sensitivity (see Figure 4.14). For the door panel, all time–critical information are found to be slow in their evolution and high in sensitivity. According to the framework, the exchanged information needs to be disaggregated. Here, for illustration purposes, I consider the panel draw information (exchanged between the Develop Panel Draw activity and the Develop Draw Dies activity). It is divided into the panel periphery and the panel interior information. Upon division, we find that the panel periphery information is a rapidly evolving, high sensitivity information while the panel interior evolution is slow and sensitivity is low. In Figure 4.15, I label the marks in the "Develop Draw Dies" row with Fast/High (for the panel periphery), and Slow/Low (for the door panel interior).

Generate Theme	•			
Develop Panel Draw	Slow/High	•		
Develop Draw Dies		Slow/High	•	
Fabricate Soft Dies			Slow/High	•

Figure 4.14: DSM with critical information exchanges and Evolution-Sensitivity estimation

Step 5: Using the framework, the panel periphery information needs to be precipitated or frozen in advance while the interior draw needs to be used in preliminary form (iterative overlapping). We use

detailed models that help determine how such overlapping should be carried out.

Generate Theme	•				
Develop Panel Periphery	Slow/High	•			
Develop Panel Interior	Slow/High		•		
Develop Draw Dies		Fast/High	Slow/Low	•	
Fabricate Soft Dies				Slow/High	•

Figure 4.15: DSM after Divisive Overlapping

Step 6 (optional): The application of the models is illustrated in the next chapter. It will be shown that by using the models, the outer panel development time can be reduced to 18 weeks from the original time of 26 weeks (see Figure 4.16 for the overlapped configuration).

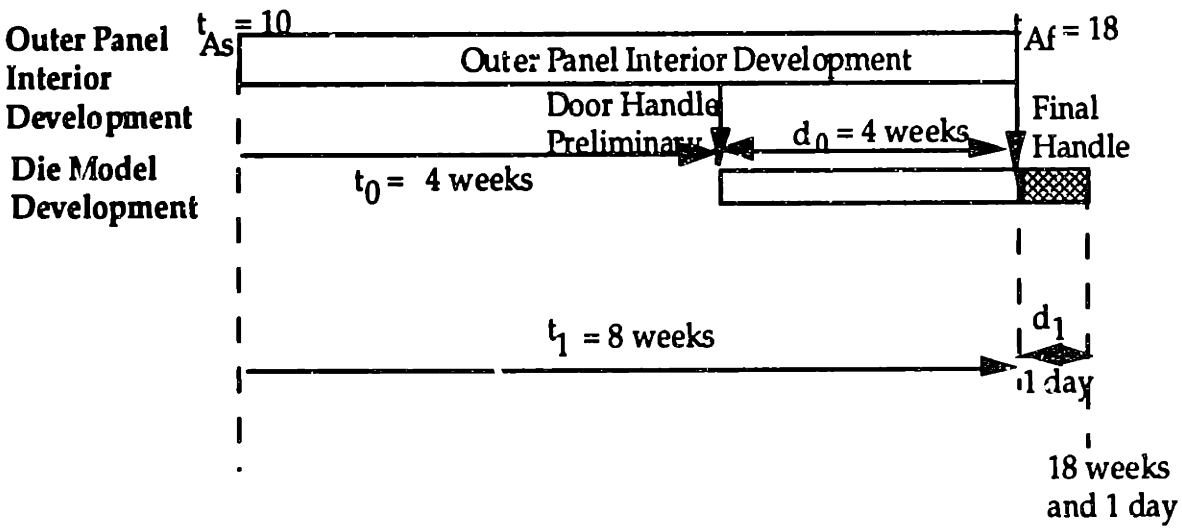


Figure 4.16: Optimal Overlapping Policy for Door Development Identified after applying Detailed Models described in the Next Chapter

Summary

In this chapter, I developed a method to categorize the information exchanged on two dimensions: upstream evolution and downstream sensitivity. The combination of evolution and sensitivity values helps decide how the activities should be overlapped. Each type of overlapping results in a different type of trade-off among the performance parameters. In the following chapter, I develop mathematical models of the performance trade-offs. As we will see, the mathematical model helps unify the different types of overlapping; it also offers guidance in deciding when to finalize upstream information, and how to exchange preliminary information when detailed evolution and sensitivity data are available.

detailed models that help determine how such overlapping should be carried out.

Generate Theme	•				
Develop Panel Periphery	Slow/High	•			
Develop Panel Interior	Slow/High		•		
Develop Draw Dies		Fast/High	Slow/Low	•	
Fabricate Soft Dies				Slow/High	•

Figure 4.15: DSM after Divisive Overlapping

Step 6 (optional): The application of the models is illustrated in the next chapter. It will be shown that by using the models, the outer panel development time can be reduced to 18 weeks from the original time of 26 weeks (see Figure 4.16 for the overlapped configuration).

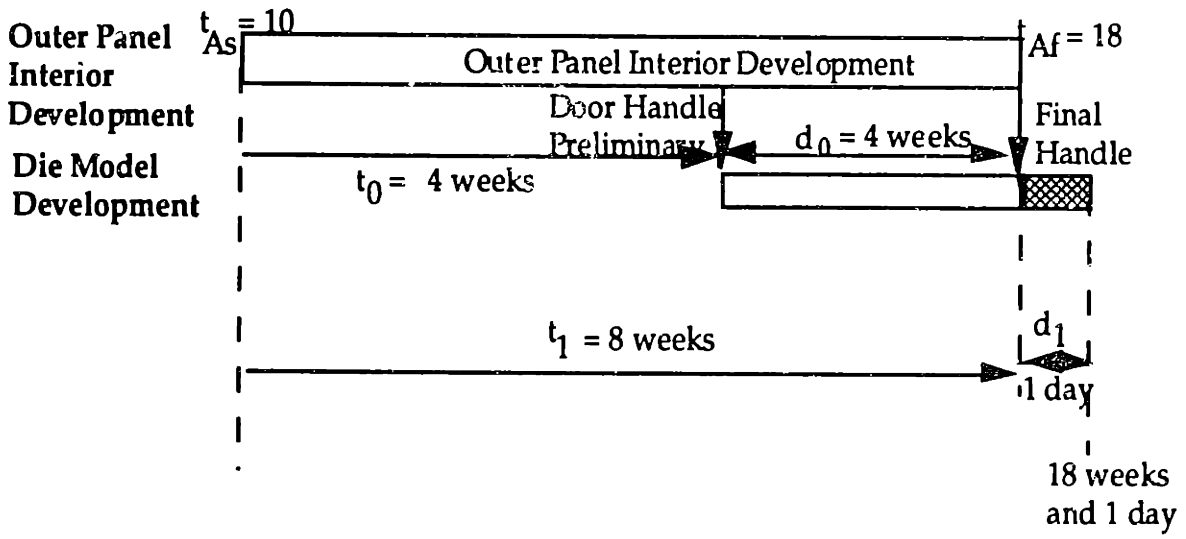


Figure 4.16: Optimal Overlapping Policy for Door Development Identified after applying Detailed Models described in the Next Chapter

5. Models of Performance Trade-offs

Mathematical models are the key to quantitative design– Neville Hogan in Lectures on "Modeling, Analysis and Control of Physical Systems" at MIT

In this chapter, I model the trade-offs among the process performance parameters, quality, development time and effort, resulting from overlapping activities. These trade-offs, briefly discussed in the previous chapter, are formulated as a mathematical model. The model helps unify the different types of overlapping, and understand the relationships between overlapping and performance trade-offs. I first discuss the constraints and objectives of the model, then derive some "intuitive results", and finally illustrate the application of the model to door development and instrument panel development in industry.

5.1 Constraints Relating Overlapped Activities

Figure 5.1 illustrates the features of the problem I am studying. Upstream activity A begins at time t_{As} to generate design information x that will be used by downstream activity B (In the rest of this paper, t_{As} is taken to be zero, the origin of the time scale).

In the nominally sequential process, the upstream-generated information x is not finalized until time t_{Af}. The downstream activity B completes a "planned iteration" of given duration d_0, upon receipt of finalized information x from A at time t_{Af} – resulting in a total development lead time of $t_{Af} + d_0$. I assume no subsequent downstream iterations exist in this approach[12], as there are no changes in the value of x ($n = 0$ and $t_F = t_{Af}$).

I model the (distributive overlapping) situation in which both the downstream activity may begin with preliminary upstream information (incorporating future changes in subsequent iterations), and the upstream information may be finalized early. (It can be shown in later sections that iterative and precipitative overlapping are special cases of this model.) The

[12] Any internal downstream iterations are assumed to be included in the planned iteration.

model helps decide how many downstream iterations to do, when to start these iterations, and how early the upstream information should be frozen.

If upstream information is finalized early, there may be a loss of quality (flexibility) for the upstream activity, modeled as a *quality loss*. Downstream effort may also be higher when more iterations are performed. Further the physical expense incurred (such as cost of metal, machining etc.) in performing subsequent downstream iterations needs to be considered. As we will see later, the objective of the model includes the cost of quality loss, lead time, iterations, and engineering effort.

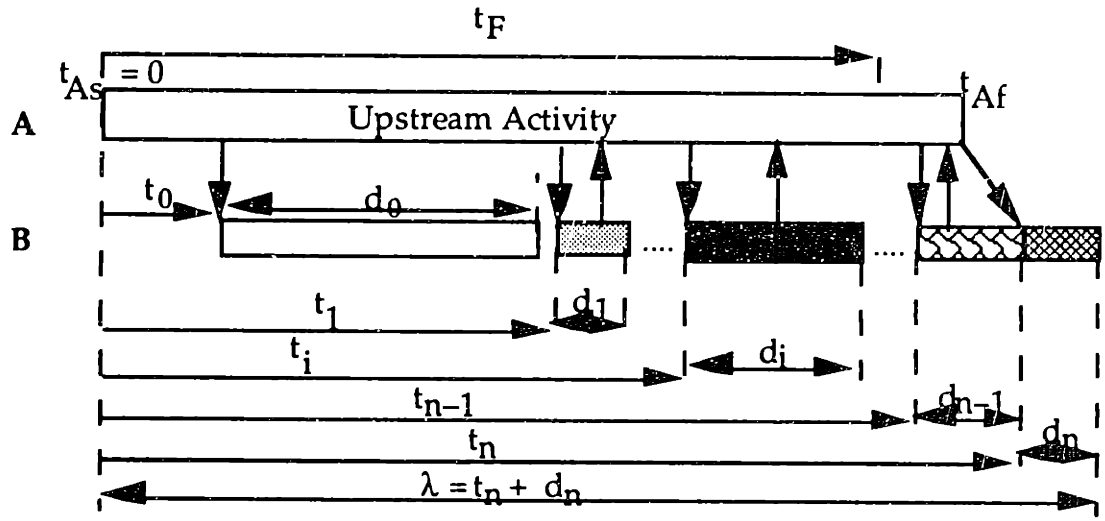


Figure 5.1: Model of Interaction for Overlapping

Legend: Shaded blocks indicate downstream iterations in progress. White space between shaded blocks denotes wait time between iterations. Dots between blocks (...) denote a series of iterations not shown in an expanded form. Arrows indicate information exchange between upstream and downstream activity.

Notation

t_i	= Start time of i^{th} iteration (decision variable)
d_i	= Duration of i^{th} iteration (decision variable)
t_{As}	= Start time of upstream activity (taken as origin of the time scale)
t_{Af}	= Nominal Finish time of upstream activity (input)
t_F	= Advance freeze time of upstream activity (decision variable)
λ	= Product Development Lead Time(decision variable)
n	= Number of subsequent iterations (decision variable)

In the overlapped process, the upstream generated information may not be let to evolve until t_{Af}, but finalized at an earlier point in time, t_F, chosen

by the designer with the aid of the model developed. The associated quality loss will be modeled in the next section. We have $t_F \leq t_{Af}$.

Also, the upstream activity A may exchange unfinalized design information x with the agreement that changes in the exchanged information will be incorporated in subsequent downstream iterations. Downstream activity once again starts with a planned iteration of duration d_0, but at time t_0, also chosen by the designer using the model developed. I define the downstream activity such that the planned iteration can start no earlier than the start of the upstream activity. (If portions of the downstream activity can be started earlier, they are excluded from the downstream activity.) This leads to the constraint $t_0 \geq t_{As}$.

Each subsequent downstream iteration i ($i > 0$), of duration d_i, starting at t_i is performed with the value of the design variables at time t_i and helps incorporate the change in the value of x from the start of the previous iteration (time t_{i-1}) to the start of the current iteration (time t_i); to facilitate learning from iterations, all changes occurring *during* a downstream iteration will be batched together and incorporated in the next downstream iteration. Let the engineering change in the value of x between times t_{i-1} and t_i be denoted by $\Delta x(t_{i-1}, t_i)$. In the next section, I model this change and the duration of the resulting iterations, d_i.

Because successive iterations are performed by the same downstream activity, iterations follow each other in a sequence (no two downstream iterations may overlap). This leads us to the constraint: $t_i \geq t_{i-1} + d_{i-1}$. If the constraint is not satisfied as an equality, it means that there is a non zero wait/idle time between iterations.

Because the value of x is finalized at t_F, exactly one downstream iteration is required (to start) after upstream information is frozen to incorporate the change that happened from the previous iteration ($t_n \geq t_F$). More than one iteration is not possible because no more changes occur in the exchanged information. So all iterations except the last start before t_F:

$$t_{As} < t_i < t_F, \quad i = 1, 2, \ldots, n-1$$

Collecting these constraints together, we have:

$t_F \leq t_{Af}$		(Information may be frozen before t_{Af})
$t_i \geq t_{i-1} + d_{i-1}$	$i = 1, 2, ..., n$	(Idle time between iterations may $\neq 0$)
$t_0 \geq t_{As}$		(Planned iteration starts after t_{As})
$t_{As} < t_i < t_F$	$i = 1, 2, ..., n-1$	(B iterations start during activity A)
$t_n \geq t_F$		(One last iteration starts after t_F)
$\lambda = t_n + d_n$		(λ is the development lead time)

The above constraints are, what I call, generic constraints whose algebraic form is independent of the product development situation. Now, I consider some problem-specific constraints whose algebraic form is dependent on the input parameters.

5.2 Design Iterations

As I noted earlier, changes in the exchanged information are incorporated in subsequent downstream iterations. By models of design iteration, I mean relationships between the durations of these iterations and the design change that causes them. Clearly, this is dependent on the product development situation in question. In this section (as an example) I use the state-independent influence function, discussed in Chapter 4, to model iterations in a class of product development processes. Recall that for such a function, the duration of the iteration is only a function of the magnitude of the design change, and is independent of the design state. The formulation developed in the previous section could accommodate other models of design iterations as well.

In the state-independent case, the duration of the downstream iteration incorporating the design change is modeled as:

$$d_i = \mathcal{F}(\Delta x(t_{i-1}, t_i))$$

where the function $\mathcal{F}$ is called the *influence function* and is obtained as an input from the designer. Figure 5.2 illustrates the influence function for an automotive door panel die model development process: a change in the door handle depth of up to 2 mm requires a downstream duration of 1 day, to update drawings, fill engineering change orders and get approvals. However, if the changes in the value of depth are beyond 2 mm, the die developers are not sure if the model developed would still be feasible, they need to update or

"wash off" the plaster model developed which could take three weeks (see appendix A2).

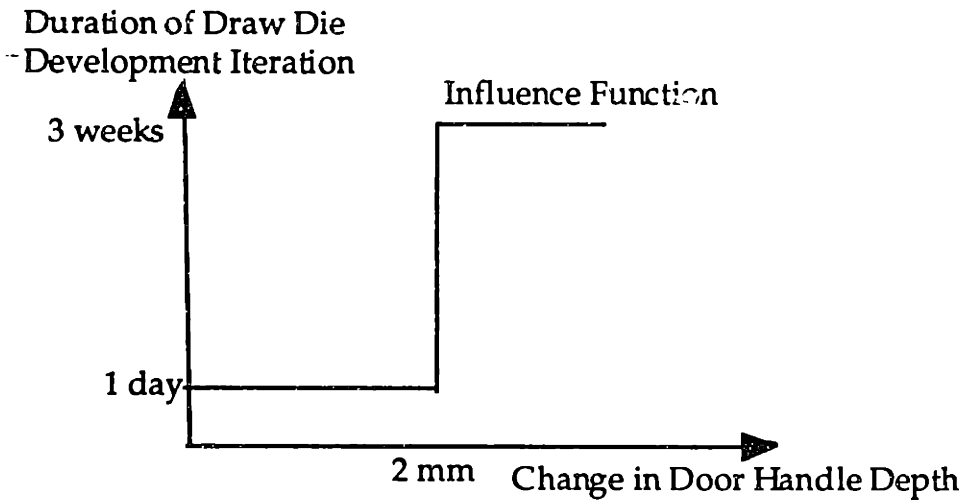


Figure 5.2: Influence Function For a Door Outer Panel

5.3 Design Change

The exchanged information changes with time, as the upstream activity progresses until when it is finalized at the completion of upstream activity. If design information were to be exchanged in advance, then future changes have to be incorporated in subsequent downstream iterations. The magnitude of the change is a key determinant of how the activities are overlapped. If for example, the exchanged information does not change much and has a small influence on the downstream activity, then the situation is a promising candidate for the exchange of advance information. If, on the other hand, the exchanged information has a large impact on the downstream activities or changes much or both, then it may not be advisable to exchange advance information. I have already modeled the impact of change on the downstream activity using the influence function. In this section, I use the notion of evolution presented in the previous chapters to obtain the magnitude of design change. It is noted that the constraint-based model presented in Section 5.1 is itself independent of the model presented in this section and can use other models of change as well.

In the previous chapter, I presented two different models of upstream information evolution: (i) degree of evolution is proportional to the width of the design interval at a given instance, and (ii) degree of evolution is

proportional to the fraction of the upstream problem solved at a given instance. Using both the models I related the degree of evolution and the magnitude of the design change as:

$$\Delta x(t_{i-1}, t_i) = \beta^* (\varepsilon_{i-1} - \varepsilon_i)$$

where $\beta = (b_{in} - a_{in})/2$ in case (i) above; $\{a_{in}, b_{in}\}$ is the initial design interval and $\beta = \Delta x(t_{As}, t_{Af})$ in case (ii) above; $\Delta x(t_{As}, t_{Af})$ is the total change in the design information during the course of the upstream process. The evolution function is obtained by studying the upstream activity in detail, and estimating, with the designer's aid, the degree of evolution at intermediate points in the upstream activity.

It is noteworthy that the model in case (i) assumes that the design interval evolves in a monotonic fashion (see Chapter 4). In appendix A4, I relax this assumption and consider a more general (but less intuitive) model.

5.4 The Overlapping Model Objective

In this section, I consider the individual components of the overlapping model objective: the quality loss incurred by the upstream activity, upstream and downstream development effort, number of iterations and the lead time.

- If the upstream generated information were to be frozen at t_F, ahead of the normal time of freeze t_{Af}, the change in the exchanged information from t_F to t_{Af} is estimated from previous section to be: $\Delta x(t_F, t_{Af}) = \beta (\varepsilon_{Af} - \varepsilon_F) = \beta^* (1 - \varepsilon_F)$. By freezing the information at t_F, the upstream activity foregoes the flexibility to make the change of this magnitude. The quality loss due to this loss of flexibility is given by: $QL = \phi(\beta^*(1 - \varepsilon_F))$, where ϕ is the Loss Function determined by the product and its market.

- The development effort is taken to be proportional to the time for which the upstream and downstream functions are committed. It is easily seen that the upstream function is committed for the period of time $(t_{Af} - t_{As})$, while the downstream function is committed for the time equal to $(\lambda - t_0)$.

- As indicated earlier, the development lead time, measured as the time distance separating the start of the upstream activity from the completion of the downstream activity, equals $t_n + d_n$.

- The cost of downstream iterations is dependent on the nature of the individual iterations. In the general model, I take the cost of iterations to be proportional to the number of iterations. When the individual iterations are very different, as in the door panel example considered later, the iterations are classified into different types, and the cost of iteration is formulated to be a weighted sum of the number of each type of iteration.

The objective is to minimize the weighted combination of lead time (λ), number of iterations (n), downstream effort (proportional to $\lambda - t_0$), upstream effort (proportional to $t_{Af} - t_{As}$), and the upstream quality loss (QL). Based on a host of factors including the product market, and the development organization, each of these goals will have different relative costs. Mathematically, the objective is:

$$\text{Minimize} \quad \omega_1*\lambda + \omega_2*n + \omega_3*(\lambda - t_0) + \omega_4*(t_{Af} - t_{As}) + \omega_5*QL$$

A word about the weights. The dimensions of the weights will be such that they convert each of the performance parameters into a $ value. Determination of the weights may be done using estimates or such techniques as conjoint analysis, empirical studies etc. One such study is reported in Clark et. al [7] where they show that the cost of each day saved in time-to-market (ω_1) of a small car is equal to a million dollars.

5.5 Optimal Overlapping Policy

Our interest is in determining when to freeze the upstream information, and how (when) to utilize or commit the upstream information for executing the downstream activity while minimizing the objective function. In mathematical terms:

Choose $P = \{t_F, n, t_0, t_1, t_2, \ldots, t_n\}$ to

$$\text{Minimize} \quad \omega_1*\lambda + \omega_2*n + \omega_3*(\lambda - t_0) + \omega_4*(t_{Af} - t_{As}) + \omega_5*QL$$

subject to constraints:

$$t_i \geq t_{i-1} + d_{i-1} \qquad i = 1, 2, \ldots, n \qquad (1)$$

$$t_0 \geq t_{As} \qquad (2)$$

$$t_{As} < t_i < t_F \qquad i = 1, 2, \ldots, n-1 \qquad (3)$$

$$t_n \geq t_F \qquad (4)$$

$$t_{Af} \geq t_F \qquad (5)$$

$$\lambda = t_n + d_n \qquad (6)$$

$$\Delta x(t_{i-1}, t_i) = \beta^* (\varepsilon_i - \varepsilon_{i-1}) \quad (7)$$

$$d_i = \mathcal{F}(\Delta x(t_{i-1}, t_i)) \quad (8)$$

$$n = \{0, 1, 2, ...\} \quad (9)$$

Because the number of iterations n is discrete, the above problem is an integer programming problem. Once the evolution and influence function are obtained as inputs, we could solve the above problem and determine when iterations should start and how many iterations should be done.

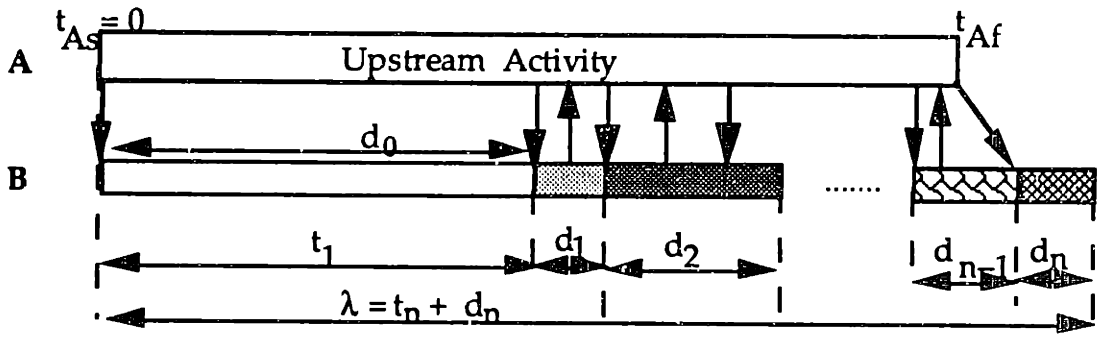


Figure 5.3: Ideal Concurrent Process: Downstream Activity completely overlapped with the Upstream Activity with no idle time between iterations

5.6 Ideal Concurrent Process

Figure 5.3 shows an ideal concurrent process for which the overlapping is complete with no idle time between iterations; note that this corresponds to the case when all of the constraints (1) and (2) in the above model are active. Had any of the constraints (1) or (2) been inactive (not satisfied as an equality), it implies that minimum lead time is attained *not* with maximum overlapping. Under these circumstances, waiting rather than iterating is more beneficial towards reducing lead time.

Below I show how to interpret the framework presented in the previous chapter using the model developed in the previous section.

5.7 The Overlapping Framework and the Models

In the last chapter, I presented the overlapping framework in which the information exchanged was classified based on its evolution and its impact on the downstream activity. For instance, when the upstream evolution is slow and the downstream sensitivity to the exchanged information is low, the development activities were to be iteratively overlapped; when the upstream

evolution is fast, and the downstream sensitivity is high, the activities were to be precipitatively overlapped. The framework can be explained using the mathematical model as follows.

Notice first that iterative overlapping corresponds to the special case when $t_F = t_{Af}$ and $n > 0$. In other words, the upstream generated information is not finalized until its normal time of finalization, and one or more subsequent iterations are performed. Also precipitative overlapping corresponds to the situation when $t_{Af} > t_F$ and $n = 0$ (upstream information is finalized early, and no subsequent iterations are done).

The upstream information evolution is said to be slow, when the quality penalty for freezing the upstream generated information at $t_F < t_{Af}$ is very high; This loss may be reflected in either the Loss Function, ϕ, or in the weight ω_5 that converts the quality loss term into a cost of quality loss. In the extreme case, when the quality penalty for early freeze is enormous (approaches a large value), the quality loss term dominates the objective function requiring $t_F = t_{Af}$. When the evolution is fast, the information gets closer to its final form well before the completion of the upstream activity and the quality penalty for early freeze is low, thereby not requiring that the upstream information be not frozen until t_{Af}.

When the downstream sensitivity is high, small upstream changes cause large, expensive changes again reflected in the algebraic form of F or the weight ω_2. In the limit, when the cost of iterations is prohibitively high, no subsequent downstream iterations can be done, requiring that the number of downstream iterations, n, be equal to zero.

It is now easily seen that in the extreme case of slow evolution and low sensitivity, iterative overlapping with nonzero (subsequent) downstream iterations, and normal upstream information freeze ($t_F = t_{Af}$) is appropriate. When the evolution is fast, and sensitivity is high, the approach involves freezing the upstream information early, and beginning downstream activity with finalized upstream information. When the evolution is slow, and sensitivity is high, $n = 0$, and $t_F = t_{Af}$, the process should be phased. When the evolution is fast, and sensitivity is low, $n > 0$, and $t_F < t_{Af}$; overlapping is distributed between both the activities.

Without specific algebraic forms of the evolution and influence functions, we could derive bounds for the performance parameters.

5.8 Bounding Lead Time and the Iterations for Iterative Overlapping

Based on our model, it is simple to find a lower bound for lead time and an upper bound for the number of iterations, for iterative overlapping when $n > 0$, and $t_F = t_{Af}$.

Suppose that the minimum duration of any iteration (subsequent to the nominal iteration) is d_{min}. This could be determined from the influence function or from the design problem under consideration. (For example, in the door stamping influence function, shown in Figure 5.2, d_{min} = 1 day, the time required to fill, store and disseminate engineering change orders). Note that our model requires exactly one iteration after the upstream activity is done. Further, the lowest possible duration of this iteration is d_{min}. The lower bound on lead time equals $t_{Af} + d_{min}$.

$$\lambda \geq t_{Af} + d_{min}. \qquad (11)$$

An upper bound on the number of iterations occurs under the case where the planned iteration is followed without any waiting by several iterations each of duration equal to d_{min}. Hence the maximum number of iterations equals $(t_{Af} - d_0) / d_{min} + 1$ (to account for the last iteration).

$$n \leq \frac{t_{Af} - d_0}{d_{min}} + 1 \qquad (12)$$

These bounds help generate the following insights:

(1) If a particular overlapping policy has lead time equaling $t_{Af} + d_{min}$, then we can use this solution to prune other branches.

(2) If the number of iterations are not bounded in a particular design problem, we can use the above upper bound.

(3) When $d_{min} \geq d_0$, it is better to do the process sequentially, than overlap because incorporating the smallest change takes as much time as the planned iteration.

(4) In a sequential process, $\lambda = t_F + d_0$. For an overlapped process, $\lambda \geq t_F +$

d_{min}. The maximum time advantage gained by iterative overlapping is $d_0 - d_{min}$.

In the next section, I apply the above ideas to the door development process.

5.9 Door Panel Design: Door Handle Case

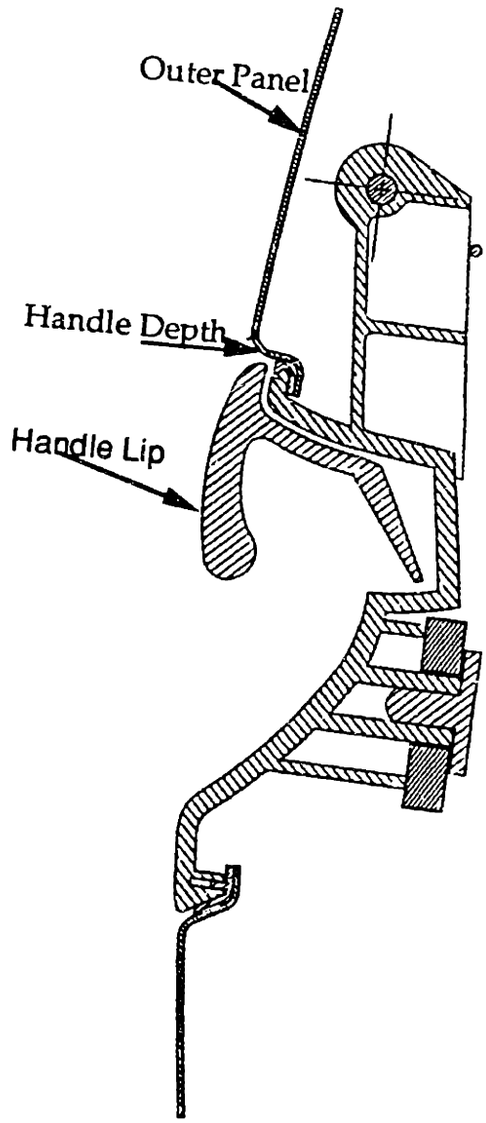


Figure 5.4: Door Handle Depth in the Outer Panel

In Chapter 1, I introduced the door design application, and observed that the outer panel draw design (upstream) and die model development (downstream) activities are coupled by the fast evolving, high sensitivity panel periphery information, and the slow evolving, low sensitivity handle information. In this section, I consider the overlapping of the outer panel

draw design and die model development activities by optimal utilization of the door handle information.

Figure 5.4 shows a door handle attached to the left side door panel of a modern day automobile. Interviews with panel designers suggests that early freeze of the door handle can entail a high quality loss for the upstream designers as the door handle may then resemble some recently introduced competitor product, and door handles being very visible parts in a car, differentiation is important. The evolution function of the door handle, obtained as an estimate from the designer, is given in Figure 5.5 – where it is seen that the intermediate activities, analyses and consultations, help establish the door handle depth. Because freezing the handle early may cause a large quality loss, the evolution is said to be slow. The door handle depth into the outer panel (x) is however used in the die design and development, and if die developers wait until the decision is finalized by the upstream designers, the development process will be delayed enormously.

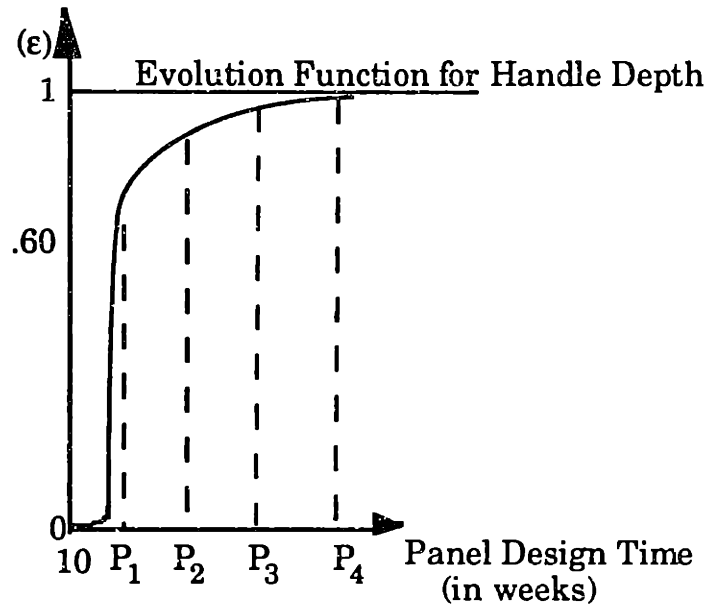
The sensitivity of die model development activity to the door handle depth is however, not high; small changes in the door handle depth require mere updates of the appropriate engineering drawings, and requisite approvals. Changes larger than 2 (mm) in the depth are likely to exceed the safety factor in die development, and the die design and development have to be updated – a process that can take as much as three weeks as described in Appendix A2. Although there are two variables, handle profile and depth, there is only one degree of freedom for die developers; the larger the depth, the less sharp (smaller radius of curvature) the profile can be[13].

From the above, it is seen that when overlapping panel design and die development, it is desirable that the door handle be not frozen early (as the evolution is slow), and die engineers use advance door handle information to start the die development activity incorporating changes in subsequent downstream iterations (as the sensitivity is not high). So the door handle

---

[13]When the panels are being stamped, metal has to be drawn from the boundary into the pocket to create it. Larger the depth, the harder is it to control the flow of metal at the sharp radiuses, and may well result in "tearing of the metal".

information exemplifies iterative overlapping of panel design and die development.



For the Door Example:

P_1 : Styling Input on Handle Pocket Profile (12th week; $\varepsilon = .60$)

P_2 : Finite Element Analysis (14th week; $\varepsilon = .75$)

P_3 : Vendor Feed back Available (16th week; $\varepsilon = .85$)

P_4 : Stamping Preliminary Feedback in (18th week; $\varepsilon = 1$)

Figure 5.5: Evolution Function for the Door Handle Depth

I will call the iterations requiring one day small iterations , and those requiring three weeks large iterations. Number of subsequent small iterations is denoted by n_s, and the number of subsequent large iterations is denoted by n_l. I determine the optimal policy for overlapping panel design and die development under the condition that any number of small iterations can be done but (to limit costs) no more than two subsequent large iterations are allowed ($n_l \leq 2$).

For the door handle case the nominal duration for die development is estimated by the engineers to be equal to four weeks ($d_0 = 4$); the initial design interval for the depth, $\{a_{in}, b_{in}\}$, is taken to be equal to {2, 12} mm, based on previous experience in designing door handles (at the start of the design

process it is known that the handle depth into the outer panel cannot be larger than 12 mm or smaller than 2 mm).

The Optimal Overlapping Policy

Our goal is to determine the optimal design information utilization policy – the time at which the upstream door handle design information should be committed for die model development. Because we consider the iterative overlapping case, where the upstream information is not frozen early, the quality loss incurred by the upstream designers is zero. Further our study company used a dedicated, platform team; the engineering effort is committed through the cycle (not borrowed for a period of time and therefore no resource contention), so the upstream and downstream effort is ignored from the objective. The cost of conducting a small iteration (updating an engineering drawing) is negligible. Also in model development[14], the physical cost of implementing changes in plaster is negligibly small compared to the cost of delay in development time (estimated to be $1 million per day by [7]). The only term that is substantial in the objective is the cost of development lead time, λ. Its relative weight is taken to be 1 (the units will be $ million per day if any reduction in development time translated into the same amount of reduction in the vehicle development day; however, some other component may become the critical path component, in which case we could use a more conservative estimate such as $ million per week)

The Formulation:

Minimize $\lambda = t_n + d_n$

subject to:

$$n = n_l + n_s \quad (12)$$

$$n_l \leq 2 \quad (13)$$

$$t_i \geq t_{i-1} + d_{i-1} \qquad i = 1, 2, \ldots, n \quad (14)$$

$$t_0 \geq t_{As} = 10 \quad (15)$$

$$10 < t_i < t_F = t_{Af} = 18 \qquad i = 1, 2, \ldots, n-1 \quad (16)$$

$$t_n \geq t_F = t_{Af} = 18 \quad (17)$$

$$\Delta x(t_{i-1}, t_i) = \beta^* (\varepsilon_i - \varepsilon_{i-1}) \quad (18)$$

$$d_i = \mathcal{F}(\Delta x(t_{i-1}, t_i)) \qquad i = 1, 2, \ldots, n \quad (19)$$

---

[14]The primary reason for using plaster is that it is a low cost medium to make changes.

$n_S, n_I = \{0, 1, 2, ...\}$ (20)

The lower bound on lead time, in (11), is given by $\lambda \geq t_{Af} + d_{min}$. Using the values for the door handle process, we have $\lambda \geq 18\ 1/7$. (From the influence function, we see that d_{min} equals 1 day or 1/7th week). For n = 0, the process is sequential with lead time = $t_{Af} + d_0 = 22$ weeks.

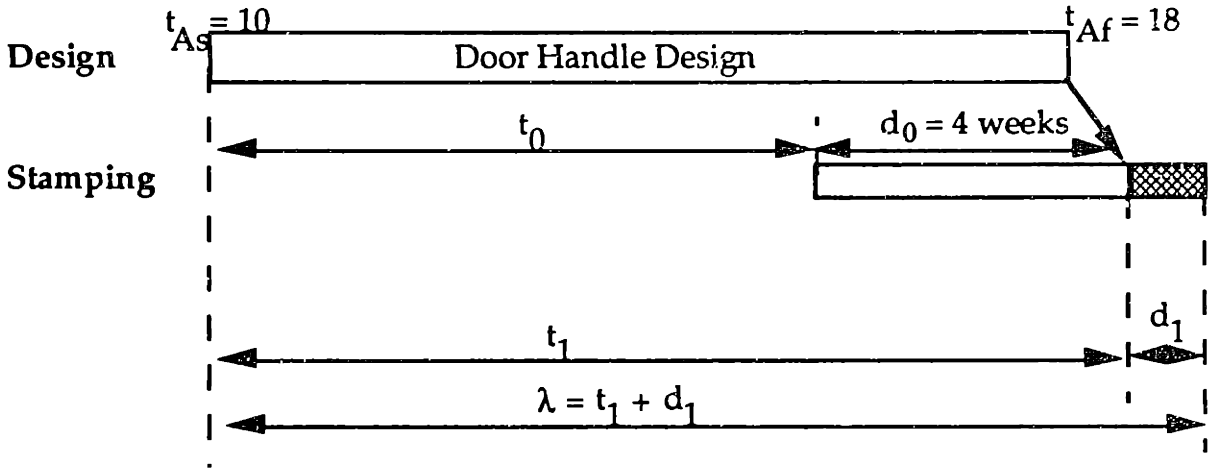


Figure 5.6: Door Design Process: n = 1 case

For n = 1, the problem becomes (see Figure 5.6):

Minimize $t_1 + d_1$

subject to:

$t_0 \geq 10$ (15)

$t_1 \geq 18$ (17)

$t_1 \geq t_0 + d_0$ (21)

$\Delta x_{est}(t_0, t_1) = 5^* (\varepsilon_1 - \varepsilon_0) = 5^*(1 - \varepsilon_0)$ (22)

$d_1 = \mathcal{F}(5^* (1 - \varepsilon_0))$ (23)

Note that the lead time (objective) grows montonically with t_1, so one of the constraints (21) or (17) should be satisfied as an equality at the optimum. We will do a branch and bound search. Suppose the active constraint is (21). Then $t_1 = t_0 + d_0$ and using this and the value of d_0 (= 4), we have $t_0 \geq 14$ (substituting in (17)) and the problem becomes as follows:

Minimize $t_0 + 4 + d_1$

subject to:

$\Delta_{est}(t_0, t_1) = 5^* (\varepsilon_1 - \varepsilon_0) = 5^*(1 - \varepsilon_0)$ (22)

$d_1 = F(5^* (1 - \varepsilon_0))$ (23)

$t_0 \geq 14$ (24)

Constraint (24) requires that the planned iteration not start before the 14th week. If we start the planned iteration beyond the 14th week, the handle depth is evolved sufficiently enough that further changes are likely to be less than 2 mm and only a small iteration would result. This is seen as follows. From the evolution function (Figure 5.5), for $t_0 \geq 14$ weeks, $\varepsilon_0 \geq 0.75$ and the estimated change in the handle depth $\Delta x_{est}(t_0, t_1) = 5^*(1-\varepsilon_0) \leq 1.25$ mm. With the estimated change less than 1.25 mm, the duration of the subsequent iteration d_1 is given by the influence function (Figure 5.2) to be 1 day.

Any iteration created by starting the planned iteration after the 14th week will be 1 day long. So, the optimal strategy is to start the planned iteration as early as possible, the 14th week. The optimal lead time is therefore 18 1/7 week for $t_0 = 14$; $d_1 = 1/7$.

We need not pursue the other branch (where constraint (17) is active), because the optimal solution we have is equal to the lower bound, so no better solution (with fewer large iterations) may be obtained. In this case, we were fortunate to reduce the branches using the simple lower bound (11).

Discussion and Sensitivity Analysis

Iterative overlapping of panel design and die development helps save 4 weeks in the development time. However, the above formulation used several inputs and made many modeling assumptions which are summarized here. A sensitivity analysis of the solution is done with respect to the changes in the inputs. The inputs used for the above model were:

- Evolution function which measures how the degree of evolution of the handle depth improves as panel design progresses.
- Influence function which includes the "model development safety factor" of 2 mm below which changes required mere drawing updates and above which changes needed a visit to the die shop (taking 3 weeks).
- Change coefficient, β, which depicts the initial range on the handle depth or the unpredictability of the new handle concept.

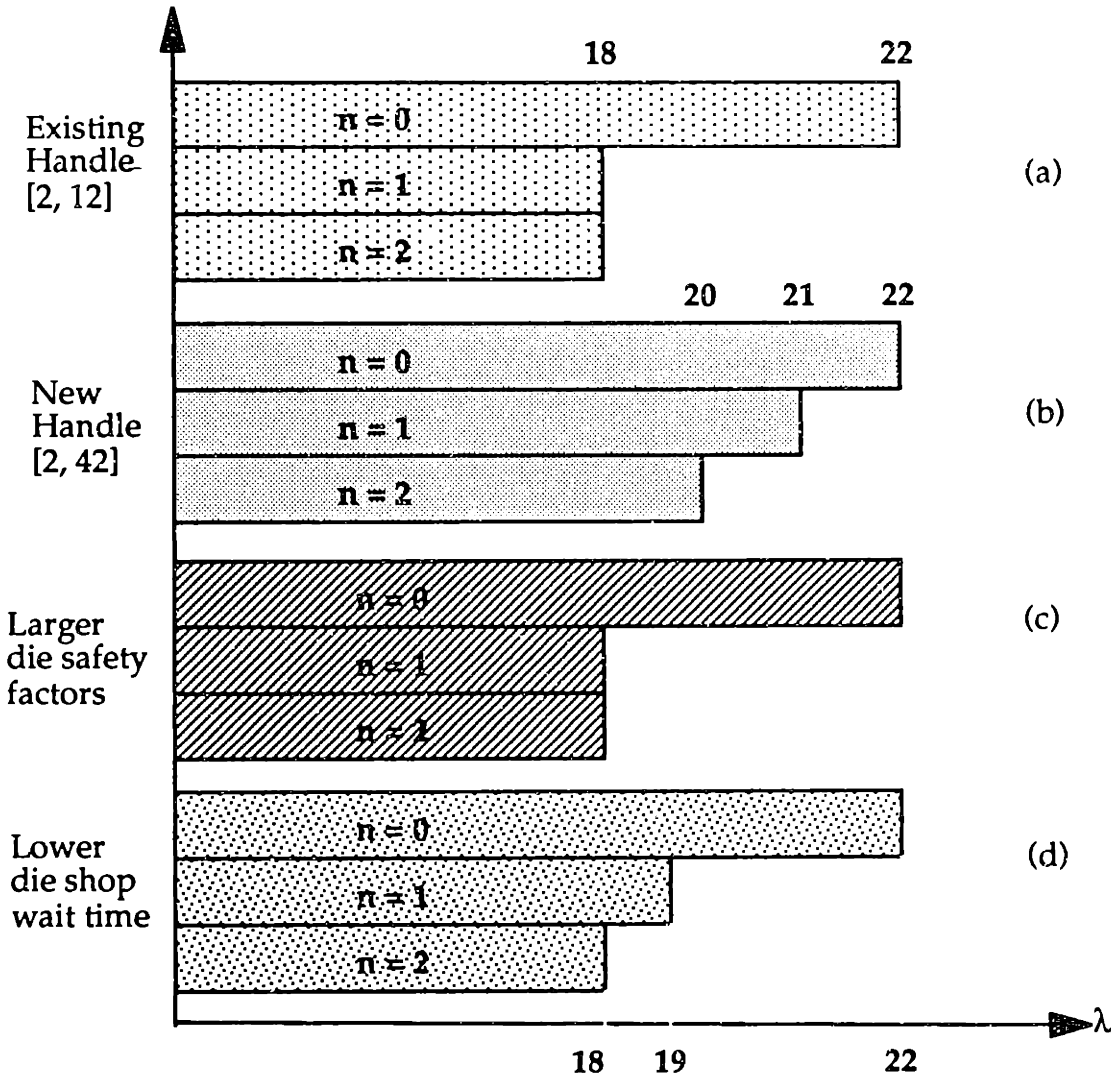


Figure 5.7: Sensitivity of the Overlapping Policy to Changes in Inputs

It may be of interest to the designer to see what effect these input parameters have on the lead time advantage gained by overlapping activities. What if the handle used is so radical that the initial depth is not known as closely as in the above example, but in a wider interval say, {2, 42} mm (β = {2, 42}). The resulting policy for the new value of β is shown in Figure 5.7 (b). In this case, two subsequent iterations (one large and one small) are needed to reach the optimal lead time of 20 weeks. As the product gets more unpredictable, the lead time advantage to be gained from overlapping gets smaller.

What can the product development organization do to accelerate such radically new processes? In Figure 5.7 (c) we see that an improvement in the die development safety factor (to 5 mm) can help reduce the new door handle optimal lead time to 18 weeks; similarly as shown in Figure 5.7 (d), a decrease in the timing of large iterations to one week (by reducing the die shop wait time) can help reduce optimal development time to 18 weeks[15]. Improving the die development safety factor is a more effective strategy, as it fetches the optimal lead time with fewer iterations, but is probably a harder capability to acquire in practice. (It must require substantial research in the area of metal forming; one work in this direction is the computer program developed by Toyota to evaluate press forming severity. It can help detect infeasibilities more easily.)

5.10 Instrument Panel Analysis

In this section, I consider the overlapping of the design and mockup development phases of a base panel. In Chapter 1, it was noted that the base panel (in Figure 1.9) is the critical component in the entire instrument panel development as it houses almost all other instrument panel components. As discussed in Appendix A3, designing the panel takes ten weeks, and building the mockups takes fifteen weeks (which involves making a wooden mold from which the fiberglass parts are made). Currently, the two stages are phased requiring a development time of 25 weeks. In the interest of reducing the development time by overlapping the two phases, I analyze the evolution of the base panel information and downstream sensitivity to the changes in the base panel design information.

The evolution of the base panel is described in detail in Appendix A3. It suffices to say here that the different parts of the base panel evolve at different rates: the driver (left) section of the base panel evolves much slower compared to the passenger (right) section and the center section because of the changes occurring in the steering column, instrument cluster which occur in the driver section. The rate of evolution is shown in Figure 5.8, where degree

---

[15]Clark and Fujimoto have observed that one of the main factors differentiating Japanese automakers from American automakers is the reduced wait time in the die shop.

of evolution of a section at a given point in time refers to the fraction of the design of that section completed at that point in time.

Interviews with mockup builders at our study company suggests that the sensitivity of mockup construction to changes in base panel is high, and iterations of mockup construction are long; so iterations subsequent to the planned iteration of mockup construction are to be avoided. Because of the different evolution patterns of the different sections of the base panel, the base panel is disaggregated into the right, center and left sections (the design and mockup construction activities will also get disaggregated into the design and mockup of each of the sections). Since only one iteration of mockup construction is done, the only decision variable in this problem is the time of finalization of each part of the base panel. As in the door handle, we will ignore the cost of development effort, because of the usage of platform teams. Because the number of subsequent iterations is zero, the cost of iterations need not be included. So the only two items in the objective are quality loss and development time. Because there are only two criteria, I will plot the trade-off between these performance parameters rather than combine them using weights into a single objective function.

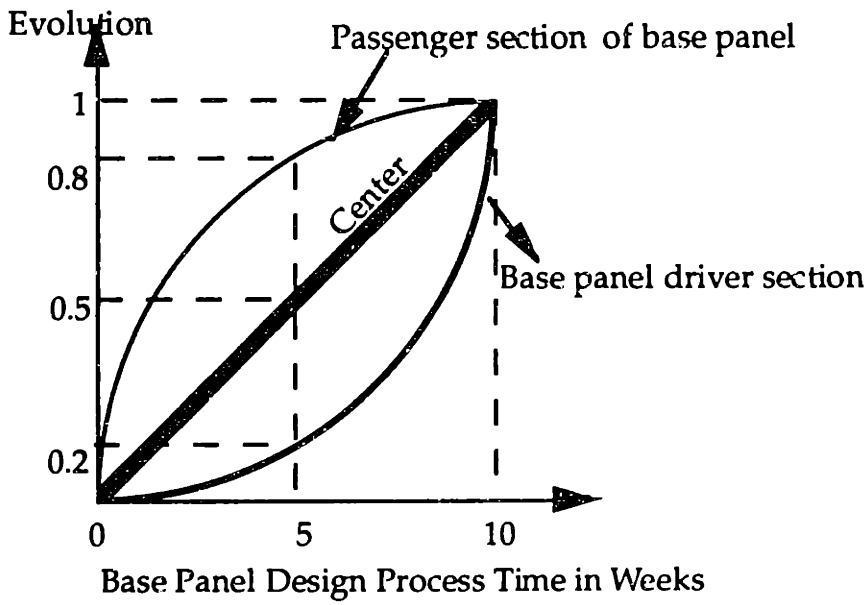


Figure 5.8: Evolution of the Parts of the Base Panel

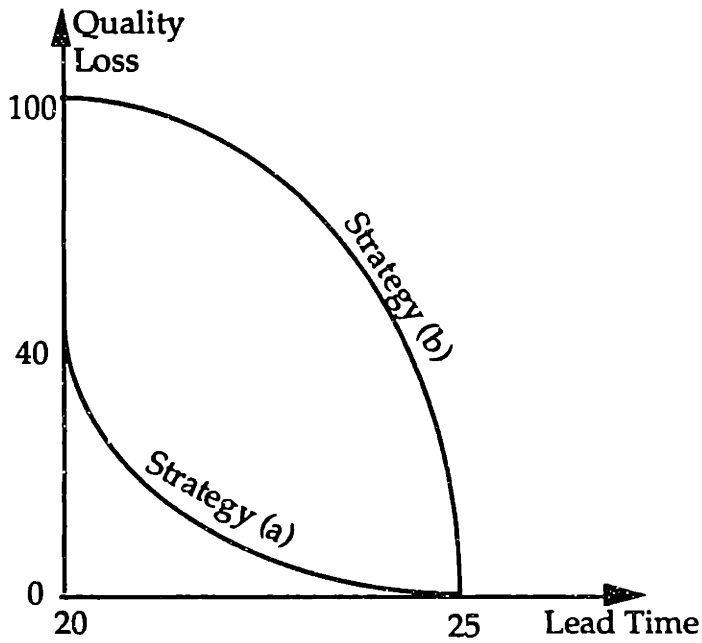
The quality loss of each section of the base panel, for freezing it at an earlier point in time t_F is, for the sake of this example, defined as the percentage of the design problem of that section left unresolved at t_F. Mathematically, the quality loss equals $(100*(1-\varepsilon_F))$. Thus freezing the right base panel at the 7th week (when $\varepsilon_F = 0.9$ approx.), would result in a quality loss of 10 (no dimensions). Freezing the left base panel at the 7th week (when $\varepsilon_F = 0.2$ approx.), would involve a quality loss of 80.

Constructing the wooden mockup of each section of the base panel takes three weeks; although the mockups of the different sections can, to a certain extent, be constructed in parallel, doing so would require a much higher rate of expenditure of resources than doing in a sequence, and would fall under what is commonly called project crashing. Under overlapping we seek a solution that would not involve crashing, so the sequential construction of the three sections is considered. (this also helps tailor sections constructed later to previously constructed sections; after all, the sections need to fit together). This results in interesting quality-time tradeoffs as shown below. Table 5.1 summarizes four different possibilities, assuming that the quality loss of different sections (right, center, left) are additive.

No.	Strategy	QL	λ
1	Current Unoverlapped process Freeze the whole base panel at 10th week	0	25
2	Freeze passenger base panel section at 7th week Freeze center and driver sections at 10th week	10	22
3	Freeze driver base panel section at 5th week Freeze center at 8th week and left at 10th week	40	20
4	Freeze passenger base panel section at 5th week Freeze center at 8th week and driver at 10th	100	20

Table 5.1: Quality Loss and Lead Time for Different Overlapping Scenarios

Rows 3 and 4 show that the same lead time can be obtained by two different strategies that result in very different quality losses. In this case, freezing the driver section of the base panel at the 5th week is a bad idea, when lot of changes are likely to happen in adjacent components such as the cluster and the steering column.



(a) : Freeze passenger section first, freeze center section three weeks later and after three weeks (or 10th week whichever is earlier) freeze driver section.
(b) : Freeze driver section first, freeze center section three weeks later, and freeze passenger section 3 weeks later (or 10th week whichever is earlier).

Figure 5.9: Quality and Time Tradeoffs for the Base Panel

The quality-time trade-off curves for two different strategies are presented in Figure 5.9. In strategy (a) the passenger section is frozen at time t_F and after 3 weeks the center section is frozen (at time t_F+3, when the passenger section mockup construction is complete), and 3 weeks later (t_F + 6) when the mockup of the center section is completed, the driver section is frozen. Each point in the curve represents a particular value of t_F. In strategy (b) the order of freeze is reversed to driver, center and passenger section. For any given lead time, strategy (a) results in a lower or equal quality loss as strategy (b); similarly, for a given quality loss, the development time resulting from strategy (a) is never greater than that from strategy (b). It is easily seen that any solution that freezes the driver section of the base panel before the passenger section is inferior to one in which the passenger section is frozen before the driver section. (Dominance of one solution over the other). The designer/manager can use the tradeoff curves to make the decision on when to freeze the different sections. *For a reasonable QL of 10, in (a), we get a lead time of 22 weeks. (a 3 week reduction in lead time)*

5.11 Vector Information Exchange

In the beginning of this chapter when the mathematical model was developed, the information exchanged was referred to as x, without distinguishing between scalar and vector information. Such a reference is valid for scalar information, as well as for vector information all of whose components possess the same evolution and sensitivity properties. However, when the components evolve differently or when the sensitivity of the downstream activity to the different components is different, it is useful to consider the components separately, and even disaggregate the components until the components themselves cannot be disaggregated into pieces with different evolution and sensitivities. This clarifies the interface between the activities and, helps identify new opportunities to overlap (as was the case with the instrument panel). However, the disaggregated components might be coupled to each other.

Consider the disaggregation of design information x into components $\{x_1, x_2, ..., x_i, ..., x_n\}$. Corresponding to this disaggregation, we also attempt to divide the upstream activity A into $\{A_1, A_2, ..., A_i, ..., A_n\}$, and downstream activity B into $\{B_1, B_2, ..., B_i, ..., B_n\}$, such that A_i is the upstream component generating x_i, and B_i is the downstream component that receives and uses x_i. In this section, I discuss the three types of situations likely to occur in product development.

- The upstream components $\{A_1, A_2, ..., A_i, ..., A_n\}$ are completely decoupled from each other and so are the downstream components. In this case, the overall overlapping problem gets decomposed into n separate overlapping problems of overlapping each A_i and B_i.

- The individual components are technically decoupled, but coupled due to resource constraints. As an example of this situation, we have the base panel case where the three sections of the base panel, albeit technically independent, share resources and so it is necessary to do them in a particular order. Due to the different evolution and sensitivity of the components, some orders may be superior to others as was the case with the base panel. In other words, the order in which the components are addressed results in interesting performance trade-offs.

- In the third case, the individual components within upstream and downstream are technically coupled. An example of this situation is the module design problem at Digital studied by Smith [41]. In this case, product design is the upstream activity, and process design is the downstream activity. Iterations occur within product design and process design phases themselves. Overlapping the product design and process design functions requires the overlapping of iterative "blocks". In this thesis, I have primarily focused on the overlapping of nominally sequential activities. As observed in the next chapter, overlapping coupled blocks will be a topic of future research interest.

5.12 Modeling Multiple Product Development Activities

Although the model developed in this chapter considered two development activities, it does not make any assumptions that restrict its applicability to just two development activities. More activities will contribute to more interface constraints, and the model will become more difficult to solve by hand (which is possible with two activities). However, with the increasing availability of more powerful computer applications to solve mathematical programs, it should be possible to solve larger collection of activities.

Summary

In this chapter, I formulated the performance trade-offs that arise from overlapping activities. I separated the overlapping constraints into generic constraints (that are independent of the design problem), and specific constraints such as, the model of design change and iterations. The model developed in this chapter unifies the different types of overlapping discussed in the previous chapter, and proves useful to improve design processes from the industry where detailed evolution and sensitivity data are available.

6. Epilogue

Conclusions, Implications for Practice, and Future Work

'Begin at the beginning,' the King said gravely, 'and go on till you come to the end; then stop.'

– "Alice's Adventures in Wonderland" by Lewis Carroll

In this dissertation, I attempted to answer the following question: how can processes that are nominally sequential be beneficially resequenced or overlapped? The relevance of the question stems from the fact that many processes in practice are nominally sequential, and resequencing them or overlapping them helps improve their effectiveness – in terms of lead time (calendar time), product quality, and engineering effort. I viewed this question with a process lens – focusing on the interdependencies and information exchanges among design activities.

The primary idea behind the sequencing problem studied in Chapter 3 is what I call *sequence constraints* – design decisions made earlier in the sequence constrain downstream decision makers by reducing their degrees of freedom; different sequences differ in the quality penalty incurred by the downstream decision makers due to the upstream decisions already made. I introduced the notion of quality loss to quantify the loss of design freedom in a sequence by the downstream decision makers, and the optimal sequence was defined as one with the lowest quality loss. To illustrate the identification of the optimal sequence (so that the process may be resequenced to the optimal sequence), I considered design problems where each activity can be construed as a nonlinear program. Several concepts were developed, such as sequence invariance, task invariance, and exclusive groups in showing that the optimal sequence in specially structured problems is equivalent to the shortest path of a graph constructed with quality losses as weights.

Unlike the sequencing problem, which considers activities that are merely coupled, overlapping deals with design situations that are nominally sequential due to the precedence relationships among activities (manifested

as information exchanges). The conventional project management approach to accelerating such processes is called project crashing – in which resources are expended at a larger rate to shorten the individual activities. Overlapping however, is concerned with shortening the development process not by shortening the individual activities, but by increasing the overlap period among them. Previous research work related to overlapping has focused on the importance of organization design, right attitudes, and team work. This thesis complements existing work by unearthing the role of the interaction among development activities in overlapping them in a profitable manner.

The classification of the exchanged information on two key dimensions, its rate of evolution, and sensitivity of downstream activities to changes in the exchanged information helped determine *how* to overlap the activities. It was argued that information whose evolution is fast may be finalized early while freezing slow evolving information can entail huge quality penalties. Similarly information to which downstream activity is not very sensitive can be used in its preliminary form while changes in high sensitivity information can cause time and cost-intensive rework. Finally, it is the combination of these two dimensions that determines how the activities are overlapped. The conceptual framework articulates the relationship between the evolution, sensitivity values, and the types of overlapping among activities.

It was also seen that different types of overlapping result in different performance trade-offs. Modeling the trade-offs helps unify the different types of overlapping, and show that the different types of overlapping differ only as a matter of degree (in evolution, sensitivity values and performance trade-offs) and not as a matter of kind. The mathematical model developed in Chapter 5 helps decide how the activities should be overlapped (when to finalize upstream information, when to start downstream iterations, and how many iterations to do). It also helps understand the relationships between the type of overlapping and the performance trade-offs.

This thesis makes two types of contributions: (i) a rigorous modeling contribution and, (ii) application to practical problems such as door development, pager development and instrument panel development. The pager development example was descriptive: it showed how an existing overlapped process can be described in terms of the overlapping framework.

The door and instrument panel development examples were prescriptive in that they helped determine how a currently nominally sequential process should be overlapped. It is noteworthy that the improvements suggested by the model have not yet been realized in practice. Because I used estimates of several timings, and input parameters, I performed a sensitivity analysis in Chapter 5 to see how the performance improvements are likely to vary if the input parameters are different. (It was seen for the door handle that when the product becomes more unpredictable, the lead time advantage reduces – but can be compensated for by an increase in the number of iterations, and an increase in upstream and downstream capabilities such as accelerated evolution and reduced sensitivity.)

Implications for Practice

A quote from Whitney [33], reproduced in Chapter 4, stated that the difficulties that firms experience in overlapping tasks involve "wasting time in extra design iterations or creating grounds for product liability if incorrect assumptions are not eliminated before the design is released". The reader will note that the framework and models developed in this dissertation can aid in addressing these difficulties:

- Consideration of the sensitivity of the downstream activity to the changes in the exchanged information can help prevent "wasting time in extra design iterations" (especially costly ones).
- Information whose early freeze results in such adverse impacts as product liability is likely to fall under slow evolution. The framework presented earlier can help ensure that such information is not finalized at a preliminary stage (without "eliminating incorrect assumptions").

Whitney added that "Companies want ways to categorize this information according to how important it is, when it is needed, and how the impact varies" based on the amount of change. I believe the notion of evolution and sensitivity serve this purpose and can be translated into the following less esoteric questions to be posed in practice (once the upstream and downstream activities, and the exchanged information are identified):

- How change-prone is the exchanged information before the upstream activity is complete? What causes the change? Can this change be forfeited

or is the exchanged information value very critical (linked to factors such as safety, recurring cost etc.) ?

- Are parts of the exchanged information less change-prone than the other parts?
- Can the inputs needed to finalize the exchanged information (such as supplier or purchasing input) be obtained early?
- Can the downstream activity be started without the final value of the exchanged information? If yes, how much work is nominally required to incorporate changes in the exchanged information? Is this work a function of (the value of) the exchanged information?
- Can the downstream activity be modified to insulate against changes in the information received from the upstream activity (such as leaving extra metal or placing components well within a box of a given size)?

The above questions have proved handy in identifying improvements in door development, instrument panel development and the pager development process by identifying the key parameters such as the door handle depth, pager size etc. Note however that in all the above examples, there exists a certain lack of "symmetry" between the upstream and the downstream activities: while upstream changes cause downstream iterations, downstream does not cause any upstream iterations. This is because the process was previously nominally sequential, so downstream activity does not contradict the upstream choices. However, there are several situations in which downstream may detect infeasibilities in the upstream output requiring subsequent upstream iterations. Apart from the simultaneous action, this may be another important benefit to accrue from overlapping and is one of the items in the future research agenda presented in the next section.

Future Work

There are several interesting extensions possible to the work presented in this dissertation. They are listed in the increasing degree of departure from this dissertation.

Overlapping Iterative Activities: In overlapping nominally sequential activities, the upstream evolution is independent of the progress of the

downstream activity. When iterative blocks are considered however, then downstream activity may start affecting the evolution of the upstream information. Two types of iterations are likely to arise: voluntary (avoidable) iterations which occur because the upstream information is used by the downstream activity before it is finalized, and (ii) involuntary or unavoidable iterations that are required because the product physics coupled the activities in a bi-directional fashion. It will be interesting to study the relationship between these two iterations: if the voluntary iterations happen to be short ones, then they can help detect some infeasibilities, and reduce some long involuntary iterations. It is also conceivable that the voluntary and involuntary iterations reinforce each other resulting in extensive, time-consuming iterations.

More Sophisticated Models of Evolution and Sensitivity: The models of evolution and sensitivity used in this thesis had certain limitations. The evolution model considered two situations: (i) the design information can be modeled as an interval variable evolving in a monotonic fashion (extensions to nonmonotonic evolution are in Appendix A4 – but they are somewhat unintuitive), (ii) the upstream activity can be modeled as changing the exchanged information from an initial value to a final value. There are several design situations that do not fit the above descriptions. Other models of evolution need to be developed that are more appropriate to those situations. This should also include random effects in the evolution (such as the CEO walking in and demanding that the tail light be changed).

Although I noted that the influence function – used to model downstream sensitivity – could be state dependent, I did not use or examine the properties of the state dependent influence functions in this work. This needs to be done in future too.

Upstream and Downstream Activities are Blocks: In the module design example at Digital studied by Smith [41], it was found that the upstream activity (product design), and the downstream activity (process design) are themselves blocks (totally coupled). The DSM from Intel reproduced from [14] in Chapter 2 had similar properties too. It will be worthwhile to examine how such blocks may be overlapped. One hypothesis is that the eigen structure of

the blocks can be used to model the evolution (or rate of convergence) of the upstream generated information.

A Network of Activities: In this work, the focus was predominantly on overlapping an adjacent pair of activities. It is indeed true, as I observed in Chapter 5, that the mathematical model can be generalized to a network of activities. It is not clear, however, if the framework with two dimensions, upstream evolution and downstream sensitivity, would be sufficient or it needs to be expanded with more dimensions.

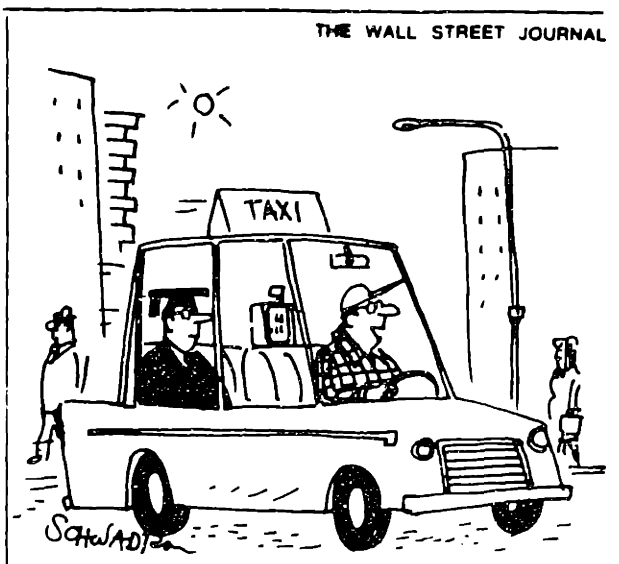
New Product Development Processes: All the example products considered in this thesis are evolutionary products (with small variations in concept). Although the evolution and sensitivity information were readily estimated for such products, it is not clear if such information can be obtained for products with new concepts. If not, the idea of real time overlapping suggested below may be tried.

Real time Overlapping: The approach to the overlapping problem in this thesis involved a priori estimation of the evolution and sensitivity data and developing a plan to overlap activities. For new products and certain old products, such information may be hard to estimate ahead of the process – but may be available while the process is being conducted (for instance, a designer may be stopped and asked, "how closely is the value of variable x known now?") Such real time data may be used to decide if the downstream iteration should start now or wait for some more time.

Including Resource Constraints: In Chapter 2, I noted that the effect of resource sharing among multiple projects is not considered in this thesis. Although this may not be restrictive for firms using dedicated platform teams, there are, in practice, many firms which do not dedicate resources to specific projects. It needs to be examined how resource sharing may be modeled and included with the formalism developed in this thesis.

In conclusion, the lessons learned from this thesis are several. Improving the process of development of a product requires a deep understanding of the structure and strength of the underlying interactions. Both task and parametric level representations of the process are needed to clarify the structure of the process, and the availability and impact of

exchanged information. With proper representation and understanding, processes can be readily resequenced and overlapped. The conceptual framework and models presented in Chapter 5 should help firms map and improve their process. These improvements can enable the transformation of defective processes into effective ones, and thereby enable firms to develop better products faster. Future work should focus on extending this work in several of the above mentioned directions, and to other applications.



"Frightened by the real world? Hey, I understand. I felt the same way when I got my doctorate."

Bibliography

1. Woolf, H.B., ed. *Webster's New Collegiate Dictionary.* 1979, G. & C. Merriam Company: Springfield, MA.

2. Humphrey, W.S., *Managing the Software Process.* Addison -Wesley Publishing Company, 1989.

3. Harrington, H.J., *Business Process Improvement.* McGraw–Hill, Inc., New York, 1991.

4. Cusumano, M.A., *Factory Concepts and Practices in Software Development.* Annals of the History of Computing. Vol. 13, No. 1, 1990.

5. McGrath, M.E., M.T. Anthony, and A.R. Shapiro, *Product Development: Success Through Product and Cycle-Time Excellence.* Butterworth-Heinemann Publishers, Boston, 1992.

6. Whitney, D.E., *Manufacturing By Design.* Harvard Business Review. pp. 83-91, Vol. July–August, 1988.

7. Clark, K.B., B. Chew, and T. Fujimoto, *Product Development in the World Auto Industry: Strategy, Organization and Performance.* Brookings Papers on Economic Activity. pp. 729-771, Vol. 3, 1987.

8. Millson, M.R., S.P. Raj, and D. Wilemon, *A Survey of Major Approaches for Accelerating New Product Development.* Jl. of Product Innovations Management. pp. 53-69, Vol. 9, No. 1, 1992.

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

10. Albano, R.E. and J.P. Keska. *Is Design Realization a Process? A Case Study.* in *The Seventh International IEEE Electronic Manufacturing Technology Symposium.* 1989.

11. Whitney, D.E. *University Research and Industrial Practice in Electro-Mechanical Design in Europe- Results of a Six Month on-site study,* Techical Report, C. S. Draper Laboratory, CSDL-P-3226, 1992.

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

13. Eppinger, S.D., *et al. Organizing the Tasks in Complex Design Projects.* in *ASME Design Theory and Methodology Conference.* Chicago, 1990.

14. Osborne, S. *Product Development Cycle Time Characterization Through Modeling of Process Iteration*, Masters Thesis, Massachusetts Institute of Technology, 1993.

15. Black, T.A., C.H. Fine, and E.M. Sachs. *A Method for Systems Design Using Precedence Relationships: An Application to Automotive Brake Systems*, MIT Sloan School of Management, Working Paper # 3208-90-MS, 1990.

16. Gebala, D.A. and S.D. Eppinger. *Methods for Analyzing Design Procedures*, Working Paper, MIT Sloan School of Management, WP # 3280-91, 1991.

17. Steward, D.V., *Systems Analysis and Management: Structure, Strategy, and Design.* Petrocelli Books, New York, 1981.

18. Colburn, E. *Abstracting Design Relations*, Masters Thesis, Carnegie Mellon University, 1988.

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

20. Imai, K., I. Nonaka, and H. Takeuchi, *Managing the New Product Development Process: How the Japanese Companies Learn and Unlearn*, in *The Uneasy Alliance*, K.B. Clark, R.H. Hayes, andC. Lorenz, Editor(s). Harvard Business School Press, Boston, 1985.

21. Blackburn, J.D., *New Product Development: The New Time Wars*, in *Time-Based Competition*, J.D. Blackburn, Editor(s). Business One Irwin Publishers, Homewood Illinois, 1991.

22. Smith, P.G. and D.G. Reinertsen, *Developing Products in Half the Time.* Van Nastrand Reinhold Publishers, New York, 1991.

23. Londono, F., K.J. Cleetus, and Y.V. Reddy. *A Blackboard Problem Solving Model to Support Product Development.* in *Proceedings of the Second National Symposium on Concurrent Engineering.* West Virginia University, Morgantown, 1990.

24. Finger, S., *et al., Concurrent Design.* Applied Artifical Intelligence. pp. 257-283, Vol. 6, 1992.

25. Kannapan, S. and K. Marshek, *A Schema for Negotiation between Intelligent Design Agents in Concurrent Engineering*, in *Intelligent CAD*, D.C. Brown, Editor(s). Elsevier Publishers, North Holland, 1992.

26. Clausing, D. *Concurrent Engineering.* in *ASME Winter Annual Meeting.* San Francisco, 1989.

27. Adler, P., *et al. From Project to Process Management in Engineering: An Emprically-based Framework for the Analysis of Product Development,* Working Paper, MIT Sloan School of Management, WP # 3503-92, 1992.

28. Watkins, M.D. and K.B. Clark. *Strategies for Managing a Project Portfolio,* Working Paper, Harvard Business School, WP # 93-004, 1992.

29. Nobeoka, K. and M.A. Cusumano. *Multi-Project Strategy and Organizational Coordination in Product Development,* Working Paper, MIT Sloan School of Management, WP # 78-92, 1992.

30. Yoshikawa, H., *General Design Theory and a CAD System,* in *Man-Machine Communications,* T. Sata and E. Warren, Editor(s). North-Holland, 1981.

31. Tomiyama, T. and H. Yoshikawa, *Extended General Design Theory,* in *Design Theory for CAD,* Elsevier Science, 1987.

32. Bell, D. and D. Taylor. *Mathematical Foundations of Engineering Design Processes.* in *ASME Design Theory and Methodology Conference.* Miami, FL, 1991.

33. Whitney, D.E. *State of the Art in Japanese CAD Methodologies For Mechanical Products – Industrial Practice and University Research,* Techical Report, C. S. Draper Laboratories, CSDL-P-3126, 1991.

34. Warfield, J.N., *Binary Matrices in System Modeling.* IEEE Transactions on Systems, Man, and Cybernetics. pp. 441-449, Vol. 3, 1973.

35. Malone, D.W., *An Introduction to the Application of Interpretative Structural Modeling.* Proc. IEEE. pp. 397-404, Vol. 63, Number 3, March, 1975.

36. Steward, D.V., *The Design Structure System: A Method for Managing the Design of Complex Systems.* IEEE Transactions on Engineering Management. pp. 71-74, Vol. EM-28, August, 1981.

37. Steward, D.V., *On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations.* SIAM Review. pp. 321-343, Vol. 4, No. 4, 1962.

38. Steward, D.V., *Partitioning and Tearing systems of equations.* J. SIAM Numerical Analysis. pp. 345-365, Vol. 2, No. 2, 1965.

39. Eppinger, S., *Model Based Approaches to Managing Concurrent Engineering.* Journal of Engineering Design. pp. 283-290, Vol. 2, 1991.

40. Sequeira, M. *Use of the Design Structure Matrix in the Improvement of an Automobile Development Process,* Masters Thesis, Massachusetts Institute of Technology, 1991.

41. Smith, R.P. *Development and Verification of Engineering Design Iteration Models*, Doctoral thesis, Massachusetts Institute of Technology, 1992.

42. Rogers, J.L. *DeMAID: A Design Manager's Aide for Intelligent Decomposition User's Guide*, Technical Memorandum, NASA, Report # 101575, 1989.

43. Sobieszczanski-Sobieski, J. *Multidisciplinary Optimization for Engineering Systems: Achievements and Potential*, Technical Memorandum, NASA, Report # 101566, 1989.

44. Rogan, J.E. and W.E. Cralley. *Meta-Design- An approach to the development of design methodologies*, Institute for Defense Analyses, IDA Paper P-2152, 1990.

45. Smith, R.P. and S.D. Eppinger. *A Predictive Model of Sequential Iteration in Engineering Design*, Working Paper, MIT Sloan School of Management, WP # 3160-90-MS, 1991.

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

47. Wiest, J.D. and F.K. Levy, *A Management Guide to PERT/CPM*. Second ed. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1977.

48. Simon, H.A., *The Sciences of the Artificial*. 2nd edition ed. Cambridge, MA, 1981.

49. Bellman, R. and S.E. Dreyfus, *Applied Dynamic Programming*. Princeton, 1962.

50. Mansfield, E., *et al., Research and Innovation in the Modern Corporation*. First ed. W. W. Norton & Company, New York, 1971.

51. Eastman, R.M., *Engineering Information Release Prior to Final Design Freeze*. IEEE. Transactions on Engineering Management. Vol. EM-27, No. 2, 1980.

52. Takeuchi, H. and I. Nonaka, *The new new product development game*. Harvard Business Review. pp. 137-146, Vol. January-February, 1986.

53. Clark, K. and T. Fujimoto. *Overlapping Problem Solving in Product Development*, Harvard Business School, Working Paper # 87-048, 1987.

54. Clark, K.B. 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.

55. Nihtila, J. *Simultaneous Engineering and its Applications in Project Oriented Production - A Case Study in the European Project Oriented Industry*, Licentiate in Technology, Helsinki University of Technology, 1992.

56. Ward, A.C. *A Theory of Quantitative Inference Applied to a Mechanical Design Compiler*, Doctoral Thesis, Massachusetts Institute of Technology, 1989.

Appendix A1 Mathematics of the Ordering Problem

Mathematical description of a Cross-Functional Design Task

I make the following assumptions:

• Each task T_i is a nonlinear program with a continuously differentiable objective J_i. J_i is explicitly expressible as a function of the design variables. Let Z_i be a set consisting of the design variables appearing in J_i and z_i be a vector consisting of these variables.

• Finite upper and lower limits are specified on each design variable so that the domain of design variables is a closed and bounded (compact) set. All equality constraints have been explicitly eliminated or used for symbolic propagation. The only inequalities are lower and upper limit constraints.

• Suppose that variables $z_{i1}, z_{i2}, \ldots, z_{ir}$ occur in task T_i. Without loss of generality consider the order $= \{T_1, T_2, T_3, \ldots, T_n\}$. The nonlinear programming problem for task T_i (in the order φ) and the corresponding optimality conditions are stated in Table 1 (see Table 1). I make the following observations. (' superscript indicates the vector transpose).

- All the components of the gradients of L_{ik} and U_{ik} are zero, but for kth component:

$$\nabla L_{ik} = \{0, 0, \ldots, 0, -1, 0, 0, \ldots, 0\}' ; \quad \nabla U_{ik} = \{0, 0, \ldots, 0, 1, 0, 0, \ldots, 0\}'$$

$$\text{Also } L_{ik}(z_i^*) = L_{ik}(z_{ik}^*); \quad U_{ik}(z_i^*) = U_{ik}(z_{ik}^*);$$

- If in an order T_i decides the value of design variable z_{is}, then the order constraints, S_{iq}^{φ}, do not affect z_{is}. Symbolically,

$$z_{is} \notin Z_i \cap \{Z_1 \cup \cdots \cup Z_{i-1}\}. \text{ So } \frac{\partial S_{iq}^j}{\partial z_{is}} = 0$$

Mathematical representation of T_i:	K-T Optimality Conditions for task T_i in the order φ.
T_i: Min $J_i(z_i)$ subject to $L_{ik}(z_{ik}) = l_{ik} - z_{ik} \leq 0$ $U_{ik}(z_{ik}) = -u_{ik} + z_{ik} \leq 0$ $S_{iq}^{\varphi} = z_{iq} - z_{iq}^{\varphi} = 0$ for $k = 1, 2, ..., r$ and $z_{iq} \in Z_i \cap \{\boxed{Z_1} \cup \cdots \cup \boxed{Z_{i-1}}\}$ $q = 1, 2, ..., Q$ • l_{ik} and u_{ik} are the limits on z_{ik} • z_{iq} represents the design freedom lost by T_i in the order φ. z_{iq}^{φ} is the value set by a preceding task.	$\nabla J_i(z_i^*) + \sum_{k=1}^{r} \lambda_{ik} \nabla L_{ik}(z_i^*) + \nu_{ik} \nabla U_{ik}(z_i^*) + \sum_{1}^{Q} \kappa_{iq} \nabla S_{iq}^{\varphi} = 0$ $\lambda_{ik} L_{ik}(z_i^*) = 0;$ $\nu_{ik} U_{ik}(z_i^*) = 0;$ $\lambda_{ik}, \nu_{ik} \geq 0;\ k = 1, 2, ..., r$ where $\nabla J_i = \begin{Bmatrix} \frac{\partial J_i}{\partial z_{i1}} \\ \vdots \\ \frac{\partial J_i}{\partial z_{ir}} \end{Bmatrix}$ and λ, ν and κ are the lagrange multipliers.

Table 1: Optimality Conditions

For variables such as z_{iS}, decided by T_i, we can rewrite the K-T conditions as follows:

$\nabla J_i(z_i^*) - \lambda_{iS} + \nu_{iS} = 0$

$\lambda_{iS} L_{iS}(z_{iS}^*) = 0;$

$\nu_{iS} U_{iS}(z_{iS}^*) = 0;$

$\lambda_{iS}, \nu_{iS} \geq 0;$

It is noteworthy that the K-T conditions for z_{iS} are still a function of z_i^*.

We will combine the K-T conditions for all variables belonging to the same exclusive group Y_j, decided by T_i. Suppose that exclusive group Y_j consists of variables $z_{i1}, z_{i2}, \cdots, z_{is}$.

$$\text{Let } y_j = \begin{Bmatrix} z_{i1} \\ \vdots \\ z_{is} \end{Bmatrix}; \quad \pi_{ij} = \begin{Bmatrix} \lambda_{i1} \\ \vdots \\ \lambda_{is} \end{Bmatrix}; \quad \rho_{ij} = \begin{Bmatrix} \nu_{i1} \\ \vdots \\ \nu_{is} \end{Bmatrix}; \quad \frac{\partial J_i}{\partial y_j} = \begin{Bmatrix} \frac{\partial J_i}{\partial z_{i1}} \\ \vdots \\ \frac{\partial J_i}{\partial z_{is}} \end{Bmatrix}.$$

Now we can write (K-T) conditions for the exclusive group y_j as:

$$\frac{\partial J_i}{\partial y_j}(z_i^*) - \pi_{ij} + \rho_{ij} = 0$$

$$\lambda_{ik} L_{ik}(z_{ik}^*) = 0;$$

$$\nu_{ik} U_{ik}(z_{ik}^*) = 0; \ \lambda_{ik}, \nu_{ik} \geq 0; \ k = 1, 2, \ldots, s$$

A1.2 Order Invariant exclusive groups

An exclusive group Y_j is said to be order invariant if the values of each design variable belonging to Y_j is the same under any order.

Proposition 1: An exclusive group Y_j is order invariant if C1.1 or C1.2 is satisfied.

C1.1 $$\frac{\partial}{\partial y_k}\frac{\partial J_p}{\partial y_j} = 0 \ \forall \ y_k, (j \neq k) \text{ spanning every} T_p \in t_j.$$

(i. e.Y_j is insensitive to all other spanning exclusive groups in every forming task)

and the independent decisions of all forming tasks equal the same value y_j^*.

C1.2 $\forall \ X_i \in Y_k$, either $\frac{\partial J_p}{\partial X_i} > 0 \ \forall T_p \in t_j$ or $\frac{\partial J_p}{\partial X_i} < 0 \ \forall T_p \in t_j$

Proof: Consider a order in which the value of the design variables in y_j is decided by one of the tasks $T_p \in t_j$. All orders have this property because by definition of decision in Section 3, the value of the design variables in y_j can

only be decided by one of the tasks $T_p \in t_j$. Now if C1.2 is satisfied, because of monotonicity, any design variable will be driven to the same decision value y_j^* in every order (this value equals one of its limits due to its monotonicity).

If all $\frac{\partial J_p}{\partial X_i} > 0$, then $\frac{\partial J_i}{\partial y_j} > 0$. To satisfy K-T conditions, $\pi_{ij} > 0$ and $L_{ik}(z_{ik}^*) = 0$. So in all orders, $z_{ik} = l_{ik}$. If all $\frac{\partial J_p}{\partial X_i} < 0$, then $\frac{\partial J_i}{\partial y_j} < 0$. Now $\rho_{ij} > 0$ and so, $U_{ik}(z_{ik}^*) = 0, z_{ik} = u_{ik}$.

If C1.1 is satisfied, then we first prove that the value decided by any task $T_p \in t_j$ for the design variables in y_j is always the same as the task's independent decision.

$$\text{When } \frac{\partial}{\partial y_k}\frac{\partial J_p}{\partial y_j} = 0 \;\forall\; y_k, (j \neq k) \text{ spanning every} T_p \in t_j., \; \frac{\partial J_i}{\partial y_j}(z_i^*) = \frac{\partial J_i}{\partial y_j}(y_j^*)$$

Now we can write (K-T) conditions for the exclusive group y_j as:

$$\frac{\partial J_i}{\partial y_j}(y_j^*) - \pi_{ij} + \rho_{ij} = 0$$

$$\lambda_{ik} L_{ik}(z_{ik}^*) = 0;$$

$$\nu_{ik}\; U_{ik}(z_{ik}^*) = 0;\; \lambda_{ik},\; \nu_{ik} \geq 0;\; k = 1, 2, \ldots, s$$

Now one observes that the K-T conditions are independent of other exclusive groups and independent of order! Thus the K-T conditions "separate out" for any exclusive group. So the (value decided by a task when it makes its) independent decision satisfies the K-T conditions under any order. This is because when the task makes its independent decision it has to satisfy the same K-T conditions that it does under any order. Now if the independent decisions of every forming task are the same, as the proposition states, then y_j is order invariant.

A1.3. Task Invariant Exclusive Groups

Proposition 2: Exclusive group Y_j is forming task T_i invariant if it satisfies C2.1 or C2.2

C2.1 $\frac{\partial}{\partial y_k}\frac{\partial J_i}{\partial y_j} = 0 \; \forall \; y_{k'} \; (j \neq k)$ spanning Ti and not satisfying C1.

(i. e. Y_j decision by T_i is sensitive only to other provenly order invariant exclusive groups).

C2.2 $\forall \; X_i \in Y_k$, either $\frac{\partial J_p}{\partial X_i} > 0$ or $\frac{\partial J_p}{\partial X_i} < 0$ (i. e. every Variable in y_j is monotonic in T_i in the range of the given design problem).

Proof: It is easy to show that if T_i makes Y_j decision and y_j is monotonic in T_i (C2.2) then the variables in Y_j will be driven to the same value (one of the limits; similar to the argument used in C1.2). This can be verified using K-T conditions for y_j, since there are no order constraints affecting the design variables in y_j.

If C2.1 is satisfied, then again the K-T conditions, written for Y_j become order independent, in a fashion similar to the Appendix A2 (we set all order invariant exclusive groups to their constant, order invariant decision value). Hence the value decided by T_i for the design variables belonging to Y_j is the same in every order in which T_i makes Y_j decision.This order invariant value equals the independent decision of T_i because the independent decision is the same as the decision of the order in which T_i makes decisions first.

A1.4. Partial Quality Loss

A task T_i is said to lose the exclusive group Y_k freedom in a order φ if the variables in Y_k are decided by a preceding task in the order φ.

Partial quality loss incurred by T_i due to loss of exclusive group Y_k freedom in a order φ, QL_{ik}^{φ} is defined as the quality loss incurred by T_i when it loses only the exclusive group Y_k freedom. Mathematically:

$$QL_{ik}^{\varphi} = J_i(y_1^*, y_2^*, \cdots, y_k^{\varphi}, \cdots, y_L^*) - J_i^*$$

where $y_1, y_2, \cdots, y_k, \cdots, y_L$ are assumed to be the vectors with the variables in the corresponding exclusive groups spanning T_i.

Proposition 3:

If all exclusive groups spanning a task T_i satisfy either C2.1, C1.1 or C1.2, then Quality Loss incurred by task T_i in a order φ, QL_i^{φ}, can be written as the sum of partial quality losses incurred for loss of each exclusive group freedom in order φ.

Proof: Quality loss in φ can be written as: $QL_i^{\varphi} = J_i(y_1^*, y_2^{\varphi}, \cdots, y_k^{\varphi}, \cdots, y_L^*) - J_i^*$

(assuming the degrees of freedom lost in φ are in $y_2, \cdots, y_k$)

Applying Taylor's Theorem for multiple variables (given below) to the quality loss term:

$$f(x + \Delta x, y + \Delta y) - f(x, y) = \left[\Delta x \frac{\partial f}{\partial x} + \Delta y \frac{\partial f}{\partial y}\right] + \frac{1}{2!}\left[(\Delta x)^2 \frac{\partial^2 f}{\partial x^2} + \Delta x\, \Delta y \frac{\partial^2 f}{\partial x \partial y} (\Delta y)^2 \frac{\partial^2 f}{\partial y^2}\right] + \cdots$$

$$QL_i^{\varphi} = J_i(y_1^*, y_2^{\varphi}, \cdots, y_k^{\varphi}, \cdots, y_L^*) - J_i(y_1^*, y_2^*, \cdots, y_k^*, \cdots, y_L^*)$$

$$= \left[(y_2^{\varphi} - y_2^*)\frac{\partial J_i}{\partial y_2} + \cdots + (y_k^{\varphi} - y_k^*)\frac{\partial J_i}{\partial y_k}\right] +$$

$$\frac{1}{2!}\left[(y_2^{\varphi} - y_2^*)^2 \frac{\partial^2 J_i}{\partial y_2^2} + \cdots + (y_k^{\varphi} - y_k^*)^2 \frac{\partial^2 J_i}{\partial y_k^2} + \text{cross-derivative terms}\right] + \cdots$$

When C1.1 or C1.2 is satisfied then the exclusive groups are order invariant in which case the Δx term is zero. If C2.1 is satisfied, then the cross-derivative terms are zero with respect to all groups that do not satisfy C1.1 or C1.2. Applying these in the quality loss term, we have:

$$QL_i^{\varphi} = QL_{i2}^{\varphi} + \cdots + QL_{ik}^{\varphi} \text{ where}$$

$$QL_{ik}^{\varphi} = (y_k^{\varphi} - y_k^*)\frac{\partial J_i}{\partial y_k} + \frac{1}{2!}(y_k^{\varphi} - y_k^*)^2 \frac{\partial^2 J_i}{\partial y_k^2} + \cdots$$

Thus the quality Loss incurred by task T_i in a order φ, QL_i^{φ}, is the sum of partial quality losses incurred for loss of each exclusive group freedom in order φ.

Corollary If the exclusive groups spanning every task in a product development process satisfies the conditions of Proposition 3, then the quality loss of a sequence is the weighted sum of partial quality loss of each individual tasks over the exclusive group freedom lost in the sequence. (Over the exclusive group freedoms not lost, the quality loss is zero, so adding such exclusive groups does not affect the summation).

From Proposition 3 and quality loss definition, $QL^{\varphi} = \sum_{i=1}^{n} w_i \sum_{k=1}^{M} QL_{ik}^{\varphi}$

Reversing the order of summation, $QL^{\varphi} = \sum_{i=1}^{n} w_i \sum_{k=1}^{M} QL_{ik}^{\varphi} = \sum_{k=1}^{M} \sum_{i=1}^{n} w_i \; QL_{ik}^{\varphi}$

Defn: Quality loss inflicted by a task T_h by deciding the exclusive group y_j in a sequence φ, QLI_{hj}^{φ}, is defined as the weighted sum of the partial quality losses incurred by each task which loses exclusive group y_j freedom to T_h in the sequence φ (tasks which do not lose the exclusive group freedom incur a loss of zero; hence the summation can include all tasks): $QLI_{hj}^{\varphi} = \sum_{i=1}^{n} w_i \; QL_{ij}^{\varphi}$

Defn: Quality loss inflicted by a task T_h, QLI_h^{φ}, is defined as the sum of the partial quality losses inflicted due to all the exclusive groups decided by T_h in the sequence φ.

(By definition of ordered decision making, each exclusive group decision is made by one task, which inflicts quality losses on all other tasks. So the sum of quality losses inflicted due to all the exclusive groups is the same as the sum of quality losses inflicted by all the tasks. So,

$$\sum_{k=1}^{M} QLI_{hk}^{\varphi} = \sum_{h=1}^{n} QLI_h^{\varphi}$$

From corollary above we have, $QL^{\varphi} = \sum_{k=1}^{M} \sum_{i=1}^{n} w_i \; QL_{ik}^{\varphi}$

Using $QLI_{hj}^{\varphi} = \sum_{i=1}^{n} w_i \; QL_{ij}^{\varphi}$, we have, $QL^{\varphi} = \sum_{k=1}^{M} QLI_{hk}^{\varphi} = \sum_{h=1}^{n} QLI_{h}^{\varphi}$

This shows that the quality loss of a sequence is also the sum of the quality losses inflicted by each task in the sequence.

Appendix A2 Details of Door Development

"A door is a small car itself". Doors, like automobiles, are easily visualized. Although comprised only of ten major parts (compared to the approximately two-hundred and seventy major parts in an automobile), door systems embody many aspects of the complexity inherent in the product development process of an automobile. The following list itemizes the reasons for studying doors.

- Doors are high lead time, critical path items in the study company.
- Doors are redesigned for every new automobile, unlike engines or transmissions. In fact, doors need to be redeveloped for four door, two door and convertible versions of the same concept.
- Development of doors is a highly interactive process requiring substantial interaction among stylists, body engineers, sealing engineers, electrical system engineers, process engineers, vendors to name a few.
- Doors incorporate such large lead time processes such as stamping and are active participants (initiators and recipients) in the engineering change process.
- The complexity of door development has been increased by the recent federal side-impact regulations, which require that automobiles pass the side-impact test at a speed of 35mph.

What aspects of the product development process are not represented by doors? First, the effects of innovation are not as critically felt in doors as for instance, in Power Trains or Suspensions. Second, doors do not drive the concept as the exterior body surface does and therefore door development does not play a significant role in the initial phases of the design. Thirdly, doors are not as expensive a component as the engine or transmission.

A typical door is shown in Figure 1. Table 1 gives a list of parts present in a typical door.

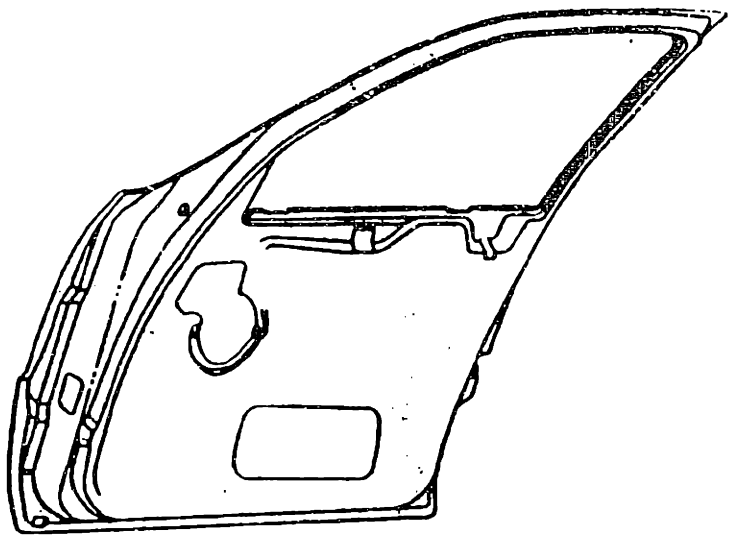


Figure 1: A Fully Stamped Automotive Door

Door Parts

Sheet Metal Parts	**Hardware Parts**
Outer Panel (O/P)	Window Glass
Inner panel (I/P)	Mirrors
Belt Reinforcements: I/P to O/P	Weather Strap
Side Impact Beam	Regulator Mechanism
Hinge Pillar	Sealing Assembly)
Hinge Reinforcement	Check Strap Assy.
I/P and O/P Reinforcements	Hinge Assembly

Table 1: (Sheet Metal and Hardware) Parts in a Typical Door

Door Terminology

Inner, Outer panels:	Stampings which form the inner and outer face of the door; usually of light gauge metal. Inner panel includes lots of formations to contain hardware parts such as the speaker, window regulator, servos (for power windows) etc.
Belt Line:	A horizontal line that separates the upper section of the door from the lower section
Belt Reinf.	Light gauge plates that run horizontally across the door attached at the belt line level.

Check Strap:	The hardware part that keeps the door open in partially open positions.
Cut lines:	Lines separating front outer panel from rear outer panels and other quarter panels.
Hinge face:	The face below the belt line where hinges attach.
Hem Flange:	A flange that sews the outer and inner panel together by a crimping or hemming action.
Impact beam:	A beam made of heavy gauge metal running in a horizontal direction; provides rigidity in side impact crashes.
P1 and P2 lines:	Lines that separate the door opening to body and go all around the door.

Steps in Door Development

Substantial amount of door development occurs at four departments: Styling, (Engineering) Design, Tooling Design, and Manufacturing. The door development process (like the overall vehicle development process) can be divided into the following phases:

Concept Development: At the beginning of this stage, high level decisions about the door are made such as number of doors (4 door, 2 door etc.), and type of door (hard top glass, fully stamped or rolled frame doors). These decisions are based on market survey, targeted customer, wind noise levels, and manufacturability considerations etc. Next, engineering designers begin establishing the door functional objectives, and stylists are involved in coming up with a clay model of several alternative concepts. Door styling especially involves studying window glass and door exterior forms. At the end of the concept development stage, the door exterior and the window glass curvature are frozen.

Theme Development: Once the vehicle concept and the window glass are frozen, the theme of the vehicle is developed (while keeping the glass curvature constant across themes); theme development involves a more detailed development of the concept considering not only the form, but also the function and fabrication issues. At this time, the critical sections of the door are designed; door swing studies are carried out to ensure that, while swinging, the door clears the fenders; manufacturing engineers review the principal locating points for tooling, and other dimensions and tolerances.

Failure modes and effects analysis is also performed. Concurrently, door engineering designers start out-sourcing their hardware parts to vendors.

Engineering Design: Engineering design of the frozen theme involves (i) packaging of the various components during which the location of the various features is decided through negotiation among engineers working on door systems, electrical systems, interior systems and other body panels, and (ii) the detailed design of the various sheet metal parts. Among the items that gets decided during this packaging include the angles of the B-pillar and C-pillar, the belt line gap, the hinge surface and the sealing surface, door handle, speaker location, door lock location etc. Detailed design of the door involves the development of a geometrically correct, and manufacturable surface that is closest to the styling intent. The engineering design process of the door panels (the most complex parts of a door) is explained in greater detail in the next section.

Soft Tooling Development: Soft tooling, made of a relatively inexpensive alloy of steel called kirksite, is used to manufacture parts for the prototype vehicle. Before the kirksite die is cut, a visual aid made of plaster, called *plaster draw die,* is developed for all drawing dies. This visual aid, explained in greater detail in a forthcoming section, helps debug the problems likely to arise in drawing. Once the plaster draw die is satisfactory, the soft tooling is fabricated, from which 300– 500 panels can be stamped out.

Prototype Vehicle Development: The parts made of soft tooling are assembled into the prototype vehicle to verify fit and function of the entire vehicle. The prototype vehicle helps debug problem in the door such as wind noise, interference with other vehicle parts etc. Changes are made both in the parts and the tooling design based on the testing of the prototype vehicle.

Hard Tooling Development: Hard tooling, made of a much more stronger and expensive alloy of steel, are used in the volume production of the vehicles. Once the hard tooling is machined, design changes are much harder to implement. The hard tooling is tested in a pilot vehicle following which the product is launched, and mass production starts with the hard tooling.

Focus

To apply the overlapping methodology I will focus my primary attention on the development of the door panels. Such a focus is justified, because these are the most complex parts in a door system. Further, to overlap, I will

consider the engineering design and the soft tooling development stages. Together these two phases could take upto 38 weeks. (18 weeks for engineering design and 20 weeks for soft die development). The two phases are explained in greater detail below (see Figure 2).

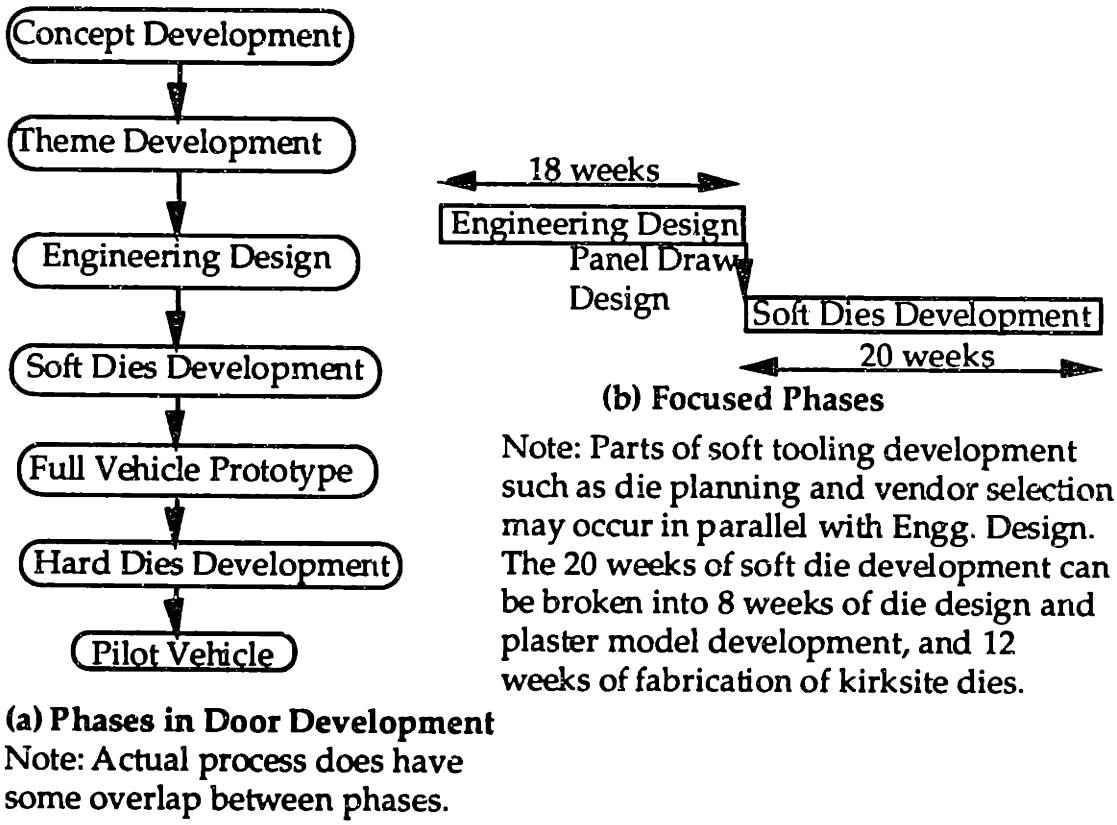


Figure 2: Phases in Door Development and My Focus

Engineering Design Process of Door Panels

The engineering design process begins with a digitized (computer data) of the door surface received from styling. Engineers work on fitting an engineering surface, free of bumps and troughs, through the digitized points. The intermediate steps involved in this process are as follows (timing data are estimates obtained by interviewing designers):

- The P1 and P2 lines (door-body interface) are defined in a week. This requires consultation with sealing, electrical and interior engineers.
- The details of the boundary and other feature lines take another week. At this time, the flanges at the periphery (called hem flanges) are also defined.

- The development of the typical sections of a door takes one week.
- Finalization of the wireframe information takes a week. The wireframe model includes outer boundaries, feature lines, flanges and wireframe sections cut in every direction.
- Resurfacing to determine the closest-fit surface to the digitized theme takes 3 weeks. Of this, one week each is spent on developing the main and trim surface, and one week on reviewing surface with stylists. Effort is taken to ensure that the surface does not deviate from "hard points and principal locating points " of the vehicle which are key landmarks frozen well in advance.
- Once the surface is reviewed, a verification model, called VERCAM, is made to assure the stylists that the engineering surface reflects the styling intent. VERCAM's are cut of a material called Renplank (above a honeycomb base) by an external vendor. Vendor selection can be finished within three weeks of the theme release. The selected vendors can block their model to the wireframe release four weeks after wireframe release. Finally, the vendors use the surface information to code and cut the model. A VERCAM can be made ready no earlier than ten weeks after theme release, seven weeks after wireframe release, and three weeks after surface release.
- The first iteration of VERCAM, cut by vendors, is reviewed by stylists and product designers. Often changes need to be made and the modified surface is submitted to the vendors for a second VERCAM. The second VERCAM is available 2 weeks after the modified surface release.
- While the VERCAM is being cut, the engineering designers work on finalizing the panel draw information which contains details of all surface formations. Currently, the panel draw information, as defined, cannot be released until 18 weeks after theme release. The intermediate steps are explained in greater detail in a later section.

"Soft" (Kirksite) Die Development

Soft die development can be divided into the following stages:

- Die planning stage when the number, and sequence of dies is determined.

- Die design stage when the die angle and the binder size and location are designed (approximately 4 weeks).
- Visual aid construction when a plaster draw die model is developed to aid in testing the feasibility of draw and the binder design (4 weeks).
- Soft die fabrication when the kirksite die is cut in a die shop (12 weeks).

It is noteworthy that the die planning is an ongoing process that starts with the theme release and is finished by the time the panel draw information is released. Die design, however, uses the panel draw information released by product designers to locate and size the binders and to decide die angles. After the design is complete, plaster models of the draw die are used to check the depth of draw in 3D. Other types of dies such as, the trim dies, flange dies, pierce dies and finish dies do not require a plaster model. The die design and the plaster model construction are explained in greater detail later in this chapter. Once the plaster model is done fabrication of the kirksite die begins.

The Critical Information Transfer Path

The above timing information is represented as a information-based network to identify the critical information transfers. There is one important way in which such a network differs from a PERT network. Unlike in the PERT chart, which is constructed based on activities, the network below is constructed based on information flows. Each node in the network below represents the time at which a particular design information is ready for transfer between functions. The labels on the edge (i, j) represents the time required to convert information i into information j. For instance, the edge between theme and panel draw information is labeled 18 weeks to show that it takes 18 weeks to convert theme information into the panel draw information.

With these timings, the critical information transfer path is obtained using calculations similar to project management shown in Figure 3 (by identifying the longest path).

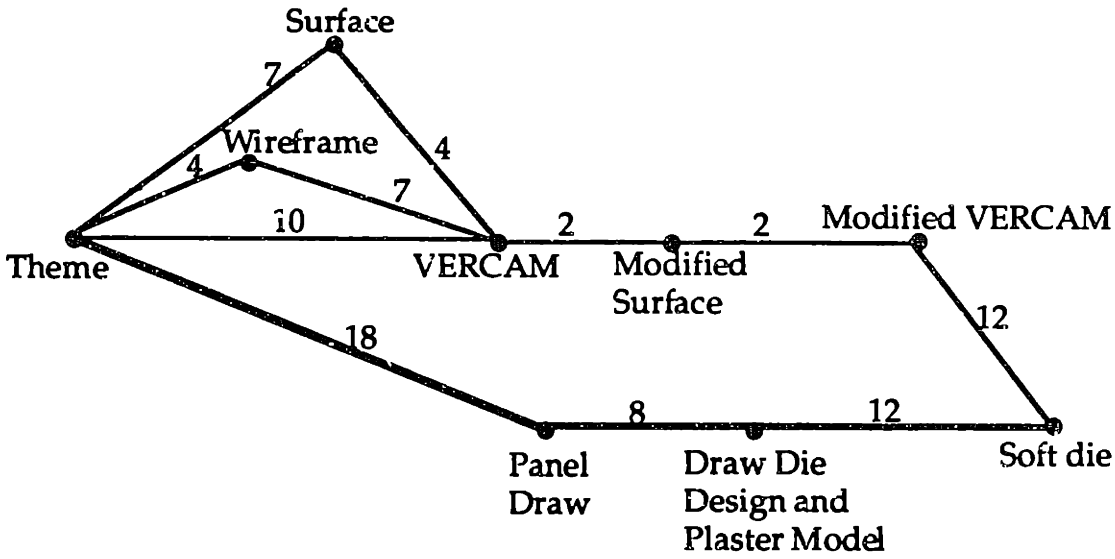


Figure 3: Network Representing Design Information Transfers

From the above, we see that the critical design information constitutes the panel draw information, and the draw design and plaster model information. In order to overlap, we need to understand these two in greater detail as shown below:

Panel Draw Information

The panel draw information is a completely surfaced and faced 3-D CATIA model. It contains all the surface formations both on the periphery and in the interior of the panels, and all holes, flanges and trim lines. The panel periphery draw information is primarily determined by the hem flange that stitches the outer and inner panels together. As described earlier, the hem flanges are worked out as early as the second week after theme release and are finalized as early as the surface release. So the periphery information is ready for exchange as early as seven weeks after theme release. The panel interior is however a different issue.

Releasing the inner panel draw information takes a long time, because the designers wait to establish the final positions of all formations by negotiating with the electrical systems group (speakers, wiring locations), and the door hardware group (regarding window regulators and connection path to the door locks). The outer panel draw information is complicated due to the delay involved in finalizing the door handle profile and depth. Both outer and inner panel interiors are not finalized until 18th week.

Using the notion of the evolution developed in earlier chapters, I note that both the outer and inner panel peripheries evolve fast, and both the outer and inner panel interiors evolve slowly. Owing to the difference in evolutions, it is useful to disaggregate the panel draw information into outer panel periphery, inner panel periphery, outer panel interior, and inner panel interior information. It is now necessary to find out what the sensitivity of the downstream activity, draw die design and plaster model development, is to the periphery and interior draw information.

Draw Die Design and Plaster Model Development

In this section, I describe the downstream activity, draw die design and plaster model development, in detail to show that the sensitivity of the downstream activity to the door panel periphery information is high and to the panel interior information is low.

In a (toggle) draw operation, a punch descends on a blank causing the (plastic) flow of metal into a die cavity to assume the shape of a seamless hollow vessel. Most body panels in a car are produced by drawing. The blank is placed on a portion of the die, called the binder surface, and is forced to conform to the shape of the punch as it is pushed into the die cavity. The edges of the blank are confined between the binder surface of the die and a blank holder. This confinement controls the metal movement and ensures that there are no wrinkles or tears in the drawn panels. (Work hardening of the metal during the drawing operation makes the panels stiff and strong – prevents "oil canning").

It is noteworthy that the panel periphery dimensions (especially the trim line location) determine the draw punch size and the binder surface location. The panel interior information, such as formations and holes, do not affect the draw die significantly. Among all dies, designing the draw dies is the most challenging task– during which the binder surface and die angles need to be decided after a careful analysis of the panel draw information released by the product engineers.

The binder surface – part of the die where the edges of the blank will be held during drawing – controls the flow of metal into the die. It must be broad enough to accommodate the blank and so situated that the blank will

not buckle. The size and location of the binder surface are important decision variables that strongly impact the strength and surface finish of the panel produced. The binder surface size is determined by the total depth of draw of the panel. The location of the binder surface should be such that the score marks created in drawing (due to portions of the metal flowing into the die) would not reach beyond the trim line – to ensure a good surface finish. It is critically impacted by the panel periphery where the trim line is located. Locating and sizing the binder surface is a real art. To aid in designing the binder surface a plaster model of the draw die is built.

The plaster draw die model is a duplicate of the surface shape of the panel with the binder surface attached. Being a three dimensional model, the plaster draw die helps the engineers visualize and debug the problems involved in drawing a panel. Plaster offers a low cost medium to test feasibility and implement changes before the fabrication of steel dies (both soft and hard dies) begins.

After the draw die (binder surface and die angle) is designed and the plaster model built, if the panel periphery (especially the trim line location) is changed, the whole draw die design and plaster model construction may have to be repeated. In this sense, the sensitivity of draw die development to the periphery information is high.

Small to medium changes (of less than 2 mm) in the panel interior information, such as the handle depth, can be easily incorporated with a mere update of an engineering drawing – especially when the process engineers are confident the change will not impact the feasibility of the draw. In this sense, the sensitivity of the draw die development to the panel interior information is not high. However, to visualize the effect of large changes in the handle depth process engineers may have to revisit the die shop, and request for a modification of the plaster model with the application of clay (the procedure is called "softening or washing off of the plaster"). When the die shops are backed up (queued), implementing a large change in the pocket depth takes up to three weeks (including the time taken to get change approvals).

From the above, we observe that, (i) the panel interior does not affect the draw die design (binder die angles etc.) much and, (ii) the panel periphery is of the high sensitivity, fast evolution category, and the panel interior is of the

low sensitivity, slow evolution category with respect to the plaster model development activity. (Although the inner panel interior could be qualitatively termed slow in evolution, more detailed data was not available. So I focus on the outer panel of the door.)

Designer estimates suggest that the outer panel periphery is available in a fairly final form as early as eight weeks after theme release. From the above, we see that the die (binder, angles) design activity can start once the panel periphery information is available 8 weeks after theme release. However, the outer panel interior is undergoing much changes, and it would be desirable that the plaster model construction not start too early – as future changes in the interior may require time consuming changes in the plaster model. In Chapter 5, I have modeled the overlapping of the plaster model construction activity (lasting 4 weeks) with the door handle design activity. This required information about the evolution of the door handle which is described below. See Figure 5.4 for an example door handle interface with the door outer panel. The door handle concept (what type of door handle, how is it connected to the lock etc.) is finalized ten weeks after theme release. Then the problem of determining the handle profile (shape of the handle boundary), and handle depth starts.

Trade-offs in Designing Handle Profile and Depth.

The trade-off involved in the design of the handle profile and depth is described in this section. From a styling point of view, it is desirable that the handle look trendy, and flush with the outer panel. This means sharp corners in the handle profile, and large depth of draw in the pocket in the outer door panel (which houses the handle). From a functional standpoint, the door handle depth needs to be sufficiently large to take 200 pounds load (FEM helps ensure this). From a stamping standpoint, however, drawing a sharp pocket with a large depth poses technical problems (the metal might tear away). Engineers work with stylists, structural analysts, plastics vendors, and stamping engineers in converging to a profile and depth. This evolution of the door handle depth is shown in Figure 5.5 – obtained as an estimate from the engineer entrusted with the door handle at our study company. It can be seen that the engineers perceive that the styling input is the most important – the handle depth is evolved as much as sixty-percent once styling approves

the profile and depth 12 weeks after theme release, and two weeks after handle concept is selected. Finite element calculations, consultations with the vendor, and stamping engineers can help the handle engineers make fine changes which would make the handle economical, functional, and producible.

The model developed in Chapter 5 shows that at around the 14th week, plaster model production can start – as future changes are not likely to be high enough to require reiteration of the entire die design activity. This helps reduce four weeks in the development time of the outer panel. Similar calculations needs to be done for the door inner panel; otherwise the inner panel development will form the critical path, and the suggested lead time reduction may not translate into a net reduction for the entire vehicle system.

Appendix A3 Details of the Instrument Panel Example

The instrument panel (I/P) system in a modern day automobile is a highly coupled system – due to interactions with structural components, body attachments, windshield, dash panel, A-pillar, floor, ducts, door and other large subassemblies such as the instrument cluster, steering wheel and air bags (driver and passenger side). The I/P group at our study company is responsible for the following major parts (see Figure 1):

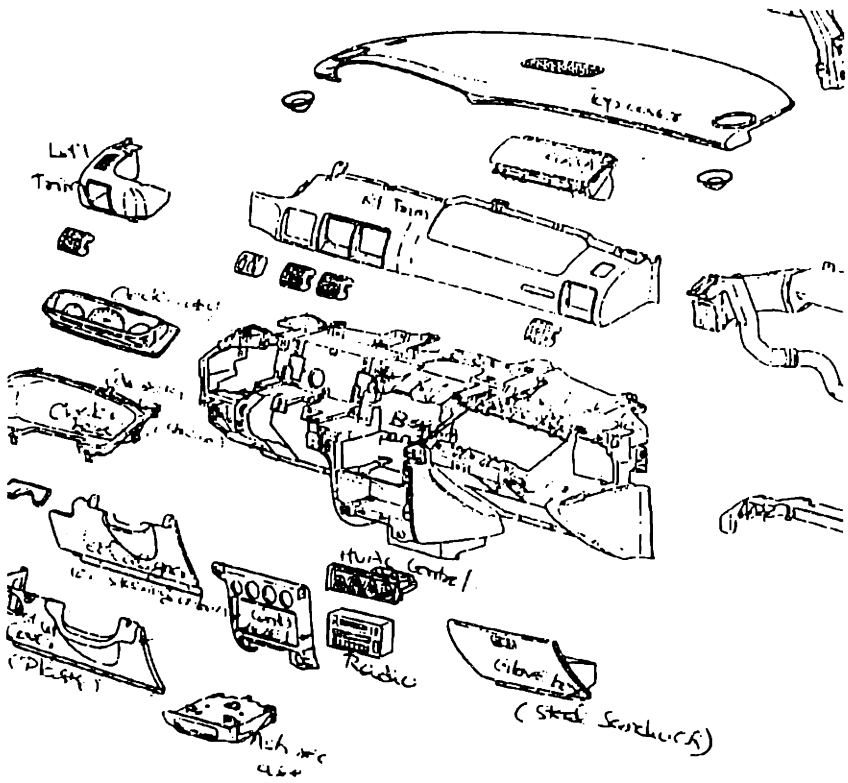


Figure 1: Parts of an Instrument Panel System

Base Panel: Base panel, (the most complex of all the I/P parts located at the center in Figure 1), is an injection molded polypropylene part in the studied process. It has many intricate formations to house the air bags, instrument cluster, ducting for HVAC (heat–ventilation–a/c), glove box, radio and the bezels. Changes in any of these parts causes a change in the base panel, making the development of base panel the most challenging task. A typical

base panel is shown in Figure 1.9 (Chapter 1). In an i/p assembly, the base panel is sandwiched between the trim panels and the reinforcements.

Steel Reinforcement assembly: This part (not shown in Figure 1), ensures the structural integrity of the I/P. It also has a safety function: ensures controlled crush and minimal injury by taking loads and blocking passenger knee; In an accident, when the knee of driver crushes against the steering column cover (or the glove box for the passenger), the load is transmitted to this reinforcement.

Right Trim: This is the ornamental part that the passengers see prominently (located above the base panel in Figure 1). Of importance in the development process is the appearance of the right trim surface, and so styling is actively involved. Also important is the choice of material for the trim panel to minimize head injury to the passenger. If the material is too soft, it could let the head pierce through it, receiving lacerations. If the material is too rigid, it might decelerate the head too fast (near 100g). The material should be fairly rigid, yet ductile (medium soft) to limit deceleration to 60g.

Left Trim: This is located at the left end of the I/P system (left of right trim in Figure 1), and interfaces with the base panel, cluster, top cover and door opening. The complexity is in the design of the vanes in the A/C outlets to ensure proper climate control within the car.

Cluster and Center Bezels: These are ornamental parts that dress the instrument cluster and the HVAC control, respectively. They are relatively simple parts to design and are made of injection molded plastic.

Glove box: This is the storage space located below the passenger air bag (PAB). Needs to be designed under-flush with respect to the base panel, and should extend all the way to the right side door. Reinforced with steel to block passenger knees. Interfaces with the PAB, HVAC, doors, and the base panel.

Top cover: Although deceptively simple in its appearance, the top cover (top-most visible part in Figure 1) is tough to get it right the first time because of (i) its interface with the A-pillar (ii) importance of its appearance (iii) high thermal stresses (subject to great temperature variations because of periodic exposure to sun). The top cover surface should be forgiving to allow for changes in the "A" Pillar– I/P interface. Further, it should have surface lines

flowing smoothly. The bump in the top pillar, called the cluster brow (which helps improve driver visibility of the instrument cluster by preventing day time glare and night time reflection) further complicates the design process.

Steps in the I/P Development Process

Concept and Theme Development: As in the door development process, this step begins with the (i) collection of customer preferences, and (ii) establishment of hard points and build objectives. Several themes are generated and developed in detail (with wireframe sketches, feasibility studies and "preliminary packaging"). Based on these studies, a single theme is selected, and refined with further feasibility evaluations. The approved theme is digitized and transferred to product engineering.

Engineering Design: Product engineers do more local packaging and detailed design for components within an I/P system ("Design for Component Fit"). The design details decided in this stage include spaces and clearances between components, attachment schemes, and the physical layout of components based upon their environment. The first iteration of this stage lasts 10 weeks, and is explained in more detail later.

Design Aids Construction: A mockup of the packaged i/p system is built to verify the fit between the various i/p components. Design aids, made of fiber glass by external vendors and assembled in-house, take 15 weeks to be built after the engineering design drawings are released in the as-is process. (This process will be described in detail later in this appendix). The mock-ups are reviewed by designers and design changes made.

Tooling Development: The finalized and released design drawings are used to develop tooling (for the base panel), to produce prototype parts, to assemble prototypes, and to subsequently get parts approved.

A network representation of the timings of the various i/p development activities (not reproduced here) shows that the engineering design phase, and design aid construction are time critical. It is therefore useful to consider the overlapping of these two phases. To do so, I study these two phases in more detail.

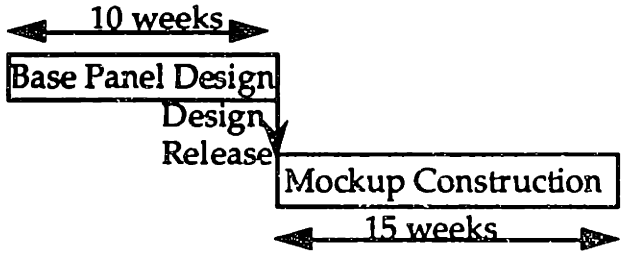


Figure 2: As-is Process of Base Panel Development

My Focus: Engineering Design and Design Aids Construction Phases

During this period, the components *within* an I/P are packaged, the fit among them established and a first iteration of mockup done. It is preceded by gross packaging in which the I/P as a whole is packaged with adjacent components; it is succeeded by a second iteration of design and mockup construction. We will focus on the design and the mockup construction of most complex, critical path item – the base panel (see Figure 2).

During the design phase, the base panel interface with the other I/P components is determined using (i) the creation of studies based upon I/P environmental layouts (ii) drawings which define surfaces and relationships in order to satisfy requirements (iii) the initial layouts of i/p which attempt to locate components using styling recognized surfaces and (iv) creation of graphics and (selective) mockups which identify all parts and subassemblies. The driver-section (left third) of the base panel is generally the most time consuming portion to design due to its interface with the highly change-prone instrument cluster and steering column. The passenger section (the right third) is the easiest as it interfaces with the glove box and PAB, whose packaging is done early. The center section is of intermediate complexity. Like in the door case (where the periphery evolved differently from the interior), here the three sections (driver section, center section and the passenger section) evolve at different rates.

The full-scale mockup construction proceeds by first carving the base panel in wood and then creating a mold out of the wooden model. In the as-is process, once the design is finalized, a quote package is assembled which instructs the mockup suppliers what they are expected to do. Suppliers quote (price and time) and one supplier is selected – one week after the receipt of the drawings from the engineering designers. The supplier takes nine weeks

to create the wooden model, three weeks each for the right, center and the left thirds of the panel. This time includes the time taken to add fillets, inspect and marry the three panels. It takes approximately a week to create the mold. In the next two weeks, fiber glass parts are created of the mold, and ribs and bosses are added. The final fit takes a week and the parts are ready for review 15 weeks after design release to mockup group. The intermediate steps and the timeline are summarized in Figure 3.

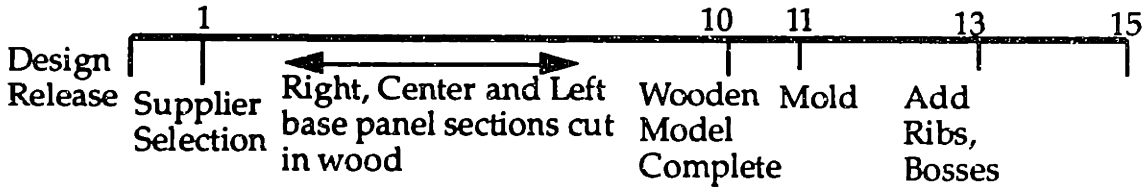


Figure 3: Mockup Construction Details

Base Panel Evolution and Sensitivity

As mentioned earlier, the base panel in its entirety is not finalized until the 10th week into the engineering design process. The evolution of the base panel with time is plotted in Figure 4. Because, changes in other components cause changes in the base panel, lot of changes are happening until the very end of the design process. (Example of late changes include change in the steering column mount location, change in the cluster brow and spacing etc). In other words, the evolution of the base panel can be characterized as slow. Further, the base panel information cannot be used in its advance form because, the changes in the base panel features, formations and split lines have a large time impact on the mockup construction activity. Often, small changes in the location of the clusters or glove box or ducting may require the wooden mockups to be started all over. The sensitivity of mockup construction to changes in the base panel can therefore be described as high and it is not practical to start mockup construction with unfinalized base panels. Using the framework, we determine the base panel information may have to be disaggregated to facilitate overlapping.

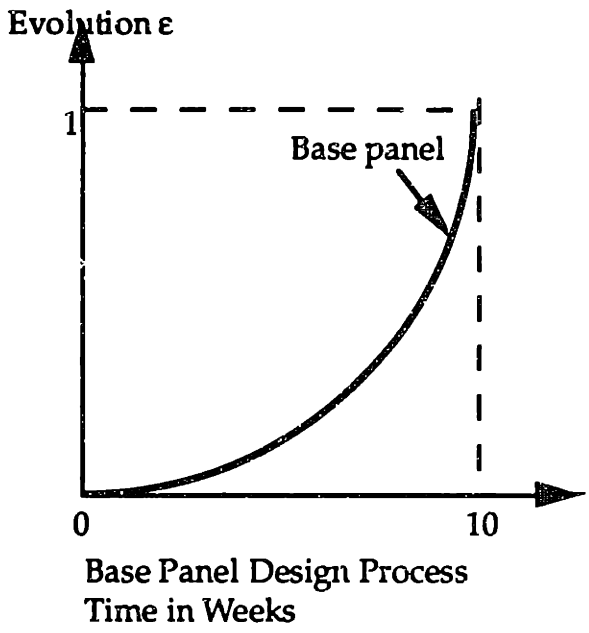


Figure 4: Evolution of the Base panel in its entirety

Evolution of the Base Panel Components

As mentioned earlier, *parts of the base panel evolve faster than the others*. For instance, the right third of the base panel does not undergo much changes after the 5th week because it does not interact with the change-prone steering column, and clusters. The evolution of the passenger section (right side) of the base panel is described in Figure 5. (based on estimates from the senior engineer of the I/P group). The center section of the base panel, which accommodates the center bezel, ducting and the HVAC unit, is a little more fluid but not as much as the driver section (left third) of the base panel. At the 5th week, one might picture it to be about 50% complete (again senior engineer's estimate). The driver section of the base panel evolves much more slowly than the other two parts, it is only 20% evolved half way down the design process. Lots of changes happen between the 5th and 10th week of the design process (due to changes in the cluster and the steering column).

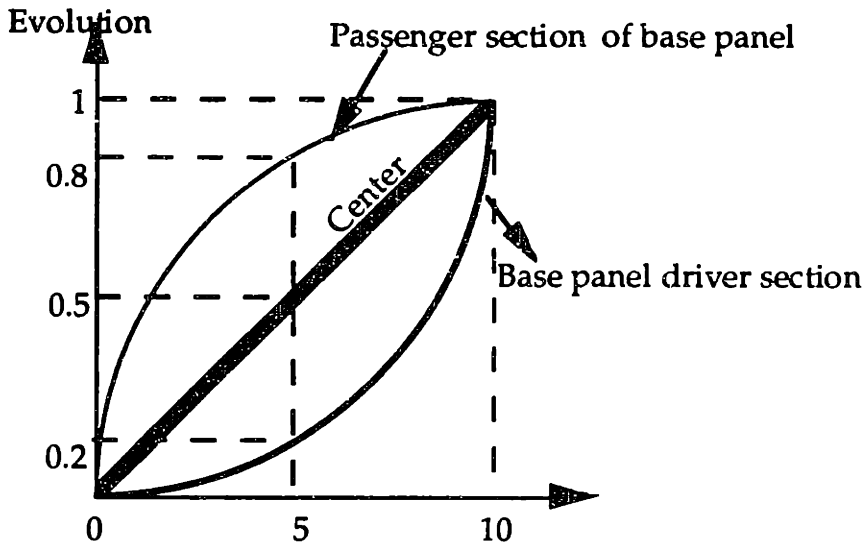


Figure 5: Evolution of parts of the base panel

It is clear that the base panel passenger section could be described as evolving fast, the center moderately fast and the left base panel as evolving rather slowly. Further, since the wooden mockups of the three parts are built separately, we have three cases to consider:

- Base panel passenger section design and mockup construction which can be described as a high sensitivity-fast evolution situation where precipitative overlapping is applicable.

- Center section panel design and mockup construction which can be described as a high sensitivity-medium evolution situation where precipitative overlapping may be applicable.

- Driver section design and mockup construction which can be described as a high sensitivity-slow evolution situation where overlapping would not be very useful.

Precipitating the evolution of the passenger section to a freeze situation at an earlier point in time may constrain the base panel designers in the sense that they have to design the rest of the parts based on the frozen passenger section. In a general case, we want to capture the loss of flexibility as a Quality Loss function. The quality loss of each section of the base panel, for freezing it at an earlier point in time t_F is, for the sake of this example, defined as the percentage of the design problem of that section left unresolved at t_F.

Mathematically, the quality loss equals ($100*(1-\varepsilon_F)$). Thus freezing the base panel passenger section at the 7th week (when $\varepsilon_F = 0.9$ approx.), would result in a quality loss of 10 (no dimensions). Freezing the driver section of the base panel at the 7th week (when $\varepsilon_F = 0.2$), would involve a quality loss of 80.

The quality-time trade-off curves for two different base panel decision making strategies are presented in Figure 5.9. In strategy (a) the passenger section is frozen at t_F, three weeks before center section, and six weeks before driver section. In strategy (b) the order of freeze is reversed to driver, center and passenger section. Each point in the curve is plotted for a particular value of t_F. It is clear that strategy (a) results in a lower quality loss and lead time performance than strategy, and is the preferred strategy.

Appendix A4 Evolution and Influence Functions

In this appendix, I investigate the impact of the algebraic form of the evolution and influence functions on the optimal overlapping policy. Also, I relax the assumption of monotonic evolution (made in Chapter 4) to see how it alters the overlapping model presented in Chapter 5.

Impact of Algebraic Form of Evolution and Influence Functions

In Chapter 5, I noted that the ideal concurrent process (with zero waiting time between and before iterations) may not always be optimal. For what kind of processes is the condition of zero waiting time between iterations optimal? To answer this question, I consider the iterative overlapping case in a product development situation which uses dedicated platform teams (so effort is committed through the process and its cost for the two stages under consideration may be ignored) and the cost of iteration is negligible. The only term in the objective is the development lead time, λ.

The Result

It can be shown that for linear evolution and influence functions, at the optimal point, the wait time between iterations is always zero, while the wait time before planned iteration could be nonzero. For nonlinear evolution and influence functions, both wait times could be nonzero.

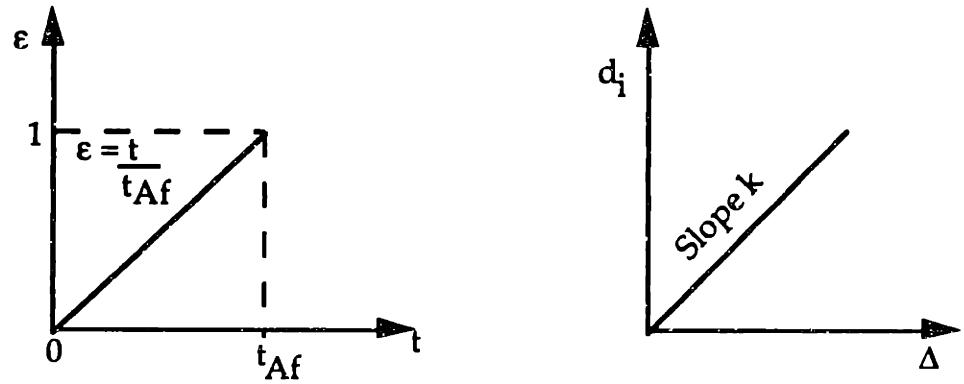


Figure 1: Linear Evolution and Influence Functions

For linear evolution and influence functions, we have from Figure 1:

$\varepsilon_i = t_i/t_{Af}$; (such that at $t = t_{As} = 0$, $\varepsilon_{As} = 0$; at $t = t_{Af}$, $\varepsilon_{Af} = 1$)

$d_i = k^*\Delta x(t_{i-1}, t_i) = k^*\beta\,(\varepsilon_i - \varepsilon_{i-1}) = k^*\beta^*(t_i - t_{i-1})/t_{Af} = \kappa^*(t_i - t_{i-1})$

where k is the slope of the influence function and $\kappa = k^*\beta/t_{Af}$

Note that the duration of the last iteration is a special case: $d_n = \kappa^*(t_{Af} - t_{n-1})$

The iterative overlapping problem becomes:

Choose $\mathbf{P} = \{n, t_0, t_1, t_2, \ldots, t_n\}$ to

Minimize $\lambda = t_n + d_n$

subject to:

$t_i \geq t_{i-1} + d_{i-1}$	i= 1, 2, ..., n	(1)
$t_0 \geq t_{As}$		(2)
$t_{As} \leq t_i \leq t_{Af}$	i = 1, 2, ..., n-1	(3)
$t_n \geq t_{Af}$		(4)
$d_i = \kappa^*(t_i - t_{i-1})$	i= 1, 2, ..., n	(5)
$d_n = \kappa^*(t_{Af} - t_{n-1})$		(6)

Because the lead time λ grows monotonically with t_n, either (1) or (4) must be active at the optimum. In other words at the optimum, either $t_n = t_{n-1} + d_{n-1}$ or $t_n = t_{Af}$. I will consider each of the branches.

- If $t_n = t_{Af}$ is active, then{

  $\lambda = t_{Af} + \kappa^*(t_{Af} - t_{n-1})$ because $d_n = \kappa^*(t_{Af} - t_{n-1})$.

  To keep the lead time bounded with respect to t_{n-1},

  the constraint $t_{n-1} + d_{n-1} = t_{n-1} + \kappa^*(t_{n-1} - t_{n-2}) \leq t_n$ should be active.

  So, $t_{n-1} + \kappa^*(t_{n-1} - t_{n-2}) = t_n = t_{Af}$ and by rearranging terms,

  $t_{n-1} = (t_{Af} + \kappa^* t_{n-2}))/(1 + \kappa)$

  By substituting for lead time, we can show that $t_{n-1} \geq t_{n-2} + d_{n-2}$ should be active to keep the objective bounded with respect to t_{n-2}. So $t_{n-1} \geq t_{n-2} + d_{n-2}$. Proceeding in a similar fashion, we can show that all of the constraints,

  $t_i \geq t_{i-1} + d_{i-1}$ i= 1, 2, ..., n-1 are active.

  So, $t_i = t_{i-1} + d_{i-1}$

  Also from (5), $d_i = \kappa^*(t_i - t_{i-1}) = \kappa^* d_{i-1}$. This results in

$d_1 = \kappa^* d_0$;

$d_2 = \kappa^{2*} d_0$

...

$d_n = \kappa^{n*} d_0$

So $\lambda = t_n + d_n = t_{Af} + \kappa^{n*} d_0$

where n is such that $t_{Af} - (d_0 + \kappa^* d_0 + \kappa^{2*} d_0 + ... + \kappa^{n-1*} d_0) \geq t_{As}$

Thus we note that all of constraints (1) need be active, but (2) need not be.

}

- In the previous branch, we observed, $t_n = t_{Af}$. If $t_n = t_{n-1} + d_{n-1} > t_{Af}$, then

{

$\lambda = t_{n-1} + d_{n-1} + d_n = t_{n-1} + \kappa^*(t_{Af} - t_{n-2})$ with $t_{n-1} + d_{n-1} > t_{Af}$

Because the objective grows monotonically with t_{n-1}, $t_{n-1} \geq t_{n-2} + d_{n-2}$ is active.

So $t_{n-1} = t_{n-2} + d_{n-2}$ and $\lambda = t_{n-2} + \kappa^*(t_{Af} - t_{n-3})$. Proceeding in a similar fashion, we can show that all the constraints (1), $t_i \geq t_{i-1} + d_{i-1}$ are active.

The lead time becomes $\lambda = t_0 + d_0 + d_1 + d_2 + ... + d_n$

As in the previous case, $d_1 = \kappa^* d_0$; $d_2 = \kappa^{2*} d_0$; ... $d_n = \kappa^{n*} d_0$

The objective becomes $\lambda = t_0 + d_0 + \kappa^* d_0 + \kappa^{2*} d_0 + ... + \kappa^{n*} d_0$

To bound objective, (2) is active. $t_0 = t_{As} = 0$

The lead time becomes $\lambda = t_{As} + d_0 + \kappa^* d_0 + \kappa^{2*} d_0 + ... + \kappa^{n*} d_0$

where n is such that $t_{As} + d_0 + \kappa^* d_0 + \kappa^{2*} d_0 + ... + \kappa^{n-1*} d_0 \geq t_{Af}$

}

Which of these branches prevails? The branch that leads to the minimum lead time based on the design situation does. Thus:

$$\lambda = \text{Min}\{(t_{Af} + \kappa^{n*} d_0), (t_{As} + d_0 + \kappa^* d_0 + \kappa^{2*} d_0 + ... + \kappa^{n*} d_0)\} \quad (7)$$

For the class of linear evolution and influence functions constraint (1) is always active – each subsequent iteration starts immediately after the previous iteration– while constraint (2) could be inactive denoting a period of wait before starting the nominal iteration.

With nonlinear evolution and influence functions, still one of the constraints (2) or (4) is active for the last iteration. i. e. $t_n = t_{n-1} + d_{n-1}$ or $t_n = t_{Af}$. But when, we move further, we find that either of the constraints (1) and (2) could be inactive for the other iterations– especially during periods of

rapid evolution of upstream information or in regions of high influence (sensitivity). This is as follows.

Suppose $t_n = t_{n-1} + d_{n-1}$. Then, $\lambda = t_{n-1} + d_{n-1} + d_n$. Consider the product development case, where more we wait (larger t_{n-1}), more the engineering change (a period of change in the design process); if the influence function is monotonically increasing, then more is the duration of the current iteration i. e. d_{n-1} increases with t_{n-1}. But if we are waiting during a period of rapid engineering change then all the change is incorporated in the current iteration and less change needs to be incorporated in the next iteration. So the duration of the next iteration d_n decreases with t_{n-1}. The objective, lead time, which is the sum of these durations does not grow monotonically with t_{n-1}. It is not likely that the constraint (1) may be active in this situation. (This suggests that it may be better to wait for the information to evolve than to execute the downstream activity with advance information during periods of high evolution).

Relation between speed and the number of iterations

For linear evolution and influence functions, we found in relation (7):

$\lambda = \text{Min}\{(t_{Af} + \kappa^n * d_0), (t_{As} + d_0 + \kappa * d_0 + \kappa^2 * d_0 + \ldots + \kappa^n * d_0)\}$

If $t_{Af} + \kappa^n * d_0 \leq t_{As} + d_0 + \kappa * d_0 + \kappa^2 * d_0 + \ldots + \kappa^n * d_0$,

then $\lambda = t_{Af} + \kappa^n * d_0$. In this case, lead time grows monotonically with n.

If $\kappa < 1$, then *larger the number of downstream iterations, smaller the lead time*! In this sense, speed is not free and takes more iterations. If $\kappa > 1$, then smaller the number of iterations, smaller the lead time.

In this case, iterations grow in duration and cause an increase in development time, so are not desirable.

If $t_{Af} + \kappa^n * d_0 > t_{As} + d_0 + \kappa * d_0 + \kappa^2 * d_0 + \ldots + \kappa^n * d_0$, then

$\lambda = t_{As} + d_0 + \kappa * d_0 + \kappa^2 * d_0 + \ldots + \kappa^n * d_0$

$= t_{As} + \{d_0 * (1 - \kappa^n)/(n - 1)\}$ if $\kappa < 1$.

$= t_{As} + \{d_0 * (\kappa^n - 1)/(n - 1)\}$ if $\kappa > 1$.

Now the objective does not necessarily bear a globally monotonic relationship with n. So it is not necessary that number of iterations be met at either the upper or lower bound. Hence speed (reduced lead time) may or may not require more iterations.

Nonmonotonic Evolution

In the previous analyses, I considered the monotonic evolution of the design interval such that for $t_j \geq t_i$, the design interval $\{a_j, b_j\}$ is a subset of $\{a_i, b_i\}$ $(a_j \geq a_i; b_j \leq b_i)$. However, this is not a general picture. One could conceive of situations in which the evolution is not monotonic. In such a case, it is not necessary that $\{a_j, b_j\}$ be a subset of $\{a_i, b_i\}$. However, to keep both these intervals bounded, I require that they be within the initial interval $\{a_{in}, b_{in}\}$. In other words, we have, $a_j \geq a_{in}$; $b_j \leq b_{in}$; $a_i \geq a_{in}$; $b_i \leq b_{in}$. I derive a relation relating evolution to the design change below. For nonmonotnic evolution, the maximum estimated change between times t_1 and t_2 is shown to be:

$$\Delta x(t_i, t_j) = \beta^* (\varepsilon_j + \varepsilon_i)$$

From the definition of evolution above, we have:

$$k^*(b_i - a_i) = 1 - \varepsilon_i ;$$

$$k^*(b_j - a_j) = 1 - \varepsilon_j$$

$$(b_j - a_j) = \frac{1 - \varepsilon_j}{1 - \varepsilon_i}(b_i - a_i)$$

Since downstream requires a point estimate of x from upstream and with a uniform distribution in the design interval, the estimate that will be provided by the upstream activity at t_i will equal the expected value, $(a_i + b_i)/2$. Similarly at time t_j, the estimate will be $(a_j + b_j)/2$. We want to find the value of a_j and b_j that maximizes the absolute value of the change in the expected values, $(a_j + b_j)/2 - (a_i + b_i)/2$.

Now we have a simple mathematical programming problem:

$$\text{Maximize} \left| \frac{a_j + b_j}{2} - \frac{a_i + b_i}{2} \right|$$

subject to:

$$b_i \leq b_{in}; \; b_j \leq b_{in}$$

$$a_i \geq a_{in}; \; a_j \geq a_{in}$$

$$(b_j - a_j) = \frac{1 - \varepsilon_j}{1 - \varepsilon_i}(b_i - a_i)$$

There are two solutions (based on the sign of $\left\{\frac{a_j + b_j}{2} - \frac{a_i + b_i}{2}\right\}$):

(1) $b_j = b_{in}$; $a_j = b_j - k^* (1 - \varepsilon_j)$; $a_i = a_{in}$; $b_i = a_{in} + k^* (1 - \varepsilon_i)$

for $\left\{\left(\frac{a_j + b_j}{2} - \frac{a_i + b_i}{2}\right)\right\} \geq 0$ and

(2) $a_j = a_{in}$; $b_j = a_{in} + k^* (1 - \varepsilon_j)$; $b_i = b_{in}$; $a_i = b_{in} - k^* (1 - \varepsilon_i)$

for $\left\{\left(\frac{a_j + b_j}{2} - \frac{a_i + b_i}{2}\right)\right\} \leq 0$

For both the cases, the maximum expected change = $\Delta x(t_i, t_j) = k/2\ (\varepsilon_j + \varepsilon_i)$

How do we determine the value of k? If we substitute $\varepsilon_i = 0$ and $\varepsilon_j = 1$, (i. e., we consider the two extremes) Estimated change = k/2. We will call this the Change Coefficient (β). If at $\varepsilon_i = 0$, the design interval = $\{a_{in}, b_{in}\}$, then the estimate is $(a_{in} + b_{in})/2$; at $\varepsilon_j = 1$, the design interval is of zero width and within $\{a_{in}, b_{in}\}$.

The maximum expected change from 100% to 0% equals $\frac{b_{in} - a_{in}}{2}$.

So $\beta = \frac{b_{in} - a_{in}}{2}$.

Estimated change = $\Delta x(t_i, t_j) = = \beta^*(\varepsilon_j + \varepsilon_i) = \frac{b_{in} - a_{in}}{2}\ (\varepsilon_j + \varepsilon_i)$