

**Process Improvement Applied to Product Development:
An Approach to Control of Development Lead Time**

by
Carey W. Mar

B. S. Mechanical Engineering, University of Waterloo 1994

Submitted to the Sloan School of Management and the
Department of Mechanical Engineering in partial fulfillment
of the requirements for the degrees of

**Master of Science in Management
and
Master of Science in Mechanical Engineering**

in conjunction with the
Leaders for Manufacturing Program

At the
Massachusetts Institute of Technology
June 1999

©1999 Massachusetts Institute of Technology, All rights reserved

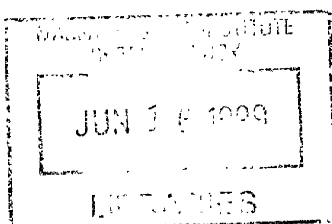
Signature of Author _____
Sloan School of Management
Department of Mechanical Engineering

Certified by _____
Steven D. Eppinger, Thesis Advisor
Associate Professor of Management Science

Certified by _____
Daniel E. Whitney, Thesis Advisor
Senior Research Scientist, CTPID, Lecturer, Department of Mechanical Engineering

Accepted by _____
Ain Sonin, Chairman, Graduate Committee
Department of Mechanical Engineering

Accepted by _____
Lawrence S. Abeln, Director of the Masters Program
Sloan School of Management



ARCHIVES

This page is intentionally blank.

Process Improvement Applied to Product Development: An Approach to Control of Development Lead Time

by
Carey W. Mar

Submitted to the Sloan School of Management and the Department of Mechanical Engineering
on May 7, 1999
in partial fulfillment of the requirements for the degrees of

Master of Science in Management and
Masters of Science in Mechanical Engineering

Abstract

Speed and flexibility in product development will be the differentiators that determine which companies will have competitive advantages in the next decade. To this end, many authors have advanced frameworks and tools for streamlining product development processes, all with the objective of reducing the time required to engineer and deliver a quality product. Frameworks such as concurrent engineering describe what the process should evolve to, while tools such as cross-functional teams address implementation. But one element that has not been clearly addressed is the measurement and improvement of the development process during the pursuit of this goal.

The focus of this thesis is on the design of an approach to improve the product development process, based on applying established process improvement techniques and project simulation. The objective of the improvement effort is to increase control over development lead time. Achieving this control promotes efficient product development by allowing the development team to: i) improve speed and flexibility, and ii) confidently evaluate the trade-off between them when necessary.

A “Voice of the Customer” study of the product development process at a large American automobile manufacturer was used to determine inhibitors of this control and efficiency. Considering the relationship between these inhibitors, three improvement efforts were identified:

- Track and address the causes of delay
- Optimize schedule recovery actions, and
- Develop a more representative measure of project progress.

To implement these efforts, two tools are discussed. A delay tracking framework was designed to collect and analyze activity delays. This analysis focuses the reduction of controllable causes of delay and aids in the determination of the efficacy of schedule recovery actions.

A project simulation was used to illustrate the expected effects of implementing the three improvement efforts. The simulation can be used to generalize the learnings from tracking delay to aid in making project management decisions. More specifically, the use of the simulation for evaluating schedule recovery actions and the effect of proposed changes in design requirements is discussed.

Thesis Advisors:

Steven D. Eppinger, Associate Professor of Management Science

Daniel E. Whitney, Senior Research Scientist, CTPID, Lecturer, Department of Mechanical Engineering

Acknowledgements

Given that the motivation and basis for this study were the numerous interviews with various product development staff at the internship company, I would like to express my sincere appreciation to those who donated their time and insight into the company's product development process. Their experience and candid comments were invaluable.

I would also like to thank my classmates for making the Leaders for Manufacturing Program a rich and exciting learning experience. I can only hope that in my career I will be fortunate enough to work with such talented, inspiring, and dedicated peers.

Thank you also to my advisors, Steven Eppinger and Daniel Whitney for their support of this work and for their valuable insight.

Table of Contents

1. DEFINING THE GOAL: THE IDEAL PRODUCT DEVELOPMENT PROCESS	7
1.1 SPEED	7
1.2 FLEXIBILITY	9
1.3 THE BUSINESS INITIATIVE	10
1.4 SUMMARY OF THE BUSINESS INITIATIVE & STRUCTURE OF THE THESIS	11
2. IDENTIFYING THE ISSUES: VOICE OF THE CUSTOMER	13
2.1 PROACTIVE PROCESS IMPROVEMENT.....	13
2.2 IDENTIFYING OBSTACLES TO IMPROVEMENT OF THE PD PROCESS.....	13
2.2.1 Existing Project Management: Background.....	14
2.2.2 Identifying the Customer Base and Focussing the Analysis	15
2.3 PROCESS ISSUES	15
2.3.1 The “Design Churning” Phenomenon.....	17
2.3.2 Poor Progress Measure	28
2.3.3 Reactive Schedule Recovery.....	31
2.4 ORGANIZATIONAL ISSUES.....	32
2.4.1 Low Credibility Schedules.....	34
2.4.2 Lack of Urgency	35
2.5 CHAPTER SUMMARY.....	37
3. DEFINING THE PROBLEM: DEFINING IMPROVEMENT EFFORTS	39
3.1 LINKING THE PROCESS AND ORGANIZATIONAL ISSUES	39
3.2 ADDRESSING THE PROBLEM: 3 IMPROVEMENT EFFORTS.....	42
3.2.1 Track and Address Controllable Delay	44
3.2.2 Optimize Effectiveness of Recovery Actions	47
3.2.3 Develop a More Representative Measure of Project Progress	48
3.3 CHAPTER SUMMARY.....	49
4. SOLVING THE PROBLEM: TOOLS FOR IMPROVEMENT	51
4.1 PROCESS IMPROVEMENT: THE TQM FRAMEWORK.....	51
4.1.1 Process Control.....	51
4.1.2 Reactive Improvement.....	52
4.2 A DATA COLLECTION TOOL: THE KEY TO PROCESS IMPROVEMENT	55
4.2.1 Inputs	56
4.2.2 Outputs.....	62
4.3 A SIMULATION TO DEMONSTRATE THE IMPROVEMENT APPROACHES.....	69
4.3.1 Description.....	69
4.3.2 Modeling the Approaches.....	73
4.3.3 Discussion of Results: Demonstrating the Improvement Approaches	81
5. CONCLUSIONS & RECOMMENDATIONS	103
5.1 CONCLUSIONS	103
5.2 RECOMMENDATIONS	105
5.3 FURTHER WORK	107
5.3.1 Improving the Simulation	107
5.3.2 A More-Leading Indicator of Project Progress.....	109
REFERENCES	113
APPENDIX A: SOURCE CODE FOR SCHEDULE RECOVERY EXTENSION	115
APPENDIX B: CALCULATIONS.....	117

This page is intentionally blank.

1. Defining the Goal: The Ideal Product Development Process

This thesis represents the culmination of observations and ideas from an internship at a large American automobile manufacturer, hereafter referred to as “the company”.

Product development faces heightened challenges in today’s environment of rapid change and intense competition. The automotive industry, as well as many others, has witnessed an important shift in strategic needs. As cost and quality have been relegated from competitive advantages to prerequisites for competition, attention has turned toward rapid product development. More recently it has been recognized that flexibility in product development is valuable in reducing the cost and time of meeting changing customer needs.

Thus speed and flexibility in product development will be the new differentiators that determine which companies will have competitive advantages in the next decade. The remaining sections of this chapter discuss speed and flexibility in more detail and then integrate these needs to propose a business initiative.

1.1 Speed

In response to this shift in strategic needs, concurrent engineering has been embraced as an approach to reducing the time to market for product development. In its simplest sense, concurrent development involves the solicitation and use of information from downstream activities by upstream activities, i.e. getting early input. In most cases, concurrent development means that the activities are overlapped, i.e. that the downstream activity begins its work before the upstream activity has completely finished its work. This is a conceptually simple departure from the serial model of development where the downstream activity does not start until the upstream activity is completely finished. Theoretically, by overlapping activities, it is possible to reduce the start to finish duration of the project by the amount of overlap.

However, in reality the implementation is not so simple and the reduction in development lead time may be less than expected, or may not be realized at all. In most development projects there are numerous dependencies between the activities that create potential rework. Some of these relationships will be one-way dependencies where activity B depends on activity A but A does not depend on B. In this case, if activity A changes information used by B after B starts, activity B is forced to do rework. Other relationships will be *interdependencies*, i.e. activity B depends on A and activity A depends on B. In this

type of relationship, there is the potential for feedback. For example, say activity A passes information to activity B allowing B to proceed; but as B proceeds, information on which A depends may be changed, forcing activity A to do rework.

Regardless of the type of dependency though, the potential for rework reduces the predictability of project duration and the project can end up taking much longer to complete than scheduled. The overlapping that forms the basis for concurrent development adds to this uncertainty by increasing the probability that the information exchanged between these dependent activities will change, requiring activities that have used this information to re-do some or all of the work done since the original transfer of information.

Determining the optimal amount of overlap between two activities is not a trivial task. Krishnan et al. propose a model for determining when development activities should and should not be overlapped based on concepts they call evolution and sensitivity (Krishnan et al., 1997). The term evolution refers to the rate at which the range for a particular design parameter is narrowed by the upstream activity, i.e. as the design activity proceeds, the design parameter is defined as a progressively narrow range and is finalized when a nominal value is reached. The term sensitivity refers to how much rework the downstream activity must perform as a function of the size of the change in the design parameter made by the upstream activity. The more the amount of rework increases with the size of the change, the greater the sensitivity.

The most important, and possibly most challenging requirement of concurrent development is communication. Not only is it important that the actual transfer of information is defined and agreed to, any changes past the original transfer must be carefully coordinated.

To meet the need for rich and timely communication, cross-functional teams are typically established. These teams are typically organized to conduct a project and include representatives from each of the required functional groups. By increasing allegiance to the project (as opposed to the function), this organizational structure promotes more frequent and earlier communication between the functional representatives. In addition, these teams are typically co-located to facilitate frequent informal communication, which is as or even more important for avoiding problems than formal meetings.

If the benefit of concurrent development is to be realized, i.e. project lead time is consistently less than serial development, the relationships between the activities must be understood. Research in this area has identified two dimensions of the relationship between two activities: the probability that rework will be

required, and the impact of that rework (how much work has to be re-done in the event that rework is required) (Carrascosa et al., 1998). Understanding these dimensions for the activities in the project allows activities to be planned and executed in such a way as to reduce the probability and impact of rework (Eppinger et al., 1994).

Secondly, recognizing the potential effect of these activity relationships on project duration, there are numerous causes of delay that can be associated with increased probability of rework and impact of rework. These must be understood and managed if development lead time is to be reduced. Thus it is these causes of delay that are the subject of this thesis.

1.2 Flexibility

More recently it has become recognized that in addition to increasing the speed of the development process, maintaining flexibility in design throughout as much of the process as possible, is a valuable advantage. In today's environment of rapidly changing technology, customer needs are subject to change and there is an increasing degree of market fragmentation (Wheelright and Clark, 1992, p.2). If development time could be reduced to be commensurate with the rate of this change, customer needs could be timely met – providing development costs can be reduced to the point where shorter product runs are profitable.

However, in many cases the development time required to meet the rate of change of customer needs and/or the fragmentation of the market into different niches is far less than the shortest development cycle that could reasonably be striven for. In these cases, it would be highly desirable to have a development process that allows changes to be made to the design in response to market changes – without severe time and cost penalties.

Approaches such as set-based design (Sobek et al., 1999) and intentional decision delay (Ward et al., 1997) are means by which this flexibility can be achieved. Providing this flexibility places additional demands on project management. In set-based design where a set of initial designs is progressively narrowed down to a single design, project management must facilitate the rate at which the set is narrowed in order to keep the project on schedule.

Because of the dependencies between activities described above, intentional decision delays upstream can affect many other downstream activities – directly delaying them or increasing the probability that they

will require rework because they are forced to start with incomplete information. Thus when using intentional decision delay Program Management must evaluate and minimize the consequences of postponing these decisions.

Another approach to flexibility is to simply allow the proposed changes in design requirements to be made. Because of the relationships between activities, making these upstream changes can have profound effects on downstream activities. Thus it is important that program management understand these relationships so that the implications of a such a change on project duration can be weighed against the benefit of making the change.

It is interesting to note that this objective of flexibility can be somewhat at odds with the objective of speed. As discussed above, reducing development lead time using concurrent development involves overlapping activities. While this offers the potential to reduce project duration, there is often a trade-off in flexibility since in order to overlap, the upstream activity generally finalizes information early to allow the downstream activity to proceed. Finalizing this information early can restrict the freedom of the upstream activity to change. The degree to which overlapping restrict flexibility depends on the characteristics of the activities being overlapped, and has been studied by Krishnan et al. (Krishnan et al., 1997).

1.3 The Business Initiative

Given these two requirements for the product development process, an overall objective can be defined. To achieve both speed and flexibility, the objective should first be to achieve *control* over development lead time. Note that this concept of control over development lead time is broader than the more commonly stated objective of reducing development lead time. A controlled development process is one where reduction of the incidence of unnecessary, unexpected but controllable delays increases the predictability of project duration. With control over development lead time, efficient product development can be achieved. An efficient development process is one in which:

- Development lead time is reduced by minimizing causes of unnecessary, controllable delay
- Flexibility is accommodated in activities where it is valuable (i.e. where market-driven changes are common but where any negative effect on project duration is low)
- Effects of a design change can be predicted, allowing trade-off decisions between responsiveness to a market change and lead time to be made with confidence

To achieve this overall objective, the causes of delay must be determined and the relationships between the activities must be understood. Knowing these causes and relationships, process improvement can be applied to reduce the delay and optimize the management of the project schedule. Hence the purpose of this thesis is to apply process improvement to the product development process in order to achieve control over development lead time.

1.4 Summary of the Business Initiative & Structure of the Thesis

Summarizing the discussion above, the need for speed and flexibility in product development can be integrated into an overall objective of increasing the efficiency of product development. Essential to reaching this objective is control over development lead time: reduced unnecessary, unexpected but controllable delays, and increased predictability of project duration.

Process improvement is an approach to achieving this control. Although it is more familiar in the context of improving quality of manufacturing processes, process improvement can be applied to product development. Shiba et al. describe process improvement in four iterative phases: Plan-Do-Check-Act, or PDCA (Shiba et al., 1993, p. 57). The structure of this thesis follows the PDCA framework and is outlined below.

The first phase of process improvement is PLANNING (Defining the goal and problems). Chapter 1 discussed the business need for efficient product development. Chapter 2 discusses proactive improvement, a type of process improvement used to establish direction for improvement efforts. The findings of a proactive improvement study to find inhibitors of efficient product development at a large automotive company are described. Chapter 3 describes how these issues are related and suggests three improvement efforts:

- track and address the causes of delay
- optimize schedule recovery actions, and
- develop a more representative measure of project progress.

The second phase of process improvement is DOING (Developing a solution). The first half of Chapter 4 outlines two implementation tools aimed at achieving the desired control and efficiency in product development. The first is a delay tracking and analysis tool to gather data about activity delays and their causes. The outputs of this tool contribute to the first two of the aforementioned improvement efforts. They also help determine parameters for the second tool which is the use of a project simulation to

analyze schedule recovery strategies, determine the effect of a change in design requirements, and improve the assessment of project progress. This simulation thus generalizes and applies the learnings from the delay tracking tool to future development projects.

The third phase of process improvement is CHECKING (Expected results). The second half of Chapter 4 discusses several simulation scenarios that were run to demonstrate the expected results of the three improvement efforts.

The last phase of the process improvement loop is ACTING on the results (Continuous improvement). Chapter 5 states conclusions and recommendations and then outlines opportunities for further work. Specific ways in which the functionality and accuracy of the simulation can be improved are described. And, in relation to the identified need for a more representative measure of project progress, the concept for a more leading indicator of project progress is proposed.

2. Identifying the Issues: Voice of the Customer

Having established the motivation for efficient product development in Chapter 1, this chapter outlines the method used to determine inhibitors to control of development lead time at a large American automobile manufacturer, and the results of the study.

2.1 Proactive Process Improvement

Proactive improvement is one of three types of process improvement (Shiba et al., 1993, p. 53). The objective of proactive improvement is to gain insight into the problems in a process and to identify general direction for improvement based on these problems. Having identified the problems, an objective and corresponding metric can be established. This metric is tracked using process control and then continually improved using reactive improvement. These latter two types of process improvement will be discussed in section 4.1 where the suggested solution to the problems in the development process are outlined.

A key tool for proactive improvement is “Voice of the Customer” analysis (Center for Quality Management, 1996). This tool is an approach to interviewing customers, organizing findings from those interviews, abstracting those findings, and then drawing conclusions and insights from them. A summary of the analysis method and the findings from its application to the product development process at the company are described in the following sections.

2.2 Identifying Obstacles to Improvement of the PD Process

In an effort to identify issues that inhibit control of development lead time, a voice-of-the-customer analysis was conducted among different vehicle development programs. Particular emphasis was placed on identifying problems related to project management.

This analysis involved:

- gaining an understanding of the existing project management process (by reviewing process documentation)
- identifying “customers” or stakeholders of the project management process
- identifying two key questions to focus the analysis:

1. What are the problems/issues that affect the timing of a program?
 2. What are the problems/issues related to supporting the workplanning function?
- conducting interviews with the stakeholders
 - grouping the interview comments (by similarity in meaning/thought)
 - determining cause and effect relationships between the groups of comments
 - drawing conclusions from these relationships to answer the two key questions.

2.2.1 Existing Project Management: Background

With the rollout of a revised product development process, the company also introduced formal project management roles and responsibilities. Each vehicle development team is led by a chief engineer (CE). Reporting to the CE is the program manager (PM). The PM's staff is responsible for tracking schedule, cost, and engineering changes. While the program management staff is dedicated to specific programs, a central process development group provides support in the form of standard development schedules and processes.

Schedule management is the responsibility of the workplanning staff that reports to the program manager. The workplanning staff includes a program schedule coordinator (PSC) and a number of schedule analysts (SAs). The workplanning staff is provided by a central workplanning group. The program schedule coordinators and analysts are responsible for different levels of project schedules as described below.

Formally, three levels of schedules are maintained, each with decreasing level of detail. Level 3 schedules coordinate activities at the system level (e.g. body, chassis) for each of the various engineering groups. They are maintained by the SAs with input from the engineering group supervisors. The level 2 schedule monitors linkages between the numerous level 3 plans and is used by the program's steering committee. It is the responsibility of the PSC to maintain this schedule. The level 1 schedule tracks selected high level activities and milestones for review by executive management. The PSC is responsible for preparing this schedule from the level 2 schedule. These plans are now being electronically linked so that activity status input need only be made at level 3.

Although not part of the formal program management role, there is typically a level 4 schedule that tracks commodities/groups of components (e.g. underbody, closures) or single, key components. These schedules are maintained and used by some of the engineering groups.

Schedule reviews occur at each of the three levels. The staff responsible for the particular schedule identifies current or potential near future issues prior to the review meeting. At each meeting, resolution plans are made to address these issues. Following the meeting, the PSC and SAs update their schedules.

2.2.2 Identifying the Customer Base and Focussing the Analysis

The stakeholders or customers of program management included: the engineering groups, SAs, PSC, program manager and chief engineer. Although they may use different levels of the project plan for different reasons, each customer can affect the integrity and therefore the utility of the plan. For this reason it was important to ensure that representatives from each role were interviewed.

A second consideration in the scheduling of interviews was the variation in complexity among different vehicle development programs. The scale of a vehicle development project reflects the degree or amount of new design in it and can range from minor “refreshening”, to the development of a completely new vehicle platform. Clearly the project scale has an effect on the amount of project management required and the number of complications that can arise. For this reason it was important to ensure that representatives from projects of various scales were interviewed.

In order to focus the analysis and differentiate between issues directly and indirectly related to the product development process, two key questions were used as the basis for generating interview discussion:

1. What are the problems/issues that affect the timing of a program? and
2. What are the problems/issues related to supporting the workplanning function?

The first question was aimed at finding issues related to the product development process itself. The second question was aimed at finding related organizational issues. With this distinction, the two analyses will be referred to as Process and Organizational respectively.

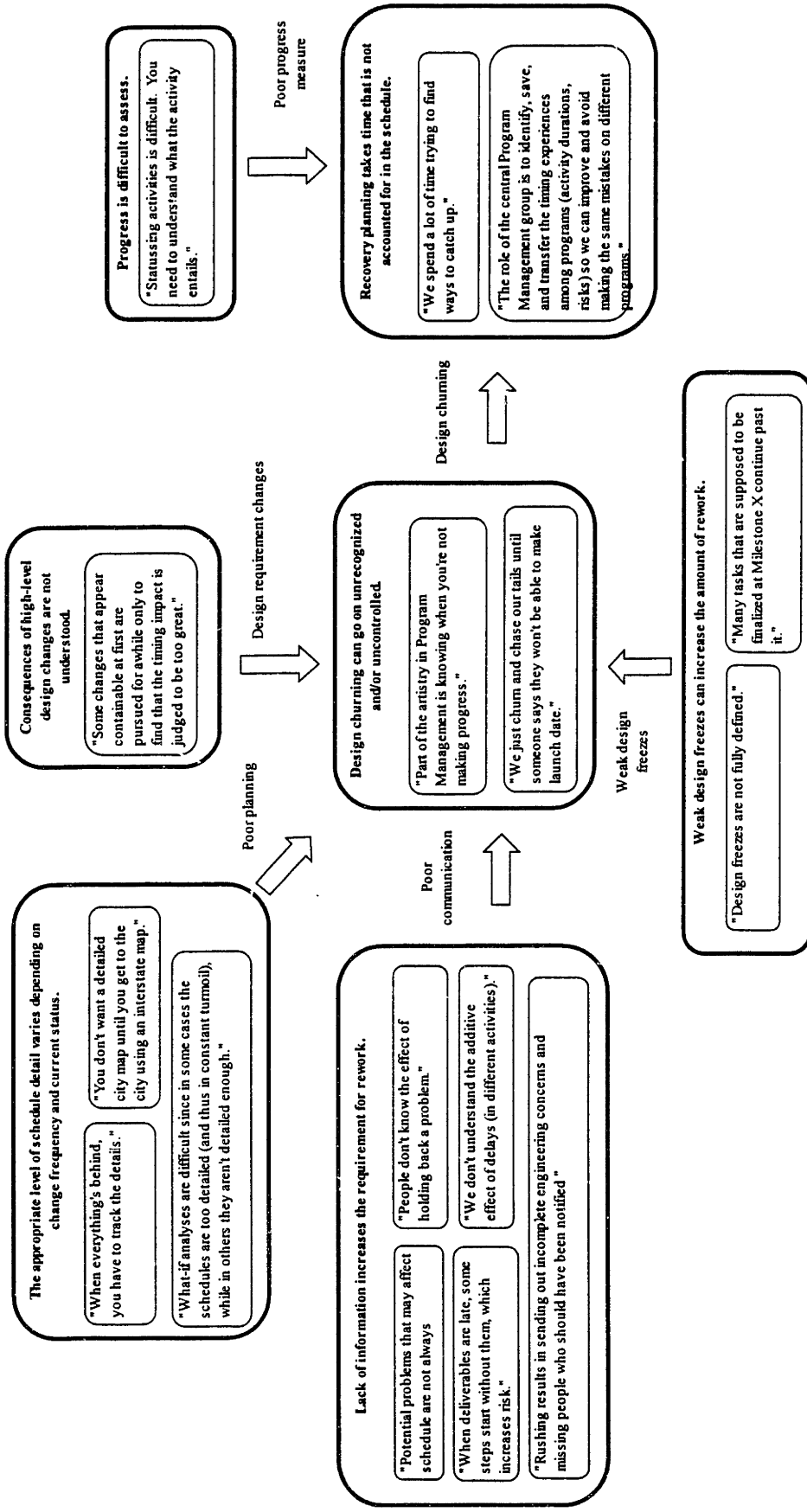
The following sections summarize the comments from the interviews and the conclusions that were drawn from their analysis.

2.3 Process Issues

Figure 1 summarizes the key interview comments related to the first key question, and the conclusion drawn from the cause and effect relationships identified between them.

Figure 1: Process issues

What are the problems/issues that affect the timing of a program?



Design churning and the difficulty in accurately measuring activity progress predispose the program toward reacting to schedule delay – as opposed to taking proactive action to avoid delay.

The issues in Figure 1 are related to the product development process itself. The following sections describe the conclusion in more detail, highlighting the issues inhibiting control of development lead time that were raised during the interviews. More specifically, the three parts of this conclusion will be discussed: the “design churning” phenomenon, poor progress measurement, and reactive schedule recovery. For each of these topics, the specific issues are described, the causes identified, and the consequences explained. The **white on black** headings refer to the box titles in Figure 1.

2.3.1 The “Design Churning” Phenomenon

In many of the interviews, “design churning” was a commonly cited cause of project delay. While this term can have many different interpretations and meanings, the term is used here to refer to avoidable rework. In other words, design churning is a situation where factors over which program management has some degree of control, cause increased design iteration or rework. This rework contributes to prolonged project durations.

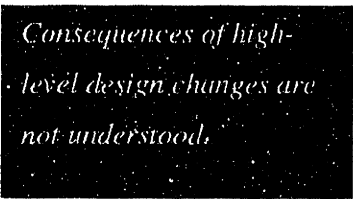
- Causes of Design Churning -

Based on the voice-of-the-customer analysis, four causes of design churning (factors that increase the probability and/or amount of rework and consequently the amount of delay) were identified:

- Design requirement changes
- Poor communication
- Poor planning
- Weak design freezes.

Each of these causes of design churning is discussed in more detail below. Note that a common connection between these causes is that they are all related to functions of Program Management and are to varying degrees, controllable by Program Management. This idea of control will be further developed in Chapter 3 when the improvement efforts for the development process are defined.

- Causes of Design Churning: Design Requirement Changes -



Consequences of high-level design changes are not understood.

The definition of design requirements occurs at the beginning of a project. Thus changes in design requirements after detailed design work has begun can cause significant amounts of rework, as the effects of the change ripple through the completed activities. Design requirement changes are changes in design objectives – as opposed to changes in the design itself. In other words, a design requirement change causes design changes. Such changes in requirements can be motivated by changes in customer needs or by design, manufacturing, or cost infeasibilities. These requirement changes may affect any of the concept phases in product development. In vehicle development for example, there are concept phases for appearance (styling), package (overall dimensions and proportions), and internal systems (powertrain, climate control, steering, etc.).

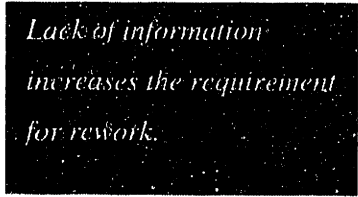
Of course these vehicle aspects are not independent of each other, but the development of each does have a distinct concept phase and design requirements could change for any of these. It should also be noted that many of the vehicle attributes of interest to the customer (such as fuel economy, safety, and vehicle handling) imply design requirements for more than one of appearance, package, and internal systems. Thus it is highly likely that a change in customer needs could affect several of these and have significant consequences on project completion time by causing rework for completed or on-going activities.

The general perception among the development teams at the company was that there is a high potential for changes in design requirements, and that these often originate in management reviews. Furthermore, the consequences of these changes on project schedule are not necessarily communicated to management and there tends to be a higher priority given to accommodating the changes.

Regardless of the priority between maintaining schedule and accommodating change, knowledge and communication of the consequences of a proposed change are valuable. In assessing these consequences, it is clearly important to identify which activities the change will affect and what the impact of the change will be on those activities. This information allows the proposed change

to be carefully evaluated and, if the decision is made to accept the change, appropriate plans to be made to reduce the impact and risk of the accommodating it.

- Causes of Design Churning: Poor Communication -



Lack of information increases the requirement for rework.

Many types of information are exchanged in a product development project including design requirements, design decisions, and test results. Information flow between and within project activities is the critical “glue” that enables concurrent product development and determines to a large extent, how much rework will be necessary. Browning describes the objective of synchronized information transfer in a design project as ensuring that all related activities have and are using the latest information.

He identifies three attributes of successful communication: making information available at the required time, place and format (Browning, 1998a). Implicit in this definition is that the information being transferred is indeed the required information. Another important attribute of successful communication is coordination, i.e. the information is transferred to all those affected by it.

Thus there are several ways in which the quality of communication can be reduced. Two forms of poor communication were cited at the company: late discovery of problems and poorly defined information exchanges. The relatively late discovery of problems with the design allows the consequences of them to grow, increasing the amount of rework that may be required. This can be seen by considering the concepts of information evolution in an upstream activity and the sensitivity to this information of a downstream activity (Krishnan et al., 1997) introduced in section 1.1.

In general, as time progresses, the design continues to evolve. Consider the evolution of a design as the narrowing of the range for a given design parameter X . As the range for X narrows, the flexibility to accommodate different values decreases. Now consider a problem with the design as the need to have a certain value of X , say X_1 . As time progresses, the gap between X_1 and the current state of X increases (see Figure 2). Since sensitivity (the amount of rework caused by a change in X) generally increases with the size of change, the impact of the design problem increases.

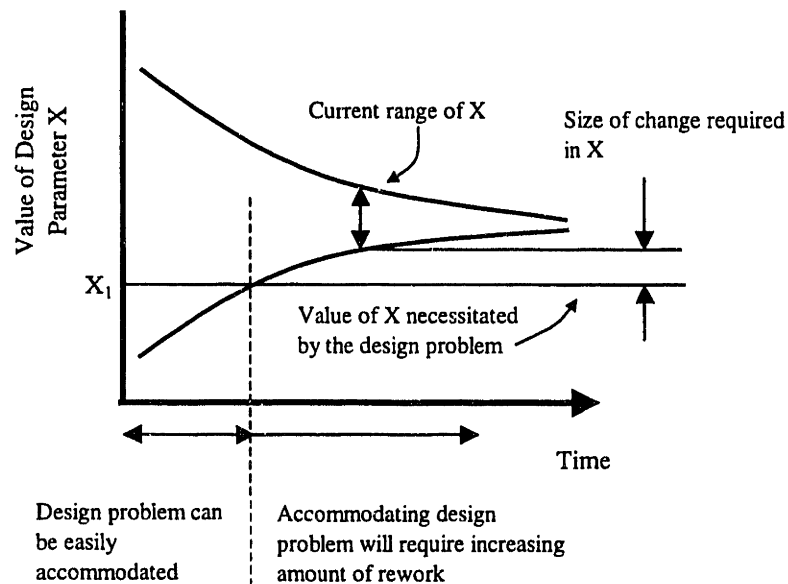


Figure 2: How impact of a design problem grows with time

There could be many reasons for this late discovery of problems, but the tendency to hold back communication of a potential problem, and inadequate schedule update frequency, are two possible explanations.

The second form of poor communication was poorly defined information exchanges. The standardized product development process at the company identifies and defines a set of “give-gets” between activities. These are agreed deliverables between an upstream and downstream activity. For example, in vehicle development the Design Studio agrees to release (give) vehicle surface dimensions within an agreed range of accuracy to Manufacturing (get) at a specified point in the development timeline.

However, despite the documented exchanges, incidents where differences in interpretation of a give-get, or where one or both activities were not aware of the give-get can, and do occur. In either case, the probability of delay is increased. In the former case, the probability of rework for one or both of the activities is increased, depending on which activity requires input from the other. In the latter case, when the date that the downstream activity expects the information from

the upstream activity arrives and the information is not available, due to the importance of keeping the project on schedule, it is likely that the downstream activity will proceed with preliminary data. In other words, an overlapping recovery action will be used. Use of this preliminary data increases the risk of rework for the activity using it. So when give-gets are poorly defined, the probability of rework is increased.

A give-get can be considered to have different degrees of definition that could be arranged on a continuum as shown in Figure 3. As a minimum, one of the parties defines and announces what information it requires. Strictly speaking, this is not a “give-get” since it is a one-sided definition, but since this scenario could conceivably occur, it should be considered as one end of the continuum of definition. Moving in the direction of increasing definition, the single party defines both what information it requires and the date by which it is required. The third degree of definition would be agreement on this information and date by the two parties.

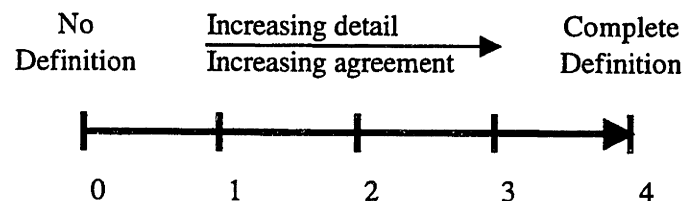


Figure 3: Continuum of give-get definition

Beyond this, agreement on these intermediate information transfers by both parties could be considered the highest degree of definition. This scale of give-get definition will be used in the implementation of process improvement described in section 4.2, but it is outlined here to illustrate the many ways in which poor communication can occur, even if so-called give-gets exist.

Regardless of the reason for it, poor communication increases potential rework, which increases expected project duration. In less-than optimal communication, the requirement for rework is created by the fact that related activities use “old” information to perform their work, often without even knowing how unstable the information may be.

- Causes of Design Churning: Poor Activity and Resource Planning -

Two fundamental planning aspects of any project that can directly affect the potential for delay are activity definition and resource planning. Errors or omissions in either of these plans can obviously add delay to the project. Thus having a reliable process to aid in the development of these plans is important. To this end, the company makes use of standard development schedule that details activities for a generic program, and resource planning models. However, there are two particular challenges associated with the complex, large-scale nature of projects such as vehicle development: determining the appropriate level of schedule detail, and dealing with the many give-gets (deliverables) between activities. These two challenges complicate activity and resource planning respectively and are discussed in turn below.

Activity Planning

After establishing a work breakdown structure (detailed project deliverables) corresponding to the scope statement of the project (objectives and major deliverables), the next step in project planning is to define activities to achieve the deliverables (PMI, 1996). The definition of these activities is closely related to the level of detail tracked in the schedule. A schedule's level of detail is characterized by the smallest aggregate of the design that the schedule tracks, i.e. the schedule may track the systems, subsystems, and/or components of the design. A schedule that tracks the components of a design has many more items than one that tracks only the systems of a design and thus has a higher level of detail.

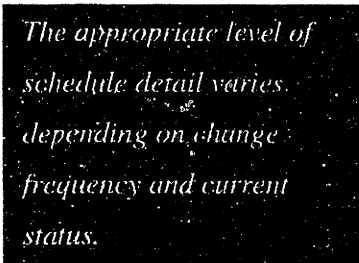
With respect to determining the appropriate level of detail, the challenge is to balance the desire to avoid missing an activity – and subsequently having to change the schedule to accommodate it – with the demands on workplanning resources that increase with the schedule's level of detail. Poor activity planning misses important activities and/or expends workplanning resources (time required to monitor and update schedules) to track unimportant activities.

In a small project, it would be relatively simple to track activities down to the component level of detail. In a large project however, the sheer number of components makes this much more difficult – given a limited number of workplanning staff. Thus there is a non-trivial question as to what the appropriate level of detail should be. If the lowest level schedule is maintained at say the sub-system level, the team risks the possibility of missing or forgetting about an important component-level activity.

However, tracking all component-level activities in a vehicle program could quickly become an overwhelming amount of work. At the company, schedules that track activities down to the system level already involve 1500 activities! Here the concept of rolling up schedules of increasing levels of detail becomes useful as a way of managing this complexity. However, although this reduces the complexity for the schedule manager at each level, it is still true that the greater the level of detail, the greater the amount of resources required to manage the schedules.

If the requisite resources cannot be provided, the poor communication problems described above become more likely, increasing the probability of project delay. For example, if workplanners are asked to manage detailed schedules, the time required to track the large number of activities may result in a lower schedule update frequency which discourages early problem recognition, which in turn increases the impact of a potential delay. Or, the large number of activities may mean that the definition and coordination of “give-gets” suffers, which increases the probability of delay.

Given that the objective of program management is to ensure the timely and cost-controlled execution of a project – as opposed to tracking the lowest level of activities just for the sake of tracking them – the solution to the level of detail dilemma should be to identify appropriate levels of detail for each group of activities. This appropriate level of detail may differ for the various types of activities and for the different phases of the project. For example, early in the project, it may be relevant and useful to track the development of each component of a system to ensure that all the necessary components are developed and that no “last minute additions” arise to cause rework. Later in the project, say in the manufacturing phase, it may be sufficient to track the state of the system as a whole.



The appropriate level of schedule detail varies depending on change frequency and current status.

The schedule’s level of detail and closely associated parameter of update frequency, have a large effect on its credibility and usefulness. Low detail and infrequently updated schedules increase the probability of missing a required activity. They also decrease the schedule’s sensitivity to early delays, and hence the team’s ability to react to the problem before it grows.

On the other hand, too much detail and/or very frequent updates result in a constantly changing schedule that is overly sensitive to minor changes in activity status, or worse, a schedule that

never gets updated because the task is so onerous. A primary role of program management is to help the project subgroups develop a schedule with an appropriate, useful level of detail.

The determination of this appropriate level of detail depends on at least three factors: the team's relevant project experience, the frequency of change for the activity, and the impact of the consequences of missing the activity. The team's relevant experience depends on the past experience of the team members, the stability of the development process definition, and the similarity of the project to past projects. The frequency of change for an activity depends on the evolution of its data, the sensitivity of its relationships to other activities (see section 1.1) and the particular phase of the project. The consequences of missing an activity are two-fold: in addition to directly increasing the schedule duration, it is very likely that accommodating the missed activity will force rework on dependent activities. The impact of these consequences obviously depend on the sensitivity of activities related to it, and the how far into the project the missing activity is discovered.

While it would be reasonable to assume that level of detail should increase with decreased relevant experience, and increased potential impact, the relationship with frequency of change is not clear. The desire to provide early warning of delays, while avoiding inordinate amounts of work to update schedules suggests that the level of schedule detail should be matched with the amount of possible change. Thus the optimum level of detail will be different for different activities in the project. In activities with short iteration cycles or many input activities, the level of detail and update frequency should be higher than in activities with longer iteration cycles or few inputs.

Note however, a reasonable counter-argument would be that increasing the level of detail in an environment of constant change means that the schedule will be in constant turmoil and will lose value unless the resources to continually maintain it can be afforded. So, given this dilemma and the fact that the assessment of these factors is not straightforward, it is not clear that there are obvious rules for determining the appropriate level of detail. Rather than attempting to fit a general rule, one solution could be to base the amount of schedule detail for different activities on historical occurrences of missed activities: a high incidence of missed activities would suggest the need for greater detail.

Resource Planning

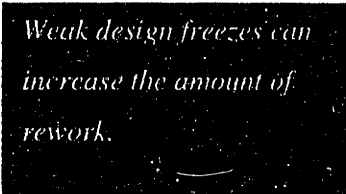
With respect to resource planning, the challenge presented by a large, complex project such as a vehicle development program, is to coordinate priorities and resource availability between activities that typically have numerous give-gets (deliverables) with other activities. The potential for rework in a product development project complicates this matter even further. Poor resource planning results in multi-tasking and mismatched priorities between dependent activities.

Although the members of a “heavyweight” project team are dedicated to one project, the numerous relationships between activities makes managing priorities and resource availability within the program team more than a trivial task. A single activity can easily have multiple give-gets that it is responsible for providing, which means that the group performing the activity must be working on more than a single task at once. Goldratt demonstrates that this multi-tasking is inefficient because it delays all of the tasks and because the switching time reduces the productivity of the group (Goldratt, 1997). And, given that the group is multi-tasking, there must be a priority established for these tasks. If two activities that must exchange deliverables do not agree on priorities, the likelihood of that exchange occurring on time is low.

An additional factor that makes resource planning difficult in vehicle development projects is the increasing reliance on outside suppliers. When an outside supplier is involved, the likelihood of delay is even higher since in addition to this multi-tasking, the program has a lesser degree of control over the group performing the activity. While the supplier relationships being formed in today’s development projects could be considered stronger and more cooperative than in the past, the project must still deal with the probability that the supplier will have restricted flexibility to shift jobs around, due to its objective of maximizing its utilization of capacity.

And even if compression and prioritization of a particular job is possible, this will almost certainly come at a premium. Thus it becomes extremely important that the potential for delay of an activity that feeds an activity to an outside supplier is low. To make matters even more uncertain, the possibility also exists that despite advance planning, the supplier may not be ready when the program is ready.

- *Causes of Design Churning: Weak Design Freezes* -



Weak design freezes can increase the amount of rework.

Another factor that contributes to the probability of rework – and in turn project delay – is poorly defined design freezes. A design freeze is a declaration by the team that particular design parameters will be finalized in their current state. The function of a design freeze is to constrain aspects of the design to permit the progress of related downstream activities. For example, the team may decide to freeze the outside dimensions of a particular component shortly after achieving what appears to be a working design. Finalizing these allows designers of related components to position their components around it and design any attachments.

Allowing a design freeze to “drift” beyond its announced date can increase the probability of rework in the following way. When the freeze is announced the related downstream activities begin their work based on the frozen design information. If the upstream design work continues, the potential for rework of the downstream activities is created since these downstream activities may now be using invalid design information.

In the case that the downstream activity is dependent on the upstream activity, the worst case associated with a change in the upstream activity would be that the downstream activity must repeat all the work since the declared design freeze. If we make the reasonable assumption that the downstream activity has an increasing sensitivity to change, the amount of rework will increase with the length of time by which the freeze is allowed to drift. This is the same reasoning used in explaining the effect of late problem discovery in section 2.3.1.

However, if the upstream and downstream activities are interdependent (Eppinger et al., 1990), the downstream activity could cause changes for the upstream activity. Drift in a design freeze creates the potential for an extended number of iterations between the activities. One would expect that these iterations would gradually converge on a final solution. In the interdependent case though, if no agreement is established between the activities as to what should be frozen, there is a potential, however small, that the iteration will not converge.

Build on the example given above. At the agreed freeze date the upstream activity freezes the outside dimensions of its component X. The downstream activity then proceeds to design the

attachment of its component Y to component X. Now say that for some reason the design of component X must be modified. If designers of X heed the freeze and are able to modify the design in such a way that maintains the attachment point used by component Y, Y designers will not be affected. If however, they do not heed the freeze, the designers of Y will be forced to rework their design.

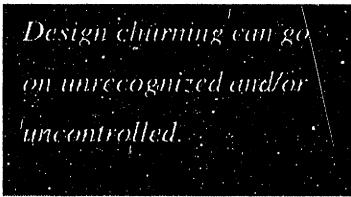
If in reworking their design, the designers of Y unknowingly cause the need for a change in X (either for lack of understanding, or because X is still changing) that may affect the attachment point of Y on X, the potential for non-converging iteration is created. While this example may seem like an extreme case – since it is difficult to believe that designers of X and Y would not quickly recognize and address the problem – its existence should be recognized. This is because in the context of a complex project with many activities that each have multiple interdependent relationships, the problem can be obscured.

The argument is not whether a design freeze is adhered to or not. Sometimes this is not possible. Thus the real issue is how to determine what design freezes can be adhered to and would be valuable to set. In either the dependent or interdependent case described above, the assumption by the downstream activity that the upstream activity was finished (and therefore that the transferred information was accurate), creates a higher potential for rework. If the downstream activity did not make that assumption it would be forced to communicate with the upstream activity to get up-to-date information.

If a design freeze is not going to be adhered to, the argument can be made that the project is better off not declaring it, thereby discouraging related downstream activities from proceeding with unstable information. The value of a design freeze should be assessed by the tradeoff between constraint on the design and the effect on progress along the critical path of the project.

- Consequences of Design Churning -

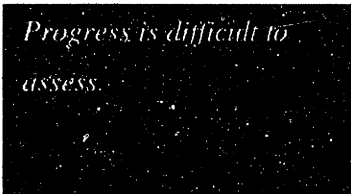
When design churning (avoidable rework) occurs engineering resources are being expended without realizing the desired project progress, i.e. the rate of increase in the degree of design definition slows or regresses. The net effect of design churning is a longer activity duration compared to a design activity without it. In turn, longer activity durations increase the potential for delay in completion of the project.



Also, when the project suffers from design churning, it can be difficult to discern whether design changes are contributing to progress (increased design definition), or whether they are due to one of the factors that cause design churning. This allows these factors to continue to add rework and drive iteration. Moreover, the development process at the company appeared to lack a sufficiently proactive trigger to stop this churning. It is not until later in the program that the increasing risk of missing the project deadline became a powerful trigger.

Design iteration is a necessary and inherent aspect of the product development process. However, design churning causes avoidable rework. This unnecessary, unexpected rework can easily slow the development process.

2.3.2 Poor Progress Measure



The assessment of activity and project progress is important to detecting potential delay problems. This allows any required remedial action to be taken as early as possible, before the impact of the problem becomes serious. However, for the reasons discussed below, the current activity and progress measures are not entirely ideal in that they are poorly defined, do not provide a composite, holistic assessment of status, are not leading indicators, and/or do not account for potential rework.

- Causes Contributing to Poor Progress Measurement -

The two prevalent quantitative status measures in use at the company were “negative float” and “percent complete”. “Float” or “slack” refers to the amount of time before an activity becomes part of the critical path of the project and is determined by estimating activity finish dates. “Negative float” then implies that the activity is on the critical path and is lengthening it beyond the original project completion date. Determining this metric for each activity, the project management strategy is to concentrate efforts on completing activities with negative float as quickly as possible to avoid further delaying project completion.

There are four shortcomings of this metric. The first has to do with defining the activity. Assessing the status of an activity requires definitions of the start and finish of the activity, and some method of estimating the finish date of the activity based on interim progress. While defining the start and finish of an activity may seem to be a simple and straightforward task, further consideration reveals that with dependent activities, it may in fact not be trivial.

When a successor activity depends on a predecessor activity, if there is not an agreed definition of what constitutes completion of the predecessor, a potential for delay is created. A primary role of program management is to help the project subgroups identify, define and communicate their “give-gets” (deliverables) between other each other, i.e. to work out agreed activity start and end definitions.

The estimation of the activity’s finish date is even less trivial. There are in general two approaches: determine the projected finish date by estimating the current percent complete, or estimate the projected finish date directly. Due to its apparent simplicity, the percent complete approach is often used despite its significant drawback. Consider as an example, the development of a product where the percentage of total required drawings is used to assess percent complete. While this is a relatively straightforward determination, it can be very misleading. It implicitly assumes that progress is linear, i.e. if 50% of the drawings were completed in 2 weeks, the estimated finish date is 2 weeks hence. The alternative approach, explicitly estimating time required to finish an activity, overcomes this problem, but requires more thought on the part of the people performing the activity.

Secondly, “negative float” a metric associated with each activity – it is not a composite metric that can be used to report the status of the project as a whole, i.e. adding or otherwise combining the negative floats for each activity is not meaningful. Thirdly, “negative float” is not a particularly leading indicator of activity delay. Because it focuses on activity starts and finishes, the project manager cannot use interim status to identify the need for remedial action. Thus there is a tendency to wait until it is more apparent that the estimated activity finish date is late. Finally, this metric does not account for potential rework. If iteration is required the duration of an activity is increased. Ignoring the potential for this clearly reduces the accuracy of the evaluation of project progress.

Perhaps the most intuitive and traditional project progress metric is “percent complete”. For each activity, intermediate deliverables are defined a-priori. The percent complete of an activity at any point in time is determined by assessing what proportion of those intermediate deliverables has been achieved. Each activity is also assigned a weighting a-priori, based on the activity’s significance to the project (the activity weights sum to one). The project status is then the weighted sum of each activity’s percent complete.

This metric overcomes the second shortcoming of “negative float” since it is a weighted sum of the percent complete of each activity. In this sense “percent complete” is a composite yet “divisible” metric. It gives a holistic assessment of project status in a single value *and* it can be easily dissected to show the reasoning behind that value. By capturing project status in a single quantity, progress can be tracked over time.

This metric is also more of a leading indicator of delay than “negative float” because it monitors the achievement of defined intermediate deliverables. The definition of intermediate deliverables provides a reference for assessing how much time is still required. Then, by comparing actual percent complete to scheduled percent complete, this metric can provide early warning of delays.

Although it still suffers from the difficulty in defining the start, end and intermediate deliverables of an activity mentioned in the discussion of “negative float”, the primary shortcoming of “percent complete” is that it does not account for potential rework in its measurement of progress.

Combined with the uncertainty in project progress due to design churning, the inability to accurately measure the completion status of an activity creates a large potential for delay.

- Consequences of a Poor Progress Measure -

Despite the fact that the shortcomings of these project status metrics contribute to delay, they are commonly used. However, a poorly defined, non-composite or late progress metric results in a misleading assessment of project status. In an iterative design project, the potential for rework adds to this inaccuracy. This inaccuracy can promote delay by not providing early warning of potential delay, or by allowing design changes to proceed under the false pretense that there is adequate time in the schedule.

2.3.3 Reactive Schedule Recovery

Recovery planning takes time that is not accounted for in the schedule.

There are two generally accepted options that can be employed to recover from a schedule delay: compression and overlapping. By activity compression, we mean that overtime or other measures are taken to reduce the standard, scheduled duration of an activity. This is a very typical and commonly accepted practice.

The primary cost associated with this recovery option is the overtime premium paid to those involved in the activity. However, there is the possibility of an additional indirect cost associated with compression. The risk in using compression is that in an effort to hurry progress, the potential of making design mistakes is increased. For example, detailed analyses and tests may be compromised, which in turn increases the probability of rework.

Of course, use of this option assumes that the resources (both people and funding) are available, which may or may not be the case. While it is relatively common for internal staff on the team to work whatever overtime is required, there is no definite obligation. And if the activity involves external resources, the ability to use compression may be further restricted by obligations that the supplier has to other customers.

Perhaps an even more important assumption is that the activity can actually be compressed. While longer days and more resources may reduce the time required for activities such as drafting and issuing part drawings, they are not likely to be effective in reducing the time required for a 60-day fatigue test that is already run 24 hours-a-day. Thus there is a limit to the amount by which an activity can be compressed, and it is not likely (nor desirable) to rely solely on compression as a recovery option.

The other option for schedule recovery is to use activity overlapping. This approach involves starting a downstream activity prior to the completion of the upstream activity. Depending on the amount of overlap, this tactic offers the potential to keep the project on schedule. However, the risk is that for the duration of the overlap, the downstream activity is subject to rework caused by changes in the still active upstream activity.

As described above in the discussion of weak design freezes, the amount of rework depends on the nature of the relationship between the activities. If it is only that the downstream activity is dependent on the upstream activity, the maximum rework for the downstream activity will be the amount of overlap with the upstream activity, i.e. in the worst case, the downstream activity will have to repeat all the work since it started. If the activities are interdependent, then there is not necessarily a limit to the number of iterations. The cost associated with overlapping is the cost of expending resources on rework. There may also be other costs such as facilities and material costs.

- Causes of Reactive Schedule Recovery -

By increasing the probability of delay, design churning and the difficulty in quantifying activity status make late, reactive schedule recovery action more common than proactive action to avoid the delay in the first place. While it is clear that the need for schedule recovery action is undesirable, it is nevertheless highly probable that it will be required at some point in the project. One way to capitalize and learn from this however, would be to capture why the delay arose and how it was handled. This information can be used for planning of future programs. The historical requirements for recovery, and how effective the recovery actions were, are valuable aids in deciding appropriate recovery strategy when delay does occur.

- Consequences of Reactive Schedule Recovery -

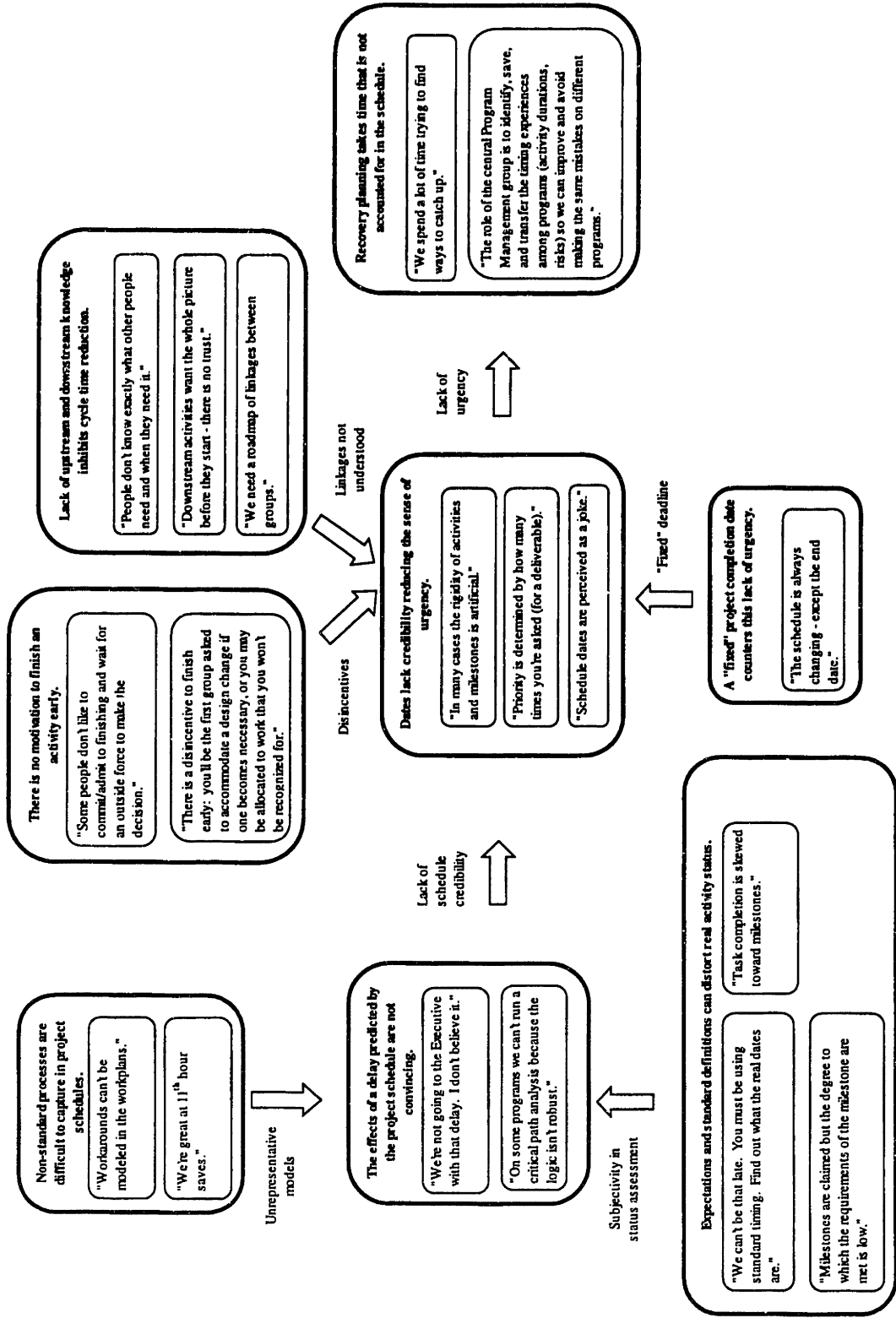
The consequence of having to continually expend time and money to recover from late activities is the reduced opportunity to take proactive action. But it is proactive action (such as taking steps to prevent delay and identifying opportunities to reduce activity durations) that is the key to reducing development lead time.

2.4 Organizational Issues

Figure 4 summarizes the key interview comments related to the second key question, and the conclusion drawn from the cause and effect relationships identified between them.

Figure 4: Organizational Issues

What are the problems/issues related to supporting the workplanning function?



Lack of schedule credibility creates a lack of urgency that perpetuates the need for reactive schedule recovery.

The issues in Figure 4 are related to organizational dynamics. The following sections describe this conclusion in more detail, highlighting the issues inhibiting control of development lead time that were raised during the interviews. More specifically, the two parts of this conclusion will be discussed: low credibility schedules and lack of urgency. For each of these topics, the specific issues are described, the causes identified, and the consequences explained. The **white on black** headings refer to the box titles in Figure 4.

2.4.1 Low Credibility Schedules

The effects of a delay predicted by the project schedule are not convincing.

The value of a schedule is two-fold. In the first instance, the development of a schedule forces the project team to identify and sequence the activities required to achieve the project deliverables. Secondly, the schedule provides a basis for project control, i.e. it tracks activity progress and predicts effects of delay. While most would agree with the value of the first function of a schedule, the study found that people were less convinced of the efficacy of the predictive function. It seemed that among the team there was a lack of convincing links between delays in low level activities and their effect at the project level.

- Causes Contributing to Low Schedule Credibility -

The difficulty in conveying potential or real delay problems was attributed to two factors: the existence of “workarounds” for delay, and subjectivity in measuring activity status.

Non-standard processes are difficult to capture in project schedules.

Workarounds refer to the actions that the program can employ to recover from a delay in the schedule. In general there are two alternatives: overlapping and compression, discussed above in section 2.3.3. One or both of these “workarounds” may be used to react to a delay in the schedule, and although they are not explicitly modeled in the project schedule, management has assumptions about their ability to recover from delay.

This may be an explanation for management’s apparent tendency to discount the seriousness of delays predicted by the schedules. The acceptance of these recovery actions then perpetuates their use.

However, as discussed in section 2.3.3 the risks associated with use of these recovery actions can increase project duration.

Expectations and standard definitions can distort real activity status.

The subjectivity in assessment of the status of a given activity is another source of low schedule credibility. Differences in definitions and interpretations of activity completeness add uncertainty to the schedule. The study also found that standard as opposed to actual estimates of activity duration, and subjective interpretations of milestone definitions make it difficult to determine the status of a given activity or milestone.

- Consequences of Low Credibility Schedules -

The consequence of the low credibility of project schedules is the lack of urgency to complete the activities in them.

2.4.2 Lack of Urgency

Dates lack credibility reducing the sense of urgency.

The lack of credibility in the project schedule contributes to a lack of urgency in current and upcoming activities. If the project schedule is almost continually being changed, the pressure to work towards the latest deadline is reduced.

Consider two problems that may arise when using a schedule that cannot reliably predict the effect of a delay in a current activity on the project. In one case the schedule may actually point out a legitimate or real delay to the project, but because the schedule historically lacks credibility, the warning is ignored. Or, in another case the schedule may be missing an activity or link between activities and fail to provide the required warning of potential delay. Whichever the case, the value of the schedule is questioned and there is a tendency to ignore it, “dulling” the team’s motivation to recognize potential delay.

The lack of urgency to work toward the scheduled deadline is particularly true if there are no visible consequences of missing an activity or milestone completion. Thus when the project schedule and milestones lose their credibility, the urgency to complete an activity is determined by pressure from other activities depending on it. This pressure is a function of the intensity and frequency with which requests for completion of the activity are made.

- Causes Contributing to the Lack of Urgency -

In addition to low schedule credibility, there were three other important factors that affect the amount of urgency to complete an activity: lack of upstream and downstream knowledge, lack of motivation to finish an activity early, and a fixed project completion date.

Lack of upstream and downstream knowledge inhibits cycle time reduction.

In any development project, but particularly in one as large and complex as a vehicle program, each group will have its own priorities, many of which will be different from those of its upstream and downstream groups. It is difficult to justify urgency unless these priorities are understood. While the company has identified and documented many give-gets (deliverables) between activities, there still seemed to be some uncertainty on the part of the “giving” activity as to why the information was important, and exactly what information was required. This uncertainty reduces the motivation of a given group to provide a deliverable required by another group.

As an example of this uncertainty, phases of information transfer with increasingly narrow ranges for dimensions have been established, but these are general or “blanket” specifications. Further discussion among the upstream and downstream groups might reveal that the downstream group really only requires “tight” specifications for a few particular aspects of the design. The upstream groups would therefore be wasting time by holding back the required information until they had refined specifications for all the design aspects.

Without a detailed and agreed understanding of the information to be passed between activities, there can be a tendency for downstream groups to wait for complete and detailed information before proceeding with their activity. This tendency is even stronger if their experience with the upstream group has been that the upstream group “always makes changes”. However, waiting for complete information opposes the overlapping of development activities, the goal of which is to reduce the total development time.

Overcoming this opposition requires not only that the upstream and downstream groups define and agree on detailed information to be transferred and in what phases it will be transferred, but also on a cultural change that makes rework acceptable, and to some extent expected.

There is no motivation to finish an activity early.

Besides the apparent lack of convincing motivation for urgency, it was surprising to find that in some cases, there can actually be a disincentive to finish early. There were two potential reasons to discourage finishing an activity early. By committing to a solution, a group might believe that it forfeits its opportunity to continue optimizing, i.e. the “perfectionist syndrome” dictates that a group use as much time as it is given. While this optimization may contribute to a performance improvement, the improvement should be weighed against the value to the customer and the consequence of possible changes and rework for downstream groups.

The second disincentive to finish early has to do with work scope and responsibilities. A group that finishes early is more likely to be forced to accommodate changes since it has extra time. Since accommodating changes typically means rework, there is no motivation to finish early. This phenomenon suggests that a review of the incentive structure might be useful.

A “fixed” project completion date counters this lack of urgency.

The desire to avoid delaying the original project completion date counteracts this lack of urgency. As the threat of missing the date becomes more apparent, priorities become more established and schedule recovery actions will be taken. Unfortunately, this urgency “trigger” does not become active until later in the project when delays have become real problems.

- Consequences of Lack of Urgency -

The consequence of this lack of urgency is an increased probability of delay. This in turn predisposes the program toward reactive schedule recovery action rather than proactive action to avoid the delay in the first place. As noted above in section 2.3.3, a program that must spend all of its time recovering from schedule problems has little time to devote to proactive schedule improvement which a key to reducing development lead time.

2.5 Chapter Summary

A voice-of-the-customer analysis conducted among various vehicle development programs at the company found both process-related and organizational issues that inhibit the control of development lead time. From the perspective of the development process itself, it was concluded that design churning and

the difficulty in accurately measuring project progress predispose the project toward reacting to schedule delay. Design churning was defined as a situation where factors over which Program Management has some degree of control cause increased design iteration or rework. Four causes of design churning were identified: changes in design requirements, poor communication, poor activity and resource planning, and weak design freezes.

From an organizational perspective, it was concluded that low credibility project schedules contribute to a lack of urgency among the team, which again predisposes the project toward reacting to schedule delay.

The focus of Chapter 3 is the definition of improvement efforts based on the root causes of these issues.

3. Defining the Problem: Defining Improvement Efforts

Having identified the key issues inhibiting the achievement of control over development lead time, the next step was to determine any links between these issues. An understanding of these links then allowed the root causes of these issues to be identified and appropriate improvement efforts to be defined.

3.1 Linking the Process and Organizational Issues

The voice-of-the-customer analysis discussed in Chapter 2 uncovered two phenomena that frustrate control over product development lead time:

- Design churning and the difficulty in accurately measuring activity progress predispose the program toward reacting to schedule delay – as opposed to taking proactive action to avoid delay.
- Lack of schedule credibility creates a lack of urgency that perpetuates the need for reactive schedule recovery.

The obvious commonality between these phenomena is the predominant need for reactive schedule recovery. However, by dissecting the problem and making some logical extensions it was found that the two phenomena are quite closely linked. Figure 5 illustrates the extensions and how the phenomena are related in the form of a causal loop diagram. In the diagram, the arrows indicate cause and effect, i.e. the base and head of the arrow. The +/- signs indicate the “polarity” of the cause and effect, i.e. “+” indicates that increasing the cause increases the effect while “-” indicates that increasing the cause decreases the effect. With respect to the distinction between process and organizational issues made earlier, the process-related issues are shown with solid arrows and the organizational issues are shown with dashed arrows.

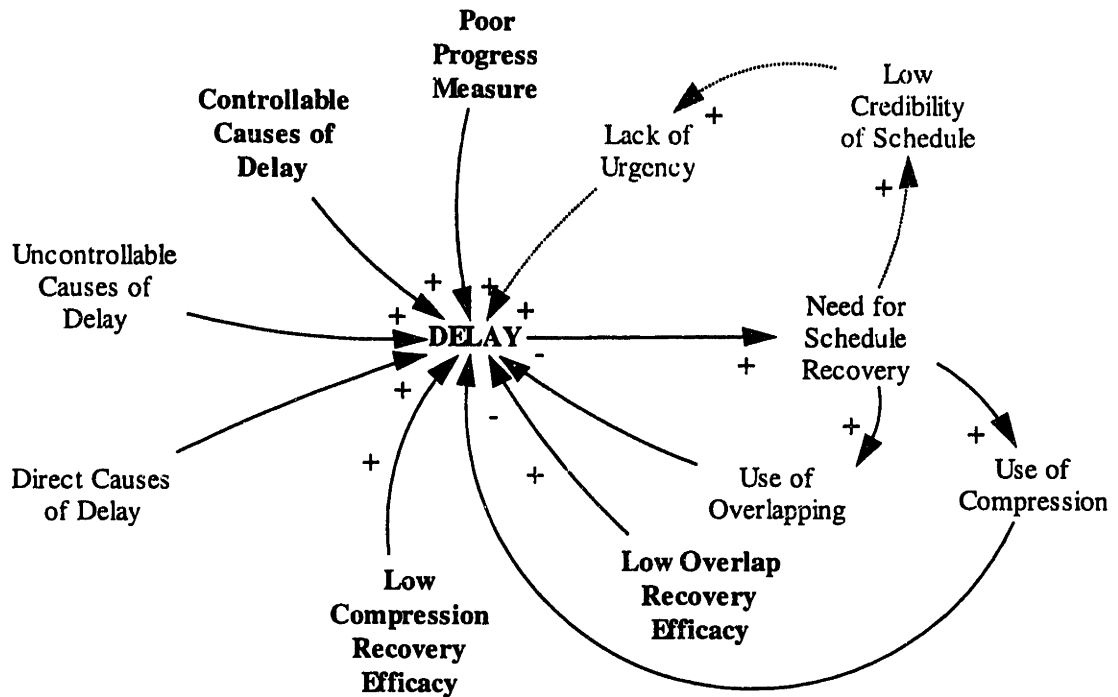


Figure 5: The causes and consequences of delay

Three key extensions were made that allowed these phenomena to be related. Given that schedule recovery was a common point, this was a logical point from which to extend. In asking the question, “What generates the need for recovery?” the obvious answer was schedule delay. Thus the first extension was to define delay as a variable of interest. The second extension was made after asking the question “What does schedule recovery mean in practice?” As described above in section 2.3.3, when an activity is delayed because of an incomplete predecessor activity, there are two typical recovery options: activity compression and activity overlap. Thus the second extension was to include these alternative recovery actions and treat the effectiveness each as modifiers of the rate at which delay is reduced.

The third extension was to define three general causes of delay:

- “uncontrollable” or natural probability for design iteration
- “controllable” probability for rework, and
- a misleading or lacking measure of project progress.

Specific examples of these types of delay are outlined in section 4.2, but the definition of these types of delay provides a basis for defining the objective for improvement of the development process. Separating controllable and uncontrollable sources of delay permits design churning to be defined as the delay over

and above the expected duration of an “efficient” project, which is not afflicted by any controllable causes of delay. These concepts are explained in further detail below.

- Design Churning Defined -

The distinction between “controllable” and “uncontrollable” factors is made on the basis that a development process with dependent or interdependent activities will always require some degree of rework due to the information dependencies between design activities. Let us define an “efficient” development process in which there are no changes in design requirements, communication between dependent activities is rich and timely, no important activities are left out of the schedule, and design freezes are defined and adhered to. In this case rework is solely due to the iteration inherent in engineering design. This inherent rework arises from the trial-and-error element and the “chicken vs. the egg” problems in engineering design and cannot be directly controlled by program management.

In the design churning situation however, factors within the control of program management (changes in design requirements, poor communication, poor planning, weak design freezes) increase rework causing design progress/definition to slow or regress. These controllable factors were discussed in section 2.3.1, but there are two general points that should be made here.

While the objective is clearly to reduce as much as possible the rework caused by these factors, the motivation behind the associated changes is not necessarily undesirable. For example, a management-driven change in design requirements may better position the product to a change in the market. Hence the potential conflict between reducing development lead time and maintaining flexibility in design.

Secondly, while these factors were classed as “controllable” by program management, the degree of control varies and is not by any means complete. For example, while the promotion and coordination of communication among the team is a primary role for program management, successful communication also relies on the willingness of the team members to share information. Similarly, while program management has input in the decision of whether or not to accept a design change, market requirements may prevail as a more significant force.

The important points are the distinction between controllable and uncontrollable factors and that while some of the iteration in the development process is due simply to the dependencies between activities, program management has the ability to affect the amount of potential rework beyond this inherent

iteration. From a process improvement perspective, it would be valuable to know what the active controllable factors are, how much delay they add to the project, and where in the project duration they become active.

This knowledge provides focus for improvement efforts. But, unlike process improvement in manufacturing where the objective is to eliminate variation in process output due to controllable causes, in product development we should seek to: i) eliminate rework due to controllable causes that do not have a benefit associated with them, and ii) understand and consciously control the causes that do have a benefit associated with them.

The Relationship between the Extended Phenomena

Summarizing the discussion of this section so far, the insights gained from the voice-of-the-customer analysis suggest that the schedule-related problems in a project can be described by two causal loops:

Controllable and uncontrollable causes of delay result in the need for schedule recovery that can take the form of compression or overlapping of activities. These recovery actions may not be sufficient to make up the delay, perpetuating the need for recovery. The lack of a progress measure that takes the possibility of rework into account and predicts delay is another cause of delay.

The seemingly continual need for recovery reduces the credibility of the project schedule, which in turn causes a lack of urgency in activity execution and/or sub-optimal prioritization of activities. This lack of urgency increases the probability of delay, which in turn increases the need for recovery, creating a second loop.

The first loop consists of factors in the execution of a project that can be affected by process design and policy, i.e. actions can be taken to modify them. The second loop describes a “second order” response to the first loop. It is “second order” in the sense that by reducing the need for recovery in the first loop, the effect of the second loop can be eliminated, i.e. the second loop only exists because the first loop exists.

3.2 Addressing the Problem: 3 Improvement Efforts

Given this distinction between first and second order loops of delay, the most effective approach to process improvement will be to concentrate efforts on “breaking” the first loop. Reducing the occurrence of controllable delay in the primary loop reduces the need for recovery, which in addition to reducing the propagation of delay in the primary loop also lessens the delay added by the second loop.

Then, in order to increase control over development lead time, three improvement efforts targeted at the primary loop were proposed:

- Track and address the controllable causes of delay to reduce design churning
- Optimize effectiveness of schedule recovery
- Develop a more representative measure of project progress

In any improvement effort, the selection of a metric to monitor the effort is key. In attempting to quantify the effects of the loops of delay on a development project, activity delay (actual duration minus planned duration) presents itself as an obvious metric. In other words, as the causal loops illustrate, delay in an activity can be used to gauge the degree to which the loops are active, and to determine whether improvement efforts have had any effect.

However, before accepting this as the metric to drive improvement of the product development process, there are several criteria that this choice should be checked against. In his discussion of process control, Reinertsen suggests that a metric should possess three characteristics: simplicity, relevance, and be a leading indicator. By simplicity, he means that a metric should be easy to generate and understand. Ideally he says, a metric should be “self-generating”, i.e. not require extra work outside the normal course of business (Reinertsen, 1997, p. 203). A test for ease of understanding is whether the metric means the same thing to all those who are affected by it and/or affect it. Activity delay fulfills this criteria in that it is easy to determine, is an inherent part of program management, and subject to the agreement of what deliverables constitute the completion of an activity, is universally understood among the team.

Secondly, a metric that is well-suited to an objective must be relevant to that objective. As obvious as this may seem, cases where the use of a metric resulted in unexpected results are certainly not uncommon. Reinertsen points out that the selection of relevant metrics should be driven by the economics of the business, i.e. the metric should be relevant at the business level (Reinertsen, 1997, p. 197). In the automotive industry, while the precise dollar cost is difficult to determine, the significance of the consequences of a delay in project completion are well accepted.

As a reference point, Clark et al. estimate the marginal cost of development lead time for a vehicle to be at least \$1 million per day (Clark et al., 1987). For *activity* delay to be relevant however, the activities for which delay is tracked must be on the critical path, i.e. a delay in the activity implies a direct delay in the project.

Another test of relevance is the degree of control over the metric possessed by the people being measured. People are more motivated to control a metric if they believe they are empowered to control it. While this would suggest locally-focussed metrics, it is also true that such a focus could compromise global optimization (Reinertsen, 1997, p. 204). In this regard, Reinertsen suggests a strategy that blends local and global metrics. Thus in measuring activity delay, the activities for which delay is tracked should be selected and defined in such a way that there are defined deliverables and clear accountability.

Also in regard to relevance, in order to make the process improvement applicable to future projects, the activities for which delay is tracked should be chosen so that they are common to all projects, although their durations may have to be normalized to account for projects of different scales. Alternatively, the activities could be selected in a modular way so that the knowledge for them can be “mixed and matched” to apply to any program.

Thirdly, for real-time control purposes, an effective metric should be a leading indicator of performance, i.e. it should aid in the prediction of future performance, as opposed to reporting on past performance. Having said this though, lagging indicators are generally more accurate (Reinertsen, 1997, p. 204). Delay is inherently a lagging indicator. However, in this case since the objective is process improvement, the report of an actual delay is more relevant. And note that delay in one project has a predictive use in future projects.

Having identified the three improvement efforts and a process improvement metric, the details of each improvement effort will now be discussed.

3.2.1 Track and Address Controllable Delay

The first step toward the objective of achieving the ideal, efficient development process described in Chapter 1 is to make the delay caused by controllable factors visible, i.e. by measuring it. Only then can steps be taken to reduce this delay. In addition to making these controllable delays visible, measuring them also provides a reference point to assess any improvement efforts against. Despite this seemingly obvious point, Reinertsen notes that it is surprising how many companies do not collect data on activity durations (Reinertsen, 1997, p. 59). Section 4.2 details a proposed tool for tracking and analyzing activity delays, but briefly it involves the identification and measurement of delays in activity starts and a method for determining the root cause of the delay.

Once delay data has been collected, it can be used for two purposes. The consequences of a delay in an activity to its related activities can be used to help plan and evaluate decisions in future programs. More specifically, this data can be used to determine realistic parameters for the project simulation described in section 4.3. In turn, this simulation can be used to evaluate changes in the project schedule and as a basis for determining project progress.

Secondly, analyzing the delay data for root causes allows the determination of which causes are most significant, i.e. a Pareto chart can be constructed. Having identified the significant causes of delay, improvement action can be taken – the emphasis being on controllable delays. The remainder of this section will discuss improvement actions for these controllable delays.

The focus for improvement actions will vary according to the type of controllable delay (changes in design requirements, poor communication, poor planning, weak design freezes). For delay associated with poor communication or planning, the objective is to reduce the occurrence of these delays. To this end, the delay analysis tool in section 4.2 was designed to not only identify these causes, but also to suggest how the delay might have been prevented.

For delays associated with design requirement changes, the improvement objective is somewhat different. Understanding the implications of a design requirement change and weighing the costs (time, resource costs and capital costs) with the benefits of making the change (increased sales and/or profitability per vehicle) is an important key to reducing design churning. While changes to design requirements typically cause delay, the objective is not necessarily to reduce them since they may be necessitated by changes in market demands. Rather the goals should be to: i) make the change and its effects visible to the team, and ii) reduce the effect of the change on the project.

Increased visibility of the effects of the requirement change can be achieved by recording the delay associated with the change and by communicating the need for the change. Recording the delay (what activities were affected and by how much they were delayed) increases understanding of the consequences of the change. This develops the ability to make decisions as to whether or not to accept similar changes in the future, since downstream consequences are now known. By generalizing this experience to avoid design changes driven by lack of knowledge of the consequences and, to the extent possible coordinating these changes, control of development lead time can be improved.

If it is decided to proceed with a change, communication is important. When the reasoning for a requirement change and the anticipated effects of it are communicated throughout the team, the various groups on the team – and particularly those far downstream of the change – are more likely to be motivated to accommodate it. This is much more desirable than the current perception of a random, unconnected stream of changes with no particular motivation, which generates frustration among the team.

Recording the consequences of a change in design requirements also directs efforts to increase flexibility of the design process, or in other words to reduce the effect of a design requirement change on the process. Reducing the effect of a change on completed and on-going activities of a project involves making the design robust to changes, i.e. allowing greater flexibility in design parameters further into the project

For example, if changes in appearance requirements consistently cause delay in the design activity, finding an approach to make the design activity more robust to appearance changes would be extremely valuable. Robust and set-based design are two emerging areas of study in product development that appear to be keys to being competitive in today's environment of rapid change and increased customization. Understanding the effects of design requirement changes, where in the project they occur, why they occur, and how often they occur points to areas where the application of robust and set-based design would be most beneficial in terms of reducing development lead time.

Given these needs, the delay analysis tool in section 4.2 was designed to track and illustrate the consequences of a change in design requirements in one activity to its related activities.

For delays associated with weak design freezes, the objectives are to eliminate freezes that cannot realistically be adhered to, and to identify where a design freeze would be helpful in reducing development lead time. These objectives follow from the discussion in section 2.3.1 where it was proposed that the value of a design freeze can be determined by assessing the tradeoff between flexibility to change the design and the reduction of development lead time.

Given that the primary function of a design freeze is to fix the design so that dependent activities can proceed, key considerations in establishing design freezes are definition and timing. In order to define valuable design freezes, a method for determining which design activities are important in driving subsequent activities is required. Furthermore, it is necessary to determine the optimum level of

completion at which to freeze the design, such that it balances the need for flexibility and the desire to allow subsequent activities to proceed. Finally, the sequencing of design freezes must be planned to minimize rework.

Design freezes are related to the more general concept of activity overlapping. As discussed in section 1.1, Krishnan et al. have proposed a model for determining when development activities should and should not be overlapped based on concepts they call evolution and sensitivity (Krishnan et al., 1997). They recommend that a design freeze (or in their terms, “pre-emptive overlapping”) is most effective in reducing project lead time when it is used between a fast evolution upstream activity and a high sensitivity downstream activity. A fast evolution activity is one where the activity rapidly narrows the range of the design parameter for which it is responsible. A high sensitivity activity is one where the amount of rework caused by a change in upstream information increases rapidly with the size of the change.

To this end, the delay analysis tool in section 4.2 was designed to track those delays associated with “violated” design freezes. Checking the collected delay data for these violated design freezes, the evolution and sensitivity of the involved activities can then be considered to determine if the design freeze is valuable. If the freeze is deemed valuable, the optimum freeze point can be determined using the evolution-sensitivity framework and the delay data relevant to that freeze which has been collected from past programs.

3.2.2 Optimize Effectiveness of Recovery Actions

As the causal loop diagram in section 3.1 illustrates, there are many factors that add delay to a project. However, there are only two ways in which delay is reduced: negative variation in activity durations (activity takes less time than planned), and recovery actions. Given the importance of completing the project on time, relying on negative variation is clearly not desirable. The objective should therefore be to maximize the effectiveness of recovery actions. In other words, given a delay in a particular activity and the options of compression and overlapping, it would be valuable for project management to know which is likely to be most effective in minimizing deviation from the original project completion date, and what the expected costs will be.

Given this need, the delay analysis tool described in section 4.2 was designed to track rework delay associated with the use of these recovery actions. Comparing collected delay data to the amount of

overlap and compression used, the effectiveness of these actions can be determined. This data can then be used to determine realistic parameters for the project simulation described in section 4.3. In turn, the simulation can be used to test different recovery strategies.

3.2.3 Develop a More Representative Measure of Project Progress

As discussed in section 2.3.2, a poor measure of project progress contributes to delay by providing a misleading assessment of project status. This poor assessment of project status is an obstacle to gaining early warning of potential delays and contributes to misguided project control decisions.

Recall that an ideal measure of project progress would:

- Be objectively and unambiguously defined
- Combine the status of each activity into a composite measure to capture project status in a single quantity that can be tracked over time
- Identify activities that jeopardize on-time project completion
- Account for potential rework
- Be a leading indicator of delay

Note that these characteristics follow from the characteristics for effective metrics in general (simplicity, relevance, and leading indicator), discussed above in section 3.2.

The “percent complete” measure of project progress was described in section 2.3.2. While it meets most of these criteria, its primary shortcoming was its lack of consideration for potential rework, which can affect project duration. To address this shortcoming, the project simulation described in section 4.3 can be used to determine more representative estimates of expected activity (and therefore project) durations, i.e. ones that include potential rework. These estimates can then be used to determine a more realistic measure of percent complete.

Even when the potential for rework is accounted for, “percent complete” could be further improved by making it more of a leading indicator of project progress – so that potential delays can be identified earlier. This will be discussed in section 5.3 in the context of further work.

3.3 Chapter Summary

Linking the process-related and organizational issues that inhibit control over development lead time, the process-related issues were found to be the root of the problem. To this end, three improvement efforts were defined:

- Track and address controllable causes of delay to reduce design churning
- Optimize effectiveness of schedule recovery actions, and
- Develop a more representative measure of project progress.

The objectives of tracking controllable causes of delay are: to determine and take action to reduce the most significant causes of delay, and to provide data for simulating projects. With respect to schedule recovery, the objective is to collect data to determine the effectiveness of these actions and to provide data to simulate different recovery strategies. As for project progress, the objective is to improve the current “percent complete” measure of project progress by including the potential for rework.

Chapter 4 outlines two tools for the implementation of these improvement efforts.

This page is intentionally blank.

4. Solving the Problem: Tools for Improvement

Having defined three improvement efforts aimed at increasing control of development lead time in Chapter 3, the next step was to develop tools for implementing them. This chapter begins with an outline of two more types of process improvement. They are used as a basis for the design of a delay tracking and analysis tool. The use of a project simulation to demonstrate the expected effects of the improvement efforts is then described.

4.1 Process Improvement: The TQM Framework

TQM theory (Shiba, S., A. Graham, D. Walden, p. 49) describes three types of process improvement, including:

- Process Control
- Reactive Improvement and
- Proactive Improvement.

Each of these types of process improvement has different objectives. As discussed in Chapter 2, the objective of proactive improvement is to identify general direction for improvement. The use of a proactive process improvement tool called “Voice of the Customer” to identify inhibitors to control of development lead time was described. Where proactive improvement is focussed on setting direction for improvement, process control and reactive improvement are more implementation-oriented. The following sections discuss the application of process control and reactive improvement to the product development process.

4.1.1 Process Control

The objective of process control is to determine the variation of the output of a process and identify when corrective action is required, i.e. when the process is out of control. It involves measuring the output of the process and comparing it to the desired output – on a continuous basis. A familiar example of the application of process control would be the tracking of a product dimension in a manufacturing process. However, this idea can apply to any process. In the context of product development, in section 3.2 the case was made for tracking activity delay.

The idea behind process control is that the actual process output will vary within a range. By tracking the process output over time a comparison can be made with the desired output range. If the range of actual output is outside the desired range, the need for action is identified. The objective of the action can be to i) improve the process by reducing the variation, or ii) reset the nominal or average value of the output.

With regard to reducing variation, TQM recognizes two types of variation: controlled and uncontrolled. Controlled variation is variation inherent in the process and is due to what Deming refers to as “common” causes. It results in a stable pattern of outputs over time. Uncontrolled variation is variation due to abnormal changes to the process, or what Deming refers to as “special” causes. The first step to improving a process is to understand and remove the uncontrolled variation so that the controlled variation – which is typically smaller – can be made visible and then addressed.

In the context of this thesis, the objective of process control is to improve control of development lead time by identifying controllable and uncontrollable delay. Recall from section 3.1 that controllable causes of delay referred to causes of delay over which Program Management had some degree of control. The “efficient project” was defined as one in which rework is due solely to uncontrollable causes of delay. Thus the definitions of “controllable” and “uncontrollable” delay causes in this thesis are the reverse of the standard TQM terminology described above. However, it was felt that these definitions would be more intuitive for the purposes of this thesis.

After collecting activity delay data and separating it into controllable and uncontrollable causes, reactive improvement can be used to reduce the controllable causes of delay.

4.1.2 Reactive Improvement

The objective of reactive improvement is to improve a process by determining root causes of the problem(s) and then implementing appropriate solutions to prevent their recurrence. Reactive improvement is the approach by which undesired variation in process output – identified using process control – can be reduced. The tool used in for reactive improvement of weak processes is known in TQM as the 7 steps (Shiba, S., A. Graham, D. Walden, p. 53). Shiba et al. describe these steps as:

1. Select a theme (a specific improvement, such as “decrease after-shipment bugs reported in product X”).
2. Collect and analyze data (to discover what types of bugs occur most often).
3. Analyze causes (to discover the root cause of the most frequent type of bug).

4. Plan and implement solution (to prevent the root cause from recurring).
5. Evaluate effects (to check the new data to make sure the solution worked).
6. Standardize solution (to permanently replace the old process with the improved process).
7. Reflect on the process and the next problem (to consider how the problem-solving process could have been better executed and to decide which problem to work on next, such as the next most frequent type of bug from step 2).

Note that these steps are common to the basic problem-solving processes now part of many corporations. In the context of product development and this thesis, reactive improvement means analyzing the activity delay data and determining the significant controllable causes of delay. After identifying these controllable causes of delay, action can be taken to prevent their occurrence in future projects.

Given the context of these types of process improvement it is interesting to recognize that the approach to improvement described above closely parallels that of reducing variability in a manufacturing process. Given the high degree of familiarity with process improvement in manufacturing and its acceptance as a competitive necessity, it may be helpful to compare how the process applies in manufacturing and product development contexts. Figure 6 illustrates the reactive improvement process, showing examples of its application in manufacturing and program management (in product development).

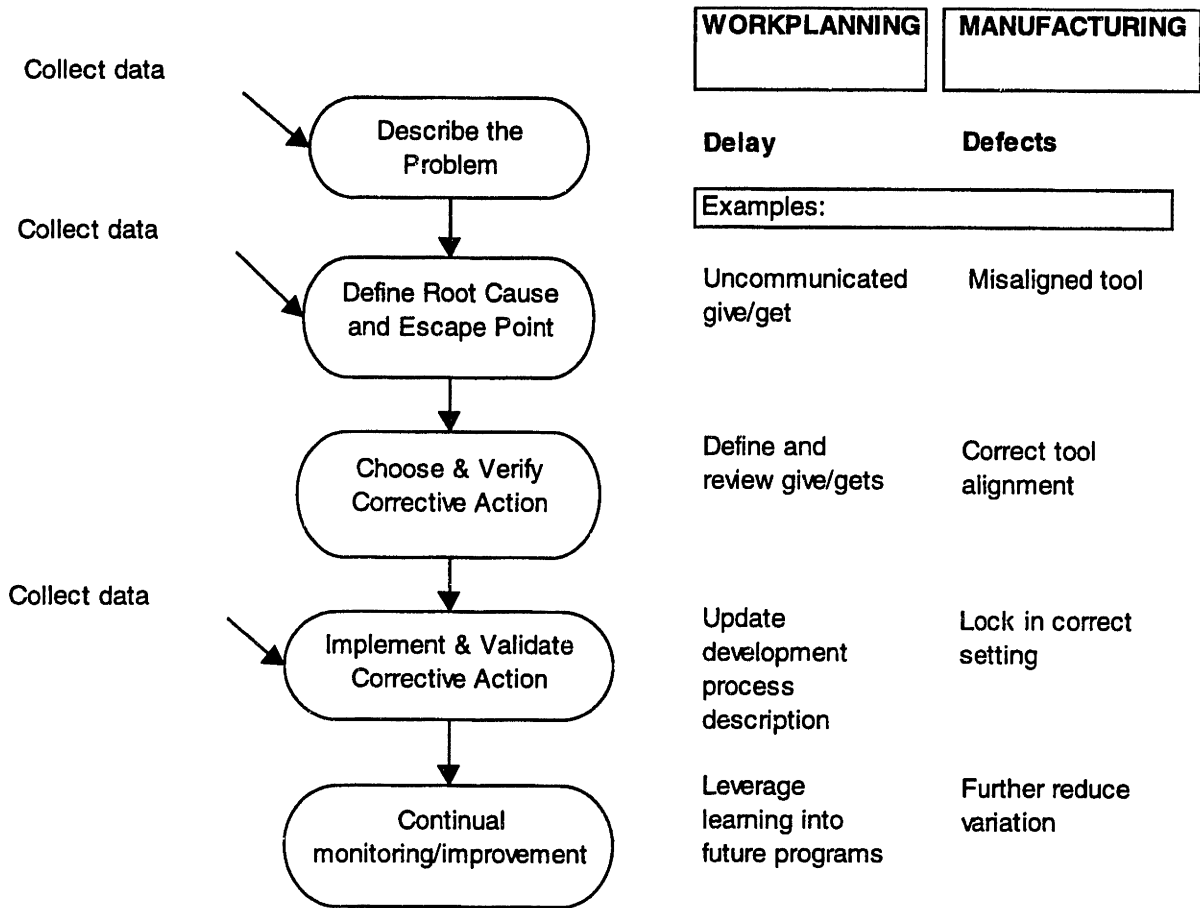


Figure 6: Reactive process improvement methodology in product development and manufacturing contexts

The key observation is that after defining activity delay in product development as the equivalent problem of product defects in manufacturing, the remainder of the process follows naturally. An important difference that should be noted however is that product development is a one-time process as opposed to the more repetitive nature of manufacturing processes. However, as Reinertsen argues, there is some level at which product development process can be considered standard (Reinertsen, 1997, pp. 119-120). He points out that while attempting to standardize a development process at high levels of detail will result in endless frustration, at a low enough level of detail, common modules can be put together in a variety of ways to “build” any project.

He defines modules as building blocks of the development process with standardized interfaces (inputs and outputs to upstream and downstream modules). The internal sub-processes in the module can vary

with the project. In the context of the delay analysis tool and simulation described in this thesis, the modules are the development activities for which delay is tracked. The module interfaces are defined by the “give-gets” or deliverables between the upstream and downstream activities.

In order to leverage the knowledge gained from the delay analysis tool and simulation across all current and future development projects, the selected modules (development activities) should be sufficient to “construct” projects of varying scale. This remains to be seen, but the activities used in this thesis fit the most typical scale project at the company. Further discussion may find that this activity set could be expanded to slightly greater detail to make it relevant to smaller scale projects.

4.2 A Data Collection Tool: the Key to Process Improvement

This section describes a tool for tracking and analyzing activity delays in a project. In the context of the preceding discussion of the different types of process improvement, this tool fulfills the process control function of collecting and separating controllable and uncontrollable delays. It also identifies significant controllable delay causes, a key step in reactive process improvement.

There are two overall objectives for the delay analysis tool:

- Identify controllable causes of delay that can then be addressed, and
- Gather real data on the relationships (dependencies) between activities.

Tracking and analyzing delay due to controllable causes provides data that can be used to: i) identify problematic delay causes, ii) monitor the effect of improvement efforts, and iii) assess the efficacy of recovery strategies.

By tracking linkages between delays associated with dependencies between activities, the delay analysis tool can be used to: i) give visibility to the effect of changes in design requirements, and ii) facilitate the determination of parameters describing the dependencies between activities (probability and impact of rework), with which a more accurate project model can be built.

The details of how the output of the tool can be used to achieve these results are described below in section 4.2.3.

Figure 7 illustrates the delay analysis tool as a schematic showing inputs and outputs. To achieve the two overall objectives described above, the tool has the following functions:

- Distinguish between root delays and delays linked to these roots
- Distinguish between delays associated with rework/changes in design and those that are not
- Separate delays associated with recovery actions
- Track level of detail for missed activities
- Track degree of definition for failed give-gets
- Identify opportunities in which delays can be detected earlier

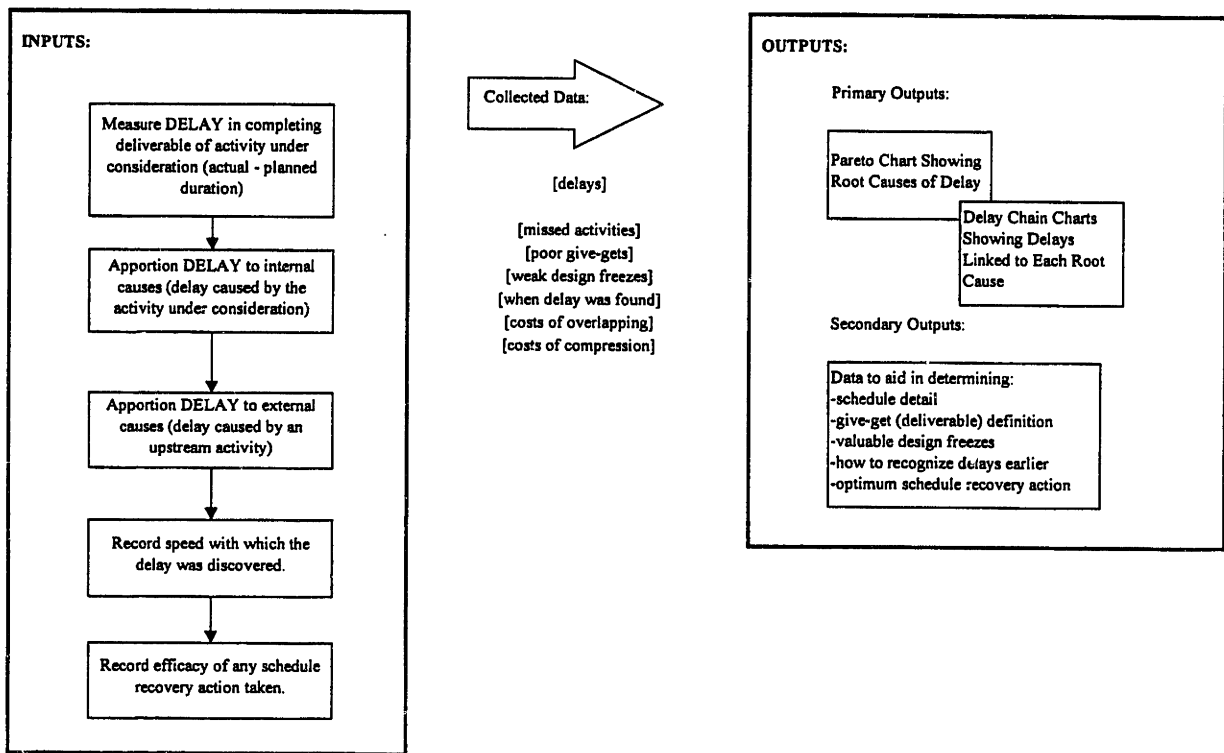


Figure 7: Schematic of delay tracking and analysis tool

The following describes the step-by-step application of the tool in two phases: input and output. It is envisioned that this tool would be implemented in a web-based format to facilitate easy data entry and analysis.

4.2.1 Inputs

Each activity has one or more give-gets or deliverables that when met, define the completion of the activity. The delay analysis tool was designed to be applied at the completion of each of these deliverables. When a deliverable is completed, the first step is to quantify the size of the delay, i.e. actual

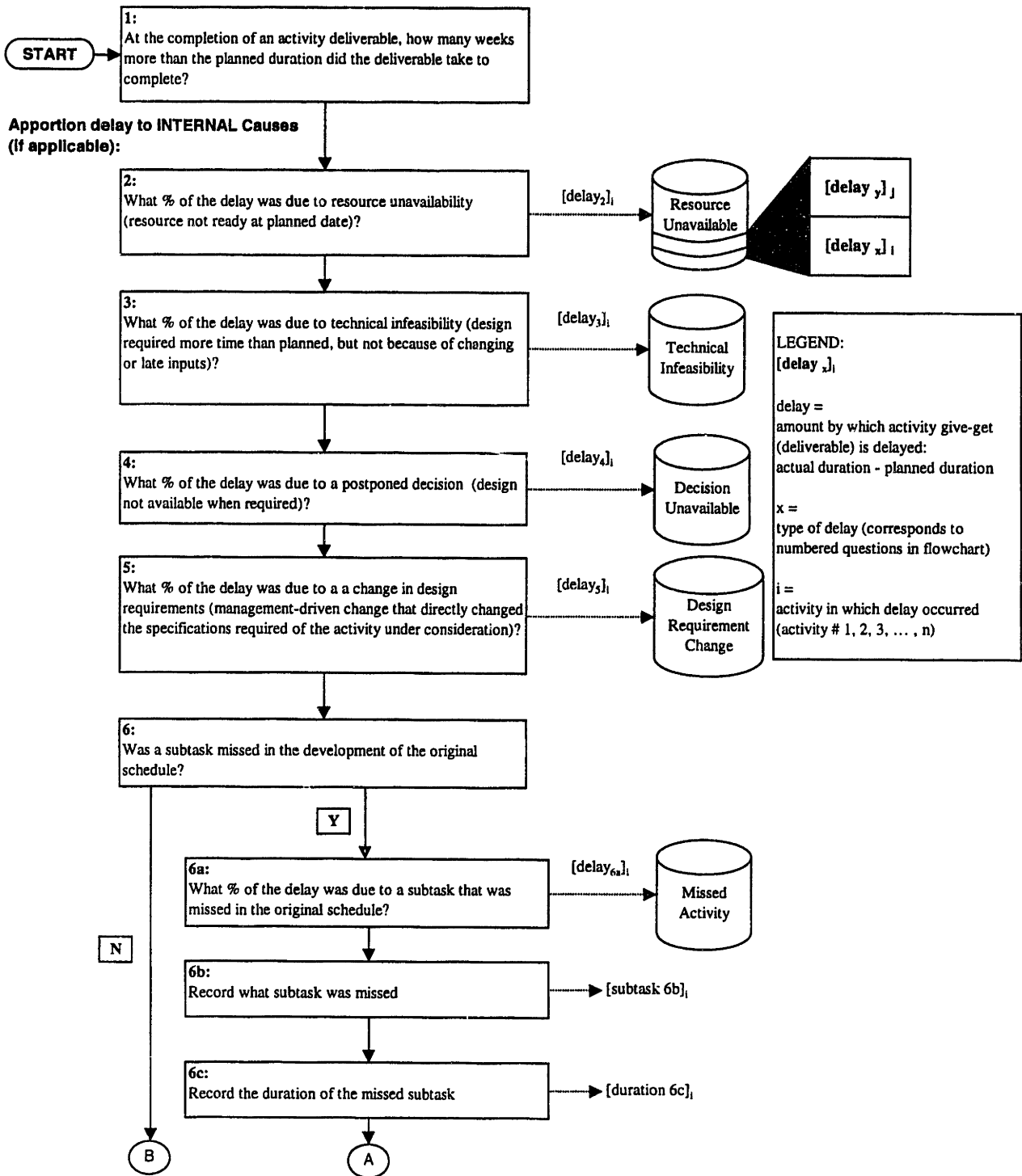
activity duration minus planned duration. The next step is to apportion the total delay to various causes. Note that the inputs are divided into 4 groups: internal cause, external cause, recovery, and discovery. These are related to the functions outlined above. Internal causes are those causes of delay that are within the control of the activity under consideration. External causes refer to causes of delay attributed to an activity upstream of the activity under consideration.

In order to allow the effects of recovery actions to be tracked, the tool provides for the recording of the amount and cost of compression and/or overlapping applied. As in the simulation model, compression refers to the use of overtime to reduce the duration of the *activity under consideration* and overlapping refers to the start of a *downstream activity* prior to the completion of the activity under consideration.

In order to promote the early detection of delays, the tool solicits ideas for identifying similar problems in the future earlier.

The following flowchart in Figure 8 describes the input process and identifies the data collected.

Figure 8: Flowchart for delay analysis and tracking (Inputs)



Apportion delay to EXTERNAL Causes (if applicable):

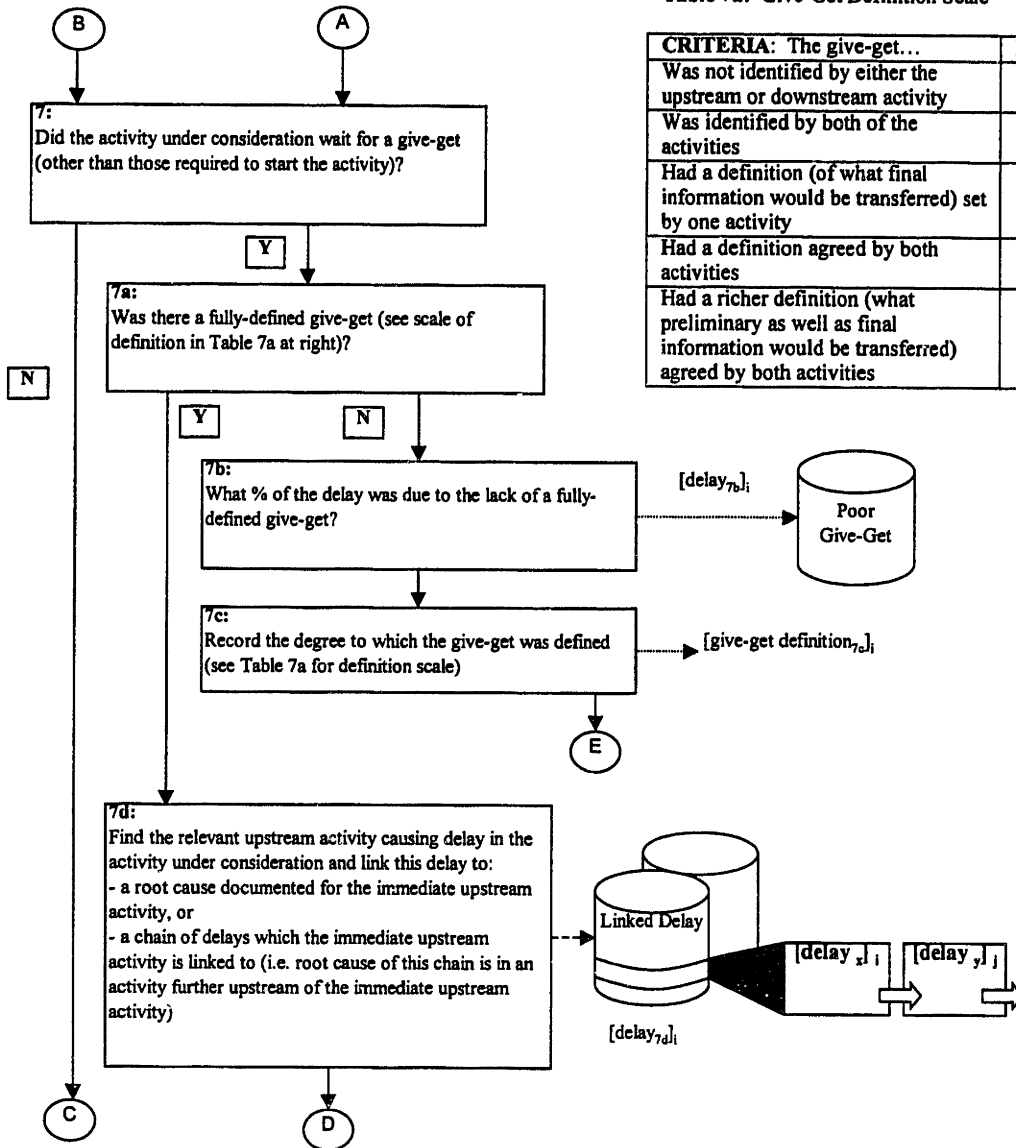
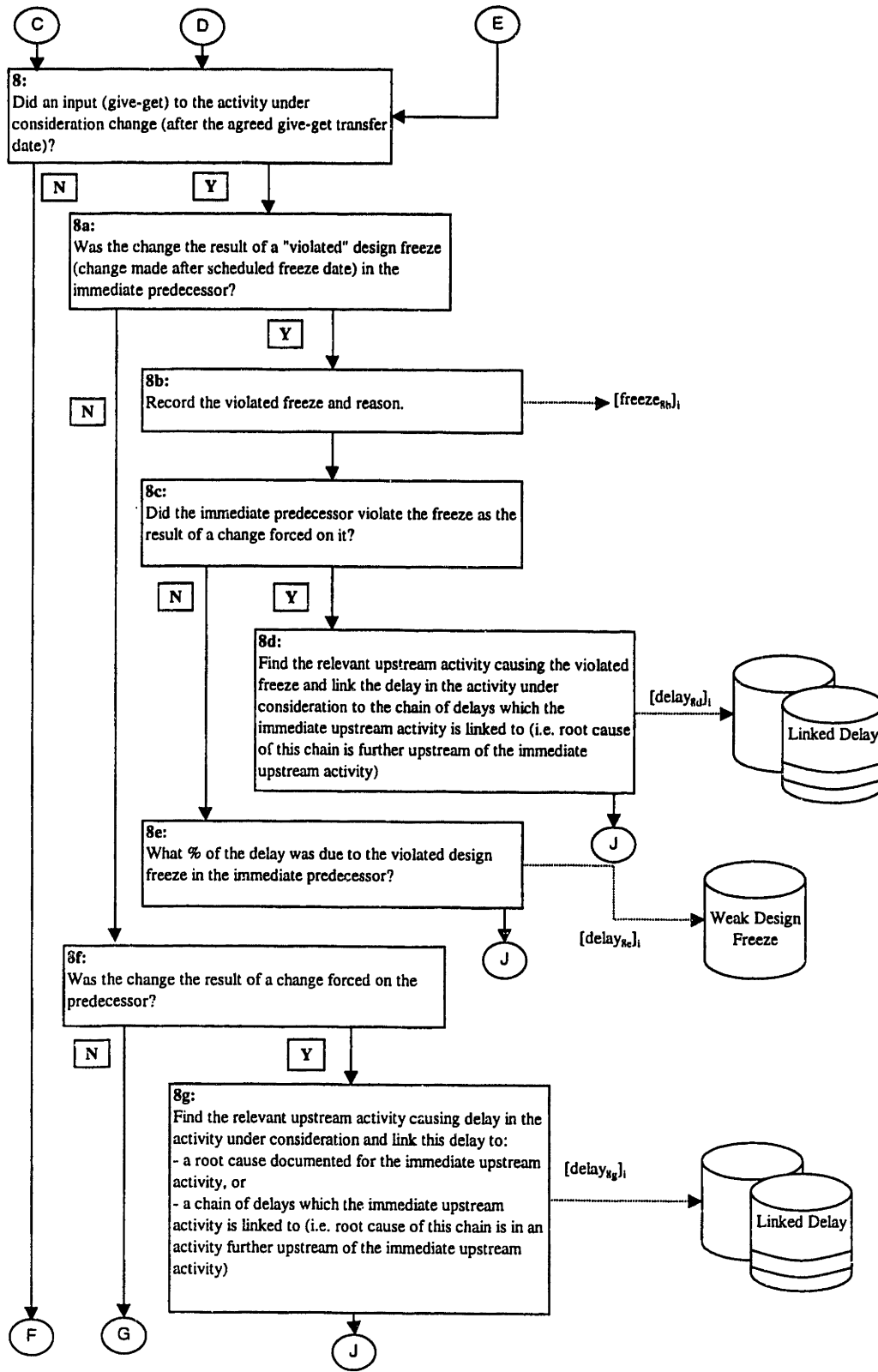
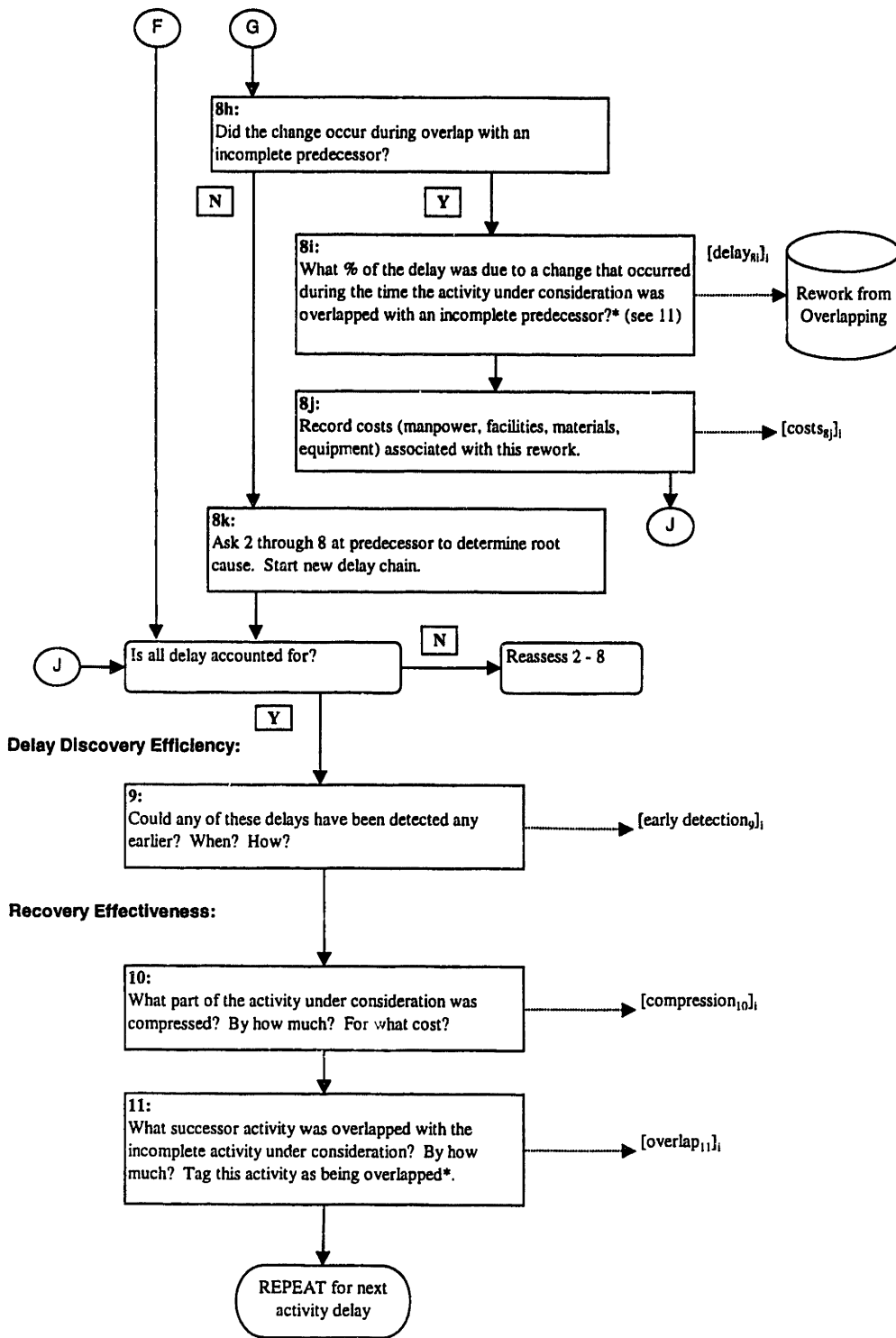


Table 7a: Give-Get Definition Scale

CRITERIA: The give-get...	DEGREE
Was not identified by either the upstream or downstream activity	0
Was identified by both of the activities	1
Had a definition (of what final information would be transferred) set by one activity	2
Had a definition agreed by both activities	3
Had a richer definition (what preliminary as well as final information would be transferred) agreed by both activities	4





4.2.2 Outputs

There are two primary outputs from the delay analysis tool related to the functions outlined above:

- Pareto chart relating delays to root causes
- Linked delay charts showing the delays related to root delay

The tool also has 4 secondary outputs related to suggestions for improving the process:

- Activity-specific charts showing the number of delays due to missed subtasks as a function of the level of detail in the schedule
- Give-get specific charts showing the number of delays due to incompletely defined give gets as a function of the degree of definition
- Opportunities for earlier detection of delays
- Problematic weak design freezes

The generation of these outputs from the inputs and the relationship of these outputs to the stated objectives of the tool are discussed below. Terms enclosed in [] refer to inputs from the various steps of the flowchart.

- Root Delays -

A root delay in an activity is a delay that originates in the activity, i.e. it is not due to a delay in an upstream activity. These root delays can be the result of any of the root causes identified in the flowchart of Figure 8 (and also shown in Figure 9).

By grouping the delays for all activities in the project $[\text{delay } x]_i$ by type of root cause, a Pareto chart in the form of Figure 9 can be constructed. Note that delays from all inputs except the linked delays (7d, 8d, 8g) are included in this chart.

Root Causes of Delay: Pareto Chart

Example of chart that could be developed from data gathered over 2 projects:
(hypothetical data)

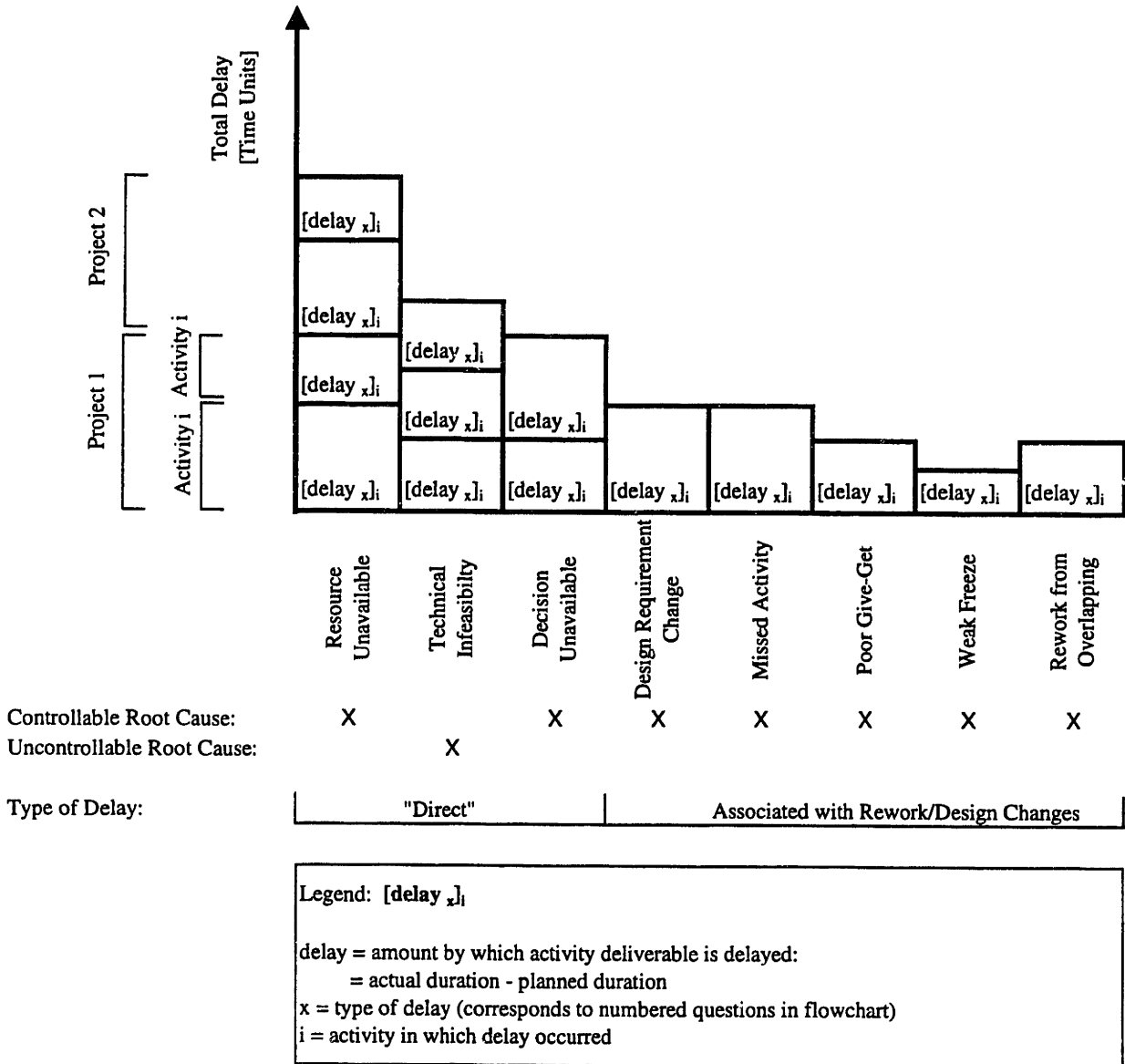


Figure 9: Root cause Pareto chart

In relation to the overall objectives of the delay analysis framework described above, this chart can be used to:

- Focus improvement efforts by identifying consistently problematic causes of delay
- Monitor the effect of improvement efforts
- Assess the efficacy of recovery actions

The Pareto chart concisely illustrates the problematic causes of delay, i.e. those that recur frequently and/or have long delays associated with them. The cumulative delay over all projects for a root cause of type 'x' is:

$$\text{Cumulative DELAY}_x = \sum_{\text{all projects}} \sum_i [\text{delay}_x]_i$$

where i = any activity in a project that was delayed by a root cause of type 'x'.

The larger DELAY_x the more problematic the type of delay. Identification of consistently problematic delay causes provides focus for process improvement efforts and provides future programs with an idea of which activities have a high-risk of extended duration.

Collecting and comparing this data across projects allows the effect of any improvement efforts to be quantified. Note that to apply this tool to different scale projects, an agreed method of normalizing the activity delays needs to be established so that they are comparable between large and small-scale projects.

This may be as simple as declaring a reference scale project with standard activity durations. Projects that are larger or smaller in scale would have correspondingly higher and lower target activity durations. The observed activity delays for a larger-scale project would then be decreased by the ratio of the target activity duration to the standard activity duration. Similarly the observed activity delays for a smaller-scale project would be increased by the ratio of the target activity duration to the standard activity duration. Alternatively, project delays could all be converted to and tracked as percentages of the target activity duration.

By linking observed activity delays to recovery actions taken, the delay analysis tool allows the efficacy of compression and overlapping to be tracked. This efficacy will be measured in terms of ability to reduce delay and the cost associated with using the recovery action.

As a minor note, the root causes can be broadly grouped into those associated with rework or a design change, and those that are not. The latter group will be referred to as "direct" delay. This distinction is a useful way of characterizing a project and is also related to the parameters used in the simulation model. In the model, the delay associated with rework is captured by the probabilities and impacts of rework in the DSM, whereas the "direct" delay is captured in the variation in initial duration of each activity (before the addition of any rework).

The secondary outputs are related to the root causes and are useful for focussing improvement of project management functions: coordinating communication, scheduling activities, and coordinating design freezes.

For delay due to subtasks of an activity that were missed during the development of the original project schedule, a chart in the form of Figure 10 can be generated. This chart is specific to a particular activity in the project, but would be added to as data from the same activity in future projects is gathered. The chart illustrates the relationship (if any) between the level of detail for the activity in a schedule and the number of delays and can thus be used to help determine the appropriate level of detail with which to schedule a given activity.

Delay Occurrences and Level of Detail

Example chart that could be developed from data gathered over multiple projects:
(hypothetical data)

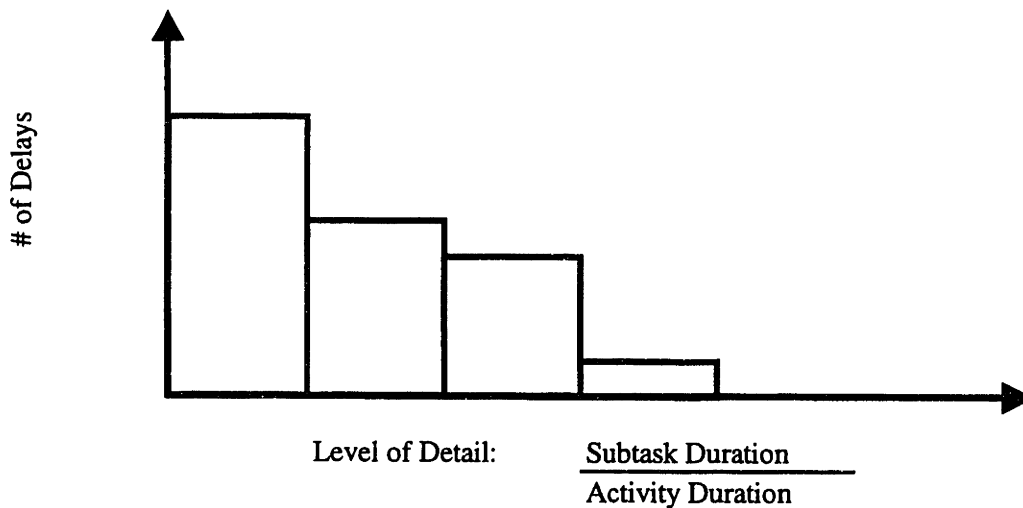


Figure 10: Chart to analyze relationship between number of delays and level of schedule detail

When a missed subtask of an activity 'i' causes delay in a project, it is captured by input 6. The particular subtask [subtask 6b]_i and amount of the delay [duration 6a]_i are recorded. To establish a measure of the level of detail of a subtask, the duration of the subtask as a fraction of the duration of the activity of which it is part, can be determined, i.e.

$$\text{level of detail (of subtask } j \text{ of activity } i) = \frac{\text{duration of subtask } j}{\text{duration of activity } i}$$

If each time a subtask of activity_i in any project is missed it is added to this chart, the data can be analyzed for patterns. Finding many of instances of missed subtasks associated with high level of detail would suggest that using a greater level of detail in the schedule would be beneficial, i.e. activity_i could be divided into smaller activities.

For delay due to poorly defined give-gets, a chart in the form of Figure 11 can be generated. This chart is associated with a specific give-get between a pair of activities, but would be added to as data about the same give-get in future projects is gathered. The chart shows the relationship (if any) between the degree of give-get definition [degree 7c]_i and the number of delays. If a particular give-get has a high number of delays associated with it, its definition (see Figure 11) should be reviewed to reduce ambiguity.

Delay Occurrences and Give-Get Definition

Example chart that could be developed from data gathered over multiple projects:
(hypothetical data)

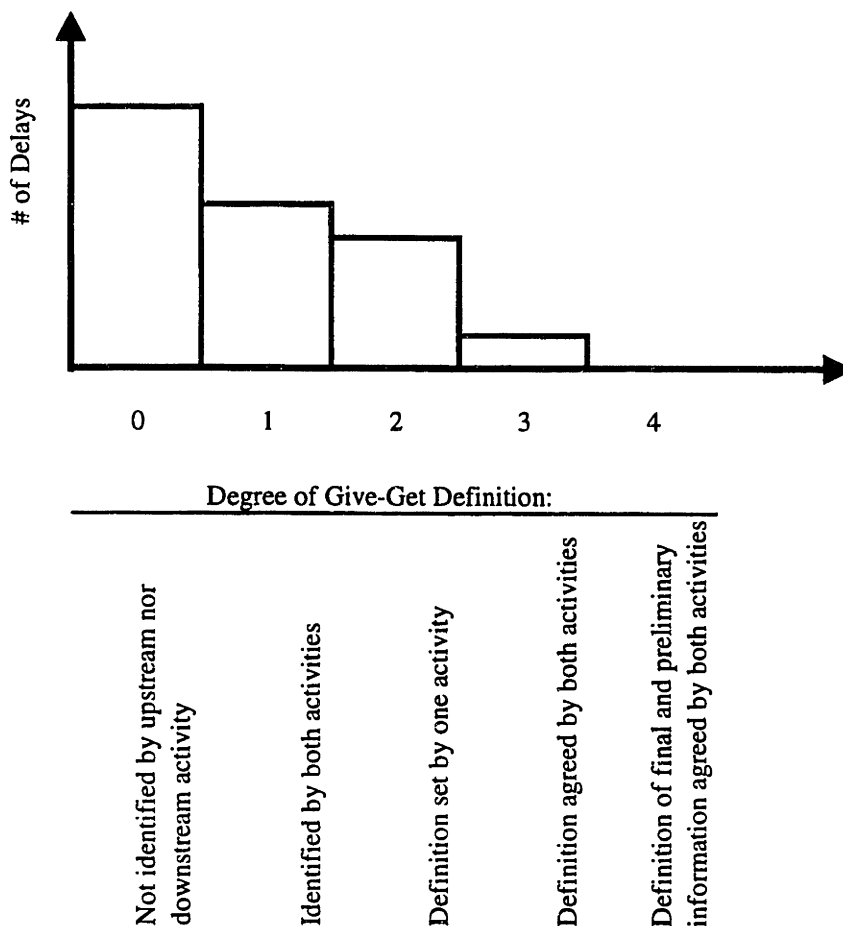


Figure 11: Chart to analyze relationship between number of delays and give-get definition

To promote the early detection and recognition of delays, the tool solicits and records suggestions as to how much earlier and by what mechanism a similar delay in future programs could be detected [early detection 9]_i.

To identify design freezes that are difficult to adhere to, the tool records the activities affected by these “drifting” freezes [delay 8d]_i and what aspect of the frozen design was changed [freeze 8b]_i. Patterns in this data can be used to help establish what freezes are valuable. As discussed in section 2.3.1, a design freeze that is consistently difficult to adhere to may be more of a disadvantage than a benefit.

- Linked Delay Charts -

The analysis tool was designed to link any activity delays associated with a previously recorded root delay in an upstream activity. By charting all the delay associated with a particular root delay, a chart in the form of Figure 12 can be constructed. This chart thus illustrates the consequences of a particular root delay, i.e. a chain of all the delays related to the root. This type of chart would be generated for each root delay (should the effects go beyond the activity in which the root delay occurred).

Linked Delays

Example delay "chain" for one project:
(hypothetical data)

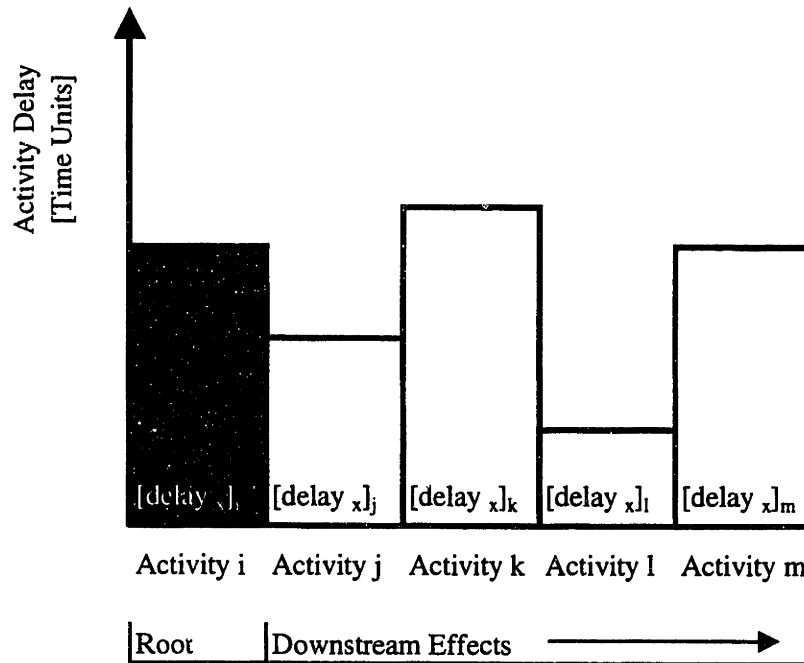


Figure 12: Chart showing delays linked to a particular root delay

The total delay associated with one particular instance of a root delay 'Y' in a project is:

$$\text{Total Linked DELAY}_Y = \text{Root} [\text{delay}_Y]_i + \sum_i [\text{delay}_x]_i$$

where i = all activities where delays linked to the root delay 'Y' occurred.

In relation to the objectives described above, this chart can be used to illustrate and quantify the effect of a change in design requirements (one type of root cause) by showing all the delays linked to it. Secondly, by analyzing the frequency of occurrence and the size of the linked delays, historically-based parameters for the simulation model can be determined. This more accurate model can then be used for three real-time project control analyses. These applications of the model are described in the next section.

4.3 A Simulation to Demonstrate the Improvement Approaches

4.3.1 Description

This section describes a project simulation that was used to investigate the following:

- The effect of controllable causes of rework on project completion date
- The efficacy of various schedule recovery strategies
- The effect of rework on project progress.

Note that these simulation applications can be interpreted in two contexts. In the context of this thesis, they illustrate the expected results of the three improvement efforts outlined in section 3.2. In the context of application in an actual project, they demonstrate how the simulation can be used to: i) evaluate the effect of a proposed change in design requirements on project duration, ii) determine optimal schedule recovery strategies when a delay occurs, and iii) determine a more representative basis against which to measure project progress, i.e. one in which expected activity durations include potential rework.

The objective of this section is to illustrate how the simulation can be used and to draw general conclusions – not necessarily to determine detailed operating policies. The point is made here since the simulation parameters (outlined below) were based on general estimates – as relevant historical data proved to be difficult to obtain. The sensitivity of the simulation to these parameter choices is addressed in the discussion of results.

However, as described above, the vision is that by implementing the delay analysis tool described in section 4.2, relevant data can be collected to determine more accurate parameter values for the simulation. With parameters based on real historical data, the output of the simulation could conceivably be used for more detailed operational planning. In this way, the simulation would generalize the learnings from the delay analysis tool. Furthermore, given a validated model, such a simulation could be used in real time, i.e. to monitor project progress and analyze schedule recovery actions during a project – as opposed to using the model for general planning a-priori. This idea will be discussed further in section 5.3 in the context of further work.

The simulation used for this thesis was an extended version of a DSM-based, Monte-Carlo development project simulator developed by Browning (Browning, 1998b, chapter 6). In this model, a development project is described using an activity-based Design Structure Matrix (DSM). This method of representing

and analyzing systems was introduced by Steward (Steward, 1981) and extended by others (Eppinger et al., 1994).

The DSM captures 3 types of dependencies between activities: independent, dependent, and interdependent. Independent activities do not depend on each other in any way and can thus be executed in parallel without consequence. A dependent activity requires information from an upstream activity and must therefore follow the upstream activity. The information flow is forward through sequential activities. This “feedforward” relationship is shown as a sub-diagonal entry in the DSM. Interdependent activities are coupled and since they require information from each other, usually result in iteration. The information flow is both forward from activity A to B and backward from B to A. The “feedback” relationship is shown as a super-diagonal entry in the DSM.

A set of 26 activities was selected to represent a generic vehicle development program. These activities and their execution sequence were based on the company’s standard work breakdown structure and the associated give/gets or deliverables traded between these activities. This work breakdown structure was developed for a program of typical complexity, i.e. carry-over platform with new body. Note that several of the activities have been divided into sub-activities. This was to allow the timing of the give/gets between upstream and downstream activities to be matched. For example, the activity “Set Specifications” spans 11 time intervals in the standard plan. However, this activity feeds information to 3 other activities at different times during this time period. Thus, the activity is divided into three sub-activities: “Set Specs-1”, “Set Specs-2”, and “Set Specs-3”.

Note that one of the benefits of developing a DSM representation of a project is the ability to determine activity sequences that minimize the potential for rework (Eppinger et al., 1994). The intent of this thesis was to model the existing process so the sequence in the company’s standard development process was used. Given the many dependencies between the rather high-level activities used in this model of the project, it is likely that further dividing the activities into more-detailed activities with fewer dependencies would improve the chance of finding more optimal sequencing.

There are two dimensions to the relationship between any two activities: the probability of rework and the impact or amount of rework should rework be required. These two dimensions are documented in two DSM matrices: a rework probability matrix and a rework impact matrix. Each cell in the probability matrix is filled by answering the question “What is the probability (between zero and 1) that activity ‘i’ causes rework for activity ‘j’?”, i.e.

$$DSM_{ji} = P(\text{activity } i \text{ causes rework for activity } j).$$

Each cell in the impact matrix is filled by answering the question “if activity ‘i’ does cause a change in activity ‘j’, how much of activity ‘j’ must be reworked?”, i.e.

$$DSM_{ji} = \% \text{ of activity 'j' that must be reworked, given that activity 'i' changes.}$$

Following are the DSM matrices (probability of rework and impact) showing the activities and the various relationships used to model a vehicle development program in this thesis. It is worth noting that the feedback loops in this DSM are relatively small, i.e. with the exception of feedback from “CP Testing” to “Design-2”, none of the feedbacks go back more than 3 activities.

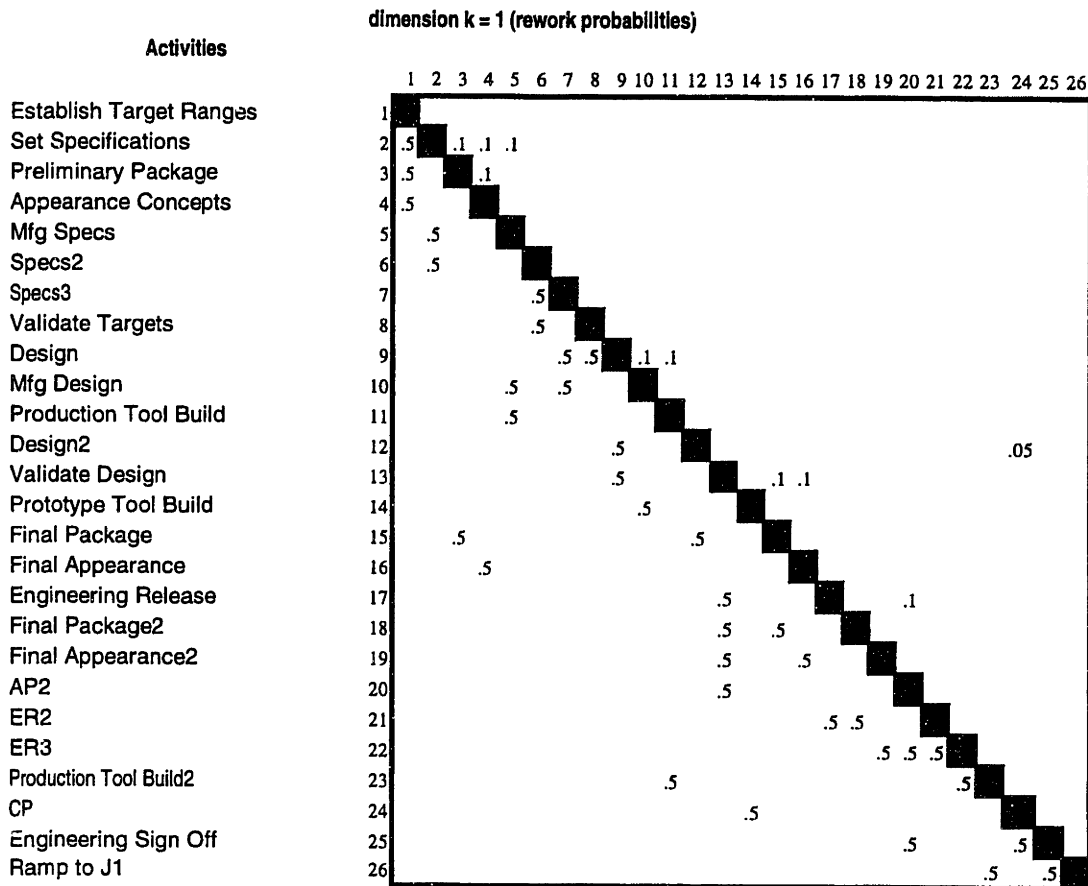


Figure 13: DSM matrix: rework probabilities

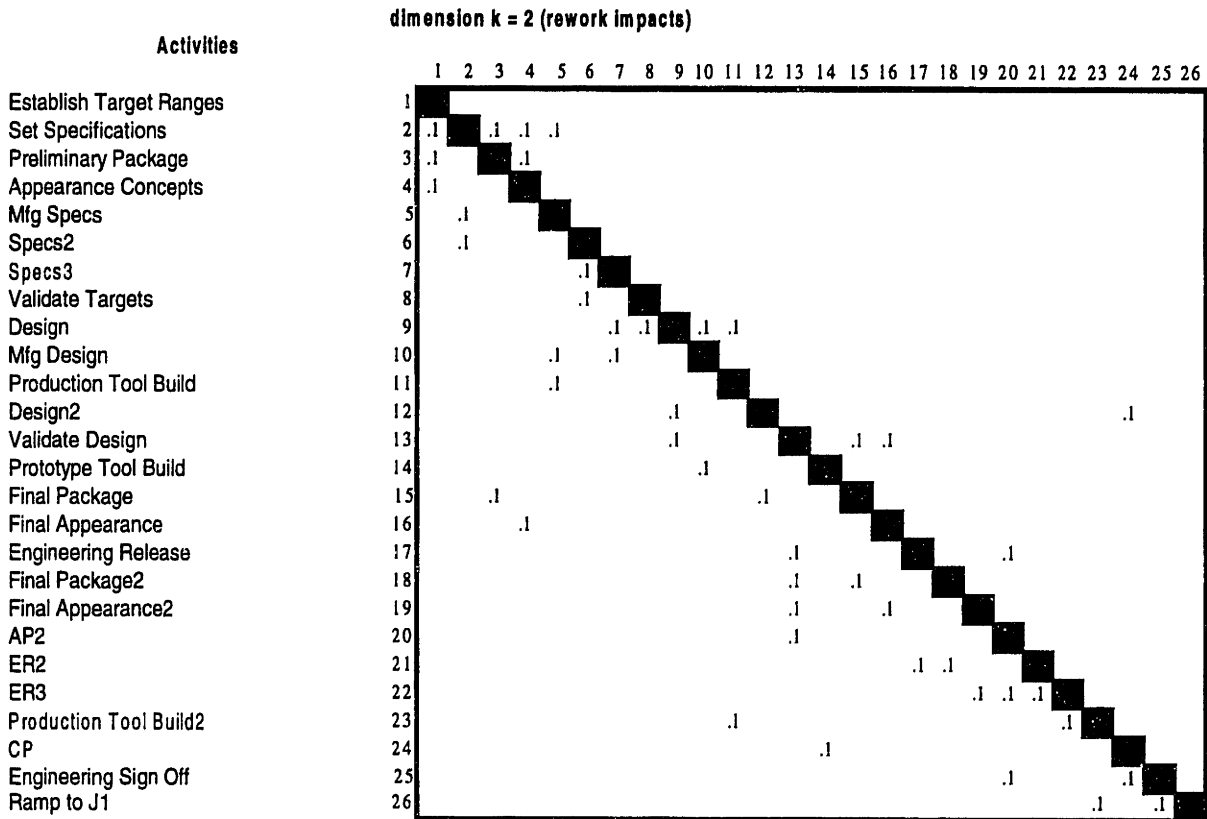


Figure 14: DSM matrix: rework impacts

In line with the point made earlier that the objective of this thesis is to illustrate *how* such a simulation could be used in planning and managing a concurrent development project, this rather general set of activities could be expanded as the need for greater detail becomes apparent. However, the goal is to make the learnings gained by using the delay analysis tool and the simulation in a specific program relevant to all future programs. Thus the activities used in them should be selected based on the ability to use these activities as “building blocks” to construct a model for any program.

Following is a brief explanation of the structure and operation of the simulation. For a more detailed description see Browning (Browning 1998b, chapter 6). The required inputs for the simulation include:

- Sequenced activities and their dependencies (DSM)
- Activity durations (min, most likely, and maximum)

Given these inputs the simulation randomly determines a duration for each activity (within the specified minimum and maximum) and then steps through the project in user-specified time intervals. At each time step the following occurs:

- Do work on each activity that does not have incomplete predecessor activities
- If an activity, say 'A' is completed during this time step, if feedback is possible (superdiagonal entries in DSM), randomly determine whether rework occurs for the related upstream activity, say 'B'.
 - If rework does occur, add the amount specified in the DSM to the affected upstream activity 'B'. This is referred to as "first-order" rework.
 - If rework does occur in 'B', if feedforward is possible (sub-diagonal entries in DSM), add the amount specified to the related activities downstream of this activity, e.g. rework in 'B' causes rework in 'C' and 'D'. This is referred to as "second-order" rework.
- Advance to next time step and repeat until project is complete (no activities with work remaining).

The output of the simulation is total project duration for each run. Each scenario in this thesis was run 300 times allowing histograms and probability vs. project duration plots to be drawn.

4.3.2 Modeling the Approaches

This section describes how the simulation was used to model the three improvement efforts described in section 3.2.

- Modeling: Effect of rework on project completion -

In the simulation, project completion date can be affected by two general factors: variation in the initial activity durations, and potential rework. Initial activity duration is the duration before the addition of any rework and is randomly selected from a user-specified distribution. The effect of rework was assessed by determining the sensitivity of project completion date to various probability and impact DSMs. This analysis was useful for two reasons. Not only did it give an indication of the sensitivity of the results to uncertainty in the benchmark DSM, it could also be interpreted as effects of some of the previously described controllable delay causes. These interpretations are discussed below.

The controllable delay causes can be related to the probability and impact DSMs. The effect of a change in design requirements on project completion date can be modeled as a feedback loop with probability of occurrence 100%. For example, consider a project where design requirements are established in activity

'j'. If at a later activity 'i' a decision is made to change these design requirements, activity 'j' is forced to do rework. The potential for rework of activities dependent on activity 'j' is also created. A change in requirements that occurs at the end of activity 'i' and directly affects activity 'j' would thus be entered into the probability matrix as: $DSM_{ji} = 1$. Note that the timing of such changes in design requirements would likely coincide with design reviews scheduled into the project.

In section 2.3.1 two types of poor communication were described as causes of controllable delay: late discovery of design problems, and poorly defined information exchanges. As described in earlier, when a design problem is discovered late, the consequence is that the amount of rework is typically increased. Thus the effect of late discovery of problems on project completion date can be modeled as increased impact, i.e. larger entries in the impact matrix.

When an information exchange is poorly defined (what is to be exchanged and when), the probability of delay is increased. As explained in 2.3.1, this can be the result of one of two general failure modes: differences between what each party believes to be accurate information, or the upstream activity not being aware that the downstream activity is expecting information from it. In either case, the probability of rework is increased and the effect of poorly defined information exchanges can be modeled using larger entries in the probability matrix.

As described in 2.3.1, a weak design freeze, i.e. one that allows the design to "drift" beyond the declared finalization date, increases the likelihood of rework for the activity using information it believes to be finalized, but in fact is not. If we make the reasonable assumption that the rework is not discovered until some time after the declared freeze, the amount of rework required increases. Thus the effect of a weak design freeze can be modeled as increased probability of rework and impact, i.e. larger entries in the probability and impact matrices.

- Modeling: Efficacy of recovery actions -

Given the schedule recovery options of compression and activity overlapping, the simulation was used to determine the ability of these actions to maintain scheduled project completion date and to estimate the costs of using them. Also of interest with regard to schedule recovery was the identification of activities for which delay should be expected, given the variation in initial activity durations (durations before the addition of any rework) and the possibility of rework.

In order to model schedule recovery actions, it was necessary to extend the simulation developed by Browning (Browning, 1998b, chapter 6). For the purposes of this model, it was assumed that schedule recovery actions are not implemented until the start of an activity is delayed, i.e. there is no proactive identification of delay. While this is certainly not ideal, it is not unrealistic given that the complexity of a vehicle development project can easily obscure the downstream effects of a delay. This complexity is due to the large number of dependencies between tasks, a significant proportion of which are interdependencies which make feedback a risk. The relaxation of this assumption is discussed in section 5.3 in the context of further work.

It should also be noted that the model only checks for delay in activity start on the activity's scheduled start date. Thus schedule recovery action can only be motivated by a given activity once. While in reality the delayed start of an activity can persist beyond the first implementation of recovery action, this decision was made to make the model conservative in its ability to eliminate delay.

A subtle but important point that should be made is that these recovery actions are driven by delays in the *start* of an activity, as opposed to the delay in the finish of an activity. This allows delays in the completion of an activity to persist so long as they do not affect the start of a downstream activity, i.e. activities not on the critical path can tolerate some degree of delay.

When an incomplete upstream activity prevents the scheduled start of an activity, the model checks to see if compression is to be used, and if so, what fraction of the delay can be eliminated using it. The model then reduces the work remaining *of the upstream activity* by the lesser of: the delay in the completion of the upstream activity and the maximum allowable amount of compression. The maximum amount of allowable compression is specified as the fraction by which the delay in completion of the upstream activity can be reduced using compression.

This raises another simplification of the model. The model makes the assumption that if compression is permissible, the resources to perform it are available. And, while different physical limitations mean that the maximum allowable compression will vary by activity, the model assumes that all activities can be treated similarly. This will be discussed in section 5.3 in the context of further work.

The incremental cost of the compression is the premium paid for overtime work. For example, if overtime is paid at "time and a half", the premium is 50% of the time worked at the overtime rate. The amount of time worked at the overtime rate is equal to the size of the delay, less the fraction of that delay

eliminated by compression. Thus to compress an upstream activity delaying the scheduled start of a related downstream activity, the incremental manpower cost, in units of time is:

$$\text{Cost}_{\text{compression}} = \text{Overtime premium} \times (\text{Delay} - \text{Fraction of delay reduced by compression})$$

If, after implementing compression the incomplete upstream activity is still delaying the scheduled start of the downstream activity (or if compression is not permitted), the model checks to see if overlapping is allowed. If it is, the model reduces the amount of work required *for the downstream activity*, by an amount equal to the lesser of: the delay in the upstream activity and the maximum allowable overlap. In the simulation, this has the same effect as allowing the downstream activity to proceed prior to the finish of the upstream activity. But using this approach allowed more of the original code to be retained.

The maximum allowable overlap is specified as a fraction of the downstream activity duration. Again, in the interest of simplification, the model uses the same maximum overlap for all activities. If overlapping is permitted, the model assumes that the conditions that facilitate it are present: activities can be overlapped (i.e. physical constraints do not prevent overlapping), and that the required communication between the overlapped activities can be achieved (i.e. through co-location, frequent updates, and a high degree of relevant experience among the group members).

The incremental cost of overlapping is the use of resources (staff, facilities, and materials), for rework, only if it is required. Overlapping involves starting an activity with unfinalized information. So while it is possible that overlapping an activity will not incur any rework, it is more likely that rework will be necessary. In the model, the need for rework is determined probabilistically using the same probability matrix used to specify the probability of rework due to feedback. In practice, the probability of requiring rework due to overlapping activities that were not originally intended to be overlapped, could be expected to be higher than that due to feedback. This will be discussed in section 5.3.

Should it be found that rework is required, the model reverses the overlap reduction. In other words, if rework is required the net benefit of overlapping is zero and the manpower cost is equal to the amount of time of the overlap. This model is conservative in that when rework is required, it is possible that some portion of the original work can be salvaged. This simplification will also be addressed in 5.3. Thus the expected incremental cost of overlapping is:

$$\text{Cost}_{\text{overlap}} = P(\text{rework required for downstream activity}) \times \text{amount of time overlapped}$$

This check for delayed activity starts and the implementation of allowed recovery action is repeated for each activity at every time step of the simulation.

Table 1 summarizes the relevant variables of the original simulation (shown in bold) and the key ones that were added to model schedule recovery.

VARIABLE	DESCRIPTION
W(n)	Work vector containing the remaining amount of work for each activity, measured as a fraction of the original activity duration, i.e. $0 \leq W(x) \leq 1$
DSM(n, n, 2)	DSM ($n \times n$ matrix with third dimension $k=1$ or 2) where: $k=1$ is the matrix of rework probabilities: Superdiagonal entries (i.e. $j > i$ represent the probability that activity j causes feedback rework for activity i , at the time step when activity j is completed). Subdiagonal entries (i.e. $i > j$ represent the probability that activity j causes second-order or feedforward rework for activity i , after activity j is reworked due to feedback). $k=2$ is the matrix of rework impacts: the amount of work (as a fraction of the activity's original duration) added to an activity given that feedback or feedforward rework occurs.
n	Number of activities in the process
r	Current run number
S	Cumulative process duration (S_{final} = project duration)
Δt	Time step used to proceed through process
t	Current time step
MAXCOMP	Maximum amount by which a delay in activity completion can be compressed, specified as a fraction (0 to 1) of the delay
MAXOVERLAP	Maximum amount by which a start of an activity can be overlapped with an incomplete upstream activity, specified as a fraction of the scheduled duration of the downstream activity being overlapped
STARTDATE(n)	Array containing scheduled start time for each activity
OTPREMIUM	Premium paid for overtime work, specified as a fraction of the regular wage rate, e.g. OTPREMIUM for overtime paid at "time-and-a-half" = 0.5
OVERLAP	Fraction of activity duration by which its start is overlapped with an incomplete upstream activity, i.e. $OVERLAP = \min(\text{delay in completion of upstream activity}, MAXOVERLAP)$
Ccompress	Cumulative cost of compression for current run of the simulation (the number of time units "paid" as a premium for using overtime work)
Coverlap	Cumulative cost of overlapping for current run of the simulation (the amount of rework, in time units)

Table 1: Key Variables in the Project Simulation

The algorithm for the schedule recovery extension is shown in Figure 15. The source code can be found in Appendix A.

Read in scheduled start dates for each activity in the project.

Read in maximum allowable compression (% of activity delay can be eliminated using overtime) and overtime premium

Read in maximum amount of activity overlap (% of downstream activity duration that can be overlapped with an incomplete upstream activity)

Loop through all activities...

 If current time is within one time step of the scheduled start date of the activity (say x)...

 Loop through all predecessors of activity x...

 If the predecessor activity (say y) is not finished...

 If compression is allowed (maximum compression > 0)...

 Then reduce the amount of work for activity y by maximum compression efficiency

 Add cost of using compression to cumulative compression cost (number of time units paid as a premium for using overtime)

 If activity y is still delayed and overlapping is allowed (maximum overlap > 0)...

 Then determine the amount of overlap to be used (minimum of remaining work for activity y and maximum overlap)

 Reduce the work remaining of activity x by the amount of overlap

 Randomly determine if rework is required (based on specified probability of occurrence). If required...

 As rework, add the amount of overlap back to the work remaining for activity x

 Add cost of overlap to cumulative overlap cost (time required for rework)

 Check next predecessor activity for delay (y+1)

Check for delay in the start of the next activity (x+1)

Figure 15: Algorithm for schedule recovery extension to DSM project simulation

- Modeling: Effect of rework on project progress -

As discussed in section 2.3.2, the primary shortcoming of the traditional “percent complete” measure of project progress is that it does not account for potential rework. An activity that depends on others is subject to rework, and its actual duration will therefore be longer than an estimate that does not include the potential for rework. The simulation can provide estimates for the expected duration of each activity in the project that include rework, i.e. by running the simulation, the total amount of work for each activity can be determined. Calculating the percent complete of an activity using this longer expected duration is a more representative measure of project progress.

To assess the error of ignoring rework when measuring project progress, the “percent complete” throughout the project was calculated (after running the simulation) in both ways: ignoring rework by assuming that the duration of activity would be as scheduled, and then including rework by using the total work done on each activity (determined by the simulation) as the expected duration.

Recall that the project simulation randomly samples the initial duration for each activity (from a user-specified distribution of minimum, most likely, and maximum durations). This initial duration is the time that the activity is expected to take, before the occurrence of any rework. To isolate the effect of rework on project progress, the confounding effect of this variation in initial activity durations was eliminated by fixing the initial duration of each activity to that used in the company’s standard development timeline.

At a broader level of analysis, the total project durations for various rework probability and impact scenarios were compared to the no-rework case to determine the “shadow factor” for each scenario. This shadow factor is simply:

$$\text{Shadow Factor} = \frac{\text{Simulated Project Duration}}{\text{Scheduled Project Duration}}$$

Because the simulation models the effects of rework, the simulated project duration accounts for rework. The scheduled project duration is simply the sum of the activity durations along the project’s critical path. This shadow factor is then representative of the amount of time rework adds to the scheduled project duration.

4.3.3 Discussion of Results: Demonstrating the Improvement Approaches

In order to make the following discussion more concise, some definitions would be helpful. The term “project” refers to the standard vehicle development project described in section 4.3.1.

TERM:	DEFINITION:
Scheduled project duration	Sum of critical path activity durations from the company’s standard timeline (49 time units)
Project with variation	Initial activity durations (before any rework additions) are randomly sampled from a user-specified triangular distribution (minimum, most likely, and maximum duration)
Probability of rework (P)	Probability that the completion of an activity forces rework for a related activity
Impact of rework (I)	Amount of rework added to an activity if rework is required (as a fraction of the initial activity duration)
Feedforward (subscript f)	A relationship where an upstream activity affects a related downstream activity (subdiagonal entry in the DSM)
Feedback (subscript b)	A relationship where a downstream activity affects a related upstream activity (superdiagonal entry in the DSM)
Benchmark project	A project with variation (see above) and the following rework probabilities and impacts: $P_f = 50\%$ (feedforward probability of rework) $P_b = 10\%$ (feedback probability of rework) $I_b = I_f = 10\%$ (feedback and feedforward impact of rework) (see DSM representation of this project in Figures 13 & 14 of section 4.3.1.) The benchmark project did not allow any schedule recovery actions (compression or overlapping)
Activity delay	Amount of work remaining (in time units) for the activity when a dependent or interdependent downstream activity is scheduled to start
On-time project completion	Project completed within scheduled duration (49 months or earlier)
90% confidence date	Number of time units within which 90% of the simulation runs complete the project (90 th percentile project duration)

- Results: Effect of rework on project completion -

In order to provide a baseline for further scenarios of rework probability and impact, the project was run with variation in the initial activity durations, but without allowing any rework or iteration. On a percentage basis, the amount of variation allowed in each activity duration was the same.

The input parameters used in the simulation were:

PARAMETER:	VALUE:
Most likely activity duration	Duration documented in company's standard timeline
Minimum duration	Most likely duration less 5%
Maximum duration	Most likely duration plus 10%

It was felt that the skewed distribution was more representative of reality, given that the standard durations were rather optimistic. The maximum duration variation of +10% was chosen to be on the conservative side. Examples of four actual delays in past programs were cited for "Validate Designs", "Preliminary Package", "Final Appearance", and "CP Testing". These delays ranged from +80% to +400% of the scheduled activity duration.

Compared to a project without variation, a simulation allowing this amount of variation reduced the probability of completing the project on schedule (i.e. at 49 time units) from 100% to only 12%. See Chart 1 below. The 90% confidence completion date was 50.5 time units (measured from the beginning of the first activity of the project), i.e. 90% of the time, the project duration can be expected to be 50.5 time units or less. Thus the addition of variation to the standard activity durations drastically reduces the probability of meeting scheduled completion date.

To determine the effect of rework on project completion, the simulation was run for the benchmark project which has feedback rework probabilities of 10%, feedforward rework probabilities of 50%, and rework impacts of 10% (see definition at the beginning of this section). With this amount of potential rework, the probability of completing the project on schedule was reduced from 12% to 8%. Thus even with the small chance of feedback and "tight" feedback loops of the benchmark project, the probability of meeting scheduled completion date is significantly reduced. Chart 1 summarizes the cumulative probability of completing the project on or before a given number of time steps, for cases where potential rework is ignored and where it is accounted for. Both projects allowed variation in the initial activity durations.

Effect of Rework on Project Duration Probability of Being Earlier than...

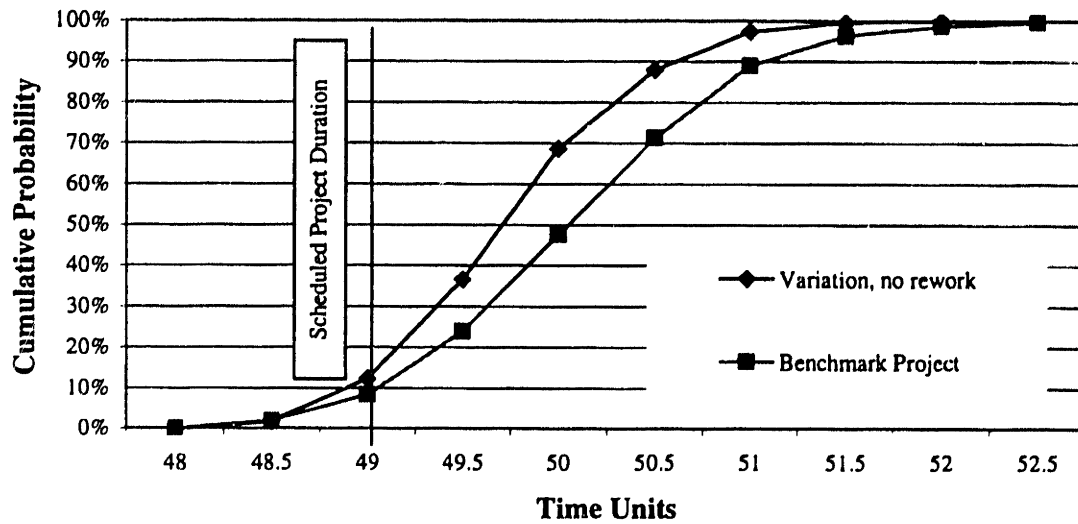


Chart 1: The effect of potential rework on project duration

To determine the sensitivity of the model to changes in the DSM probabilities, the simulation was run with different feedforward and feedback probabilities in the benchmark project. The same activity dependencies in the benchmark DSM were used but the values were varied. The rework impacts of the benchmark project were maintained. Chart 2 summarizes the cumulative probability of completing the project on or before a given number of time steps for the benchmark case (feedbacks with 10% probability, feedforwards with 50% probability, feedback and feedforward impacts 10%), higher feedforward probability (60%), higher feedback probability (20%), and higher feedforward and feedback (60% and 20% respectively).

Sensitivity of Project Duration to Probability of Rework

Probability of Being Earlier than...

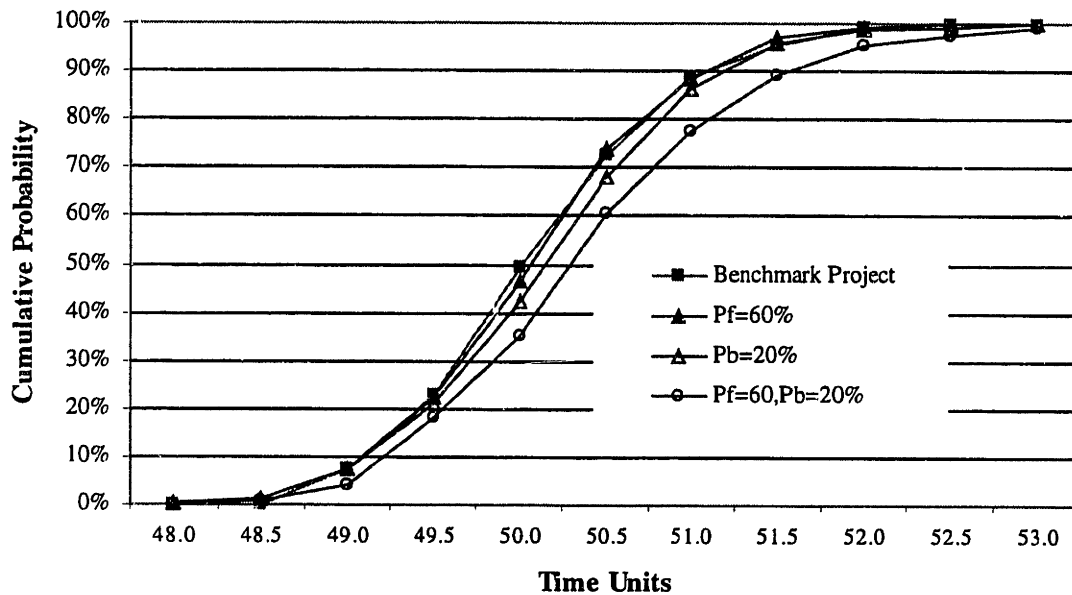


Chart 2: Sensitivity of project duration to probability of rework (P_f and P_b)

The chart suggests that in general, project completion date is not highly sensitive to an increase in feedback or feedforward probability. As expected, project completion date was more significantly affected by feedback probability. This is because feedforward probabilities do not have the potential to cause second-order rework until rework caused by feedback occurs. The combined effect of increases in both feedback and feedforward probabilities was significant, reducing the probability of completing on schedule from 8% to 4%. The 90% confidence completion date was increased from 51 to 51.5 time units.

It is worth noting that although the sensitivity of project completion date does not appear to be high, i.e. the curves are close to each other, the high cost of lost sales associated with late project completion makes small differences significant. To put this into perspective, the expected or mean value of lost profit can be determined by integrating the area under the cumulative probability curve past the scheduled completion date of 49 time units. An example calculation is detailed in Appendix B, but if it is assumed that lost profit "costs" the program \$1 million/day, the increase in mean lost profit between the benchmark project and one with increased feedback and feedforward probabilities (60% and 20% respectively) is \$6 million.

As a reference point, the expected or mean lost profit for the benchmark project was \$25 million, based on the distribution of project completion dates from the simulation. As discussed in section 4.3.2, a project with poorly defined give/gets (information exchanges between activities) could be an example of such a situation where the probability of rework is increased.

As discussed in section 4.3.2, a change in design requirements can be modeled as a feedback loop with probability one. In the scenario shown below, a change was made after “Set Specifications-3” requiring 25% of the “Appearance Concepts” work to be repeated (and any possible second order rework for “Final Appearance”). The added loop is shown in the Figure 16 below.

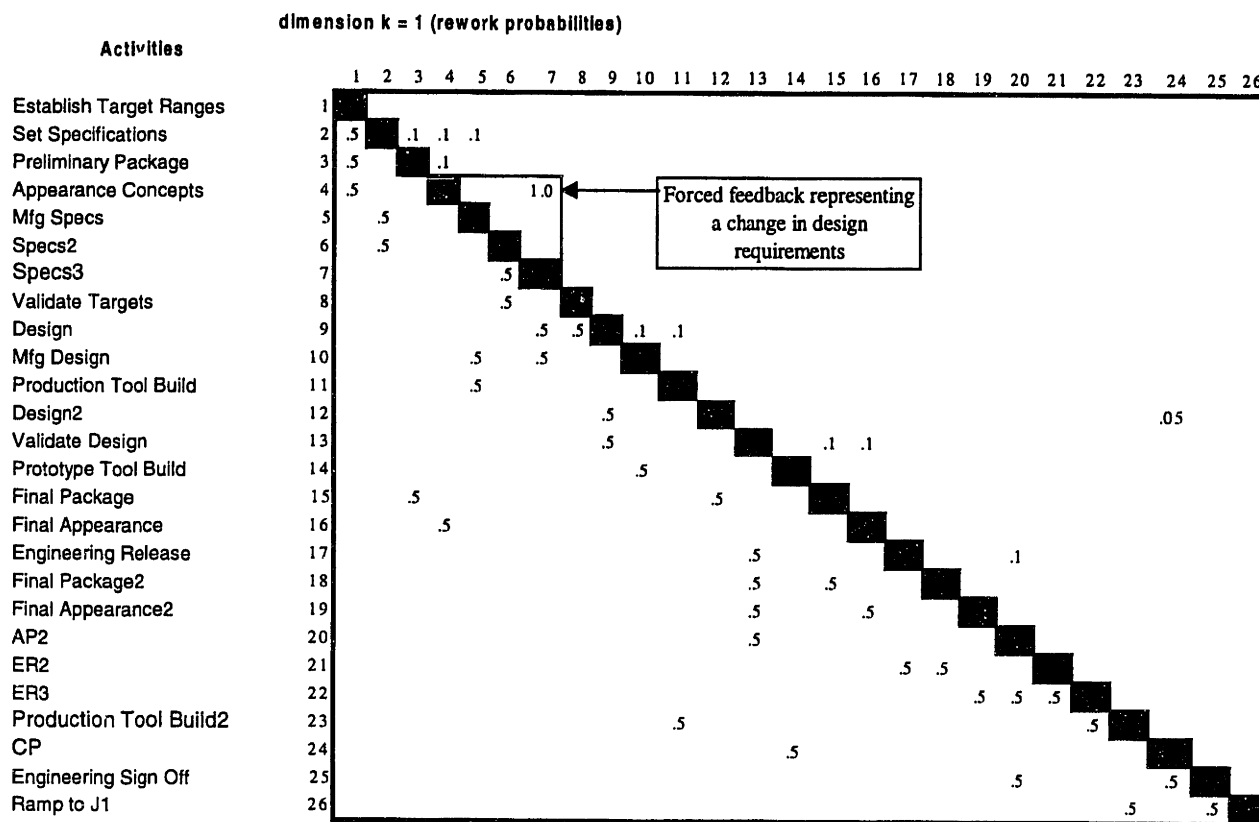


Figure 16: DSM Rework probability matrix showing model for change in design requirement

Chart 3 illustrates the results of a simulated change in design requirements, compared to the benchmark project. The change in design requirements reduces the probability of completing the project on time from 9% to 0%. The 90% confidence completion date is increased from 51 to 52 time units. The increase in

mean lost profit was \$20 million. This conceivable change in the project clearly has a very significant cost associated with it.

Effect of Change in Design Requirements on Project Duration
Probability of Being Earlier than...

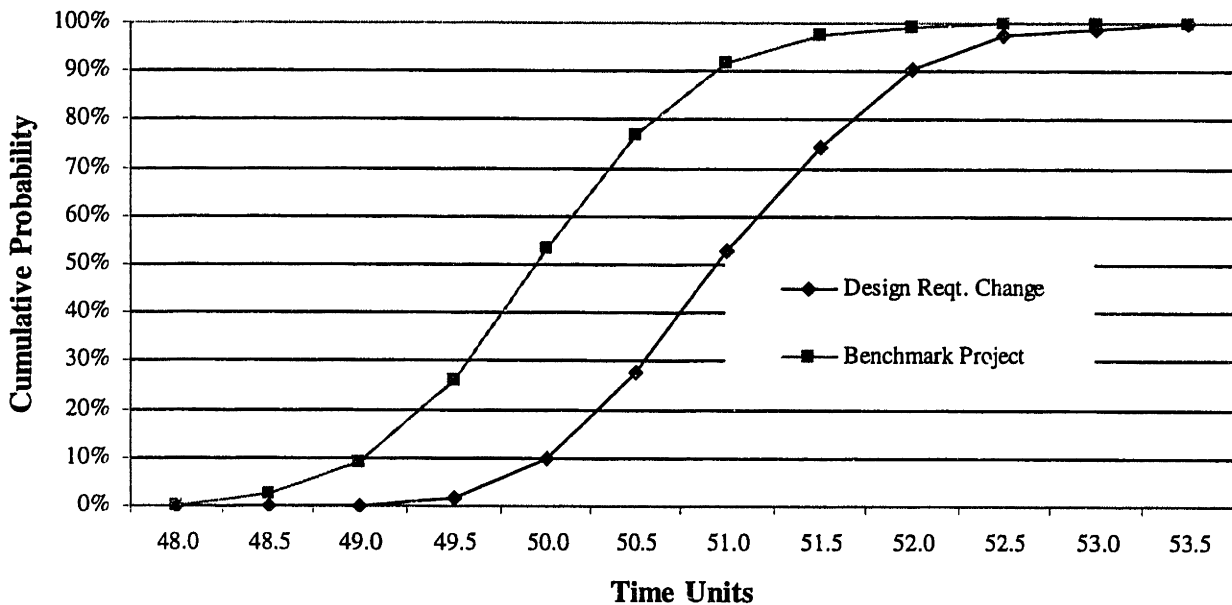


Chart 3: The effect of a change in design requirements on project duration

To determine the sensitivity of the model to changes in the DSM rework impacts, the simulation was run with different feedforward and feedback impacts in the benchmark project (recall that impact is the amount of rework added to a related task, as a fraction of the original duration). Chart 4 summarizes the cumulative probability of completing the project on or before a given number of time steps for the benchmark case (feedback and feedforward impacts 10%, feedback and feedforward probabilities 10% and 50% respectively), higher feedback impact (20%), higher feedforward impact (20%), and higher feedback and feedforward impacts (each 20%).

Sensitivity of Project Duration to Rework Impact

Probability of Being Earlier than...

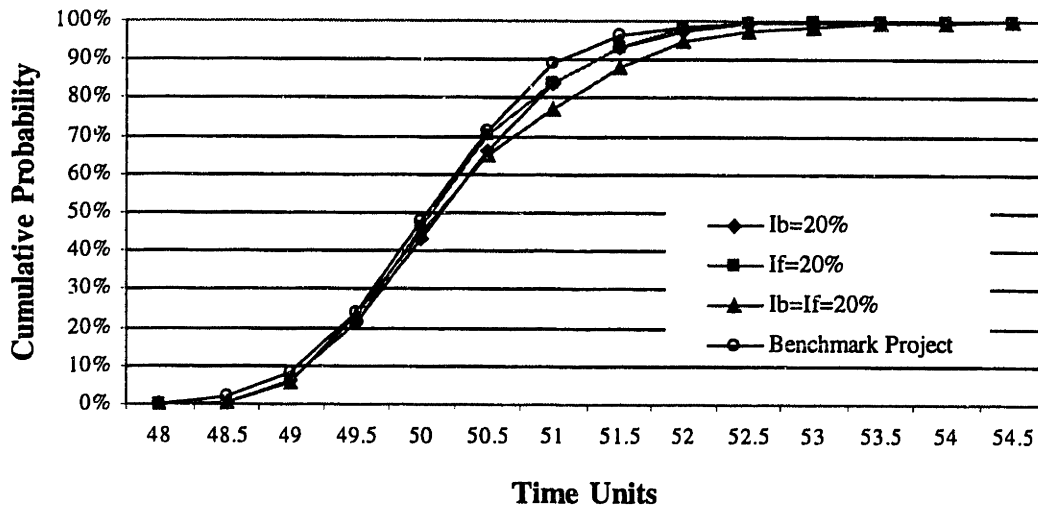


Chart 4: Sensitivity of project duration to impact of rework (I_b and I_f)

The observed results are similar to those for variation in rework probability. The sensitivity to a change in only one of feedback or feedforward impact is relatively insignificant. Increasing both however, reduces the probability of completing the project on time from 8% to 6% and increases the 90% confidence completion date from 51 to 51.5 time steps. The increase in mean lost profit between the benchmark project and one with higher feedback and feedforward impacts (each 20%) is \$1 million. As discussed in section 4.3.2, a project with late problem discovery could be an example of such a situation where the impact of rework is increased.

The fourth controllable cause of rework mentioned was weak design freezes. As discussed in section 4.3.2, a weak design freeze could be modeled as an increase in both the probability and impact of rework. Chart 5 compares the cumulative probability of completing the project on or before a given number of time units for the benchmark project and one in which the probability and impact of feedback rework is higher (each 20%, compared to 10% in the benchmark project).

Effect of a Weak Design Freeze on Project Duration

Probability of Being Earlier than...

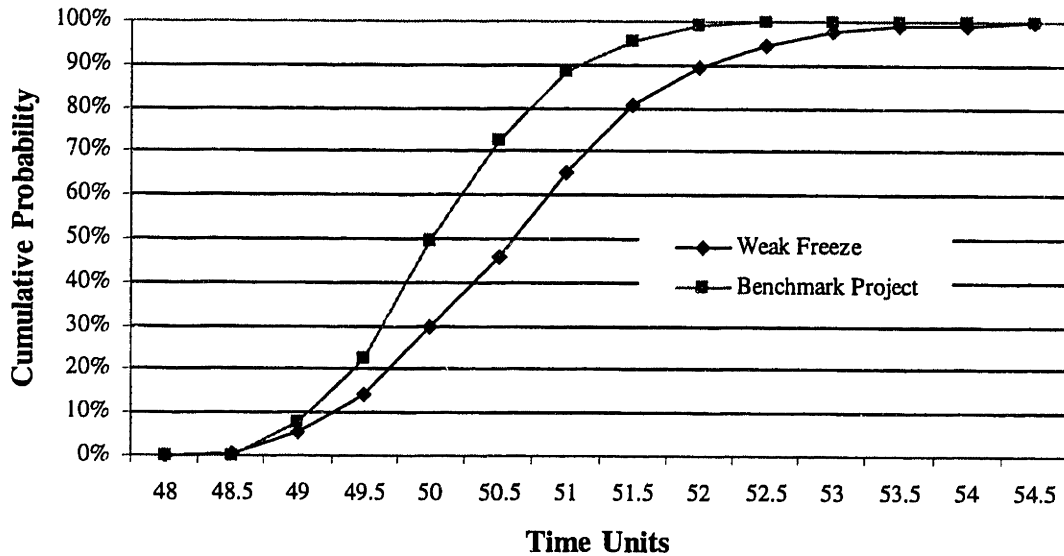


Chart 5: Effect of a weak design freeze on project duration

The weak freeze reduced the probability of completing the project on time from 8% to 4% and increased the 90% confidence completion date from 51 to 51.5 time units. The increase in mean lost profit was \$10 million. Thus the commonly accepted occurrence of design freezes that “drift” beyond their declared finalization date can incur a rather significant cost.

Generalizing all the cases discussed above, two conclusions can be drawn from the simulation results. Perhaps most significantly, the simulation shows that allowing for variation in activity durations and ignoring rework, the likelihood of completing the project within the standard schedule is low. Secondly, given the significant lost profit associated with small increases in project completion dates, the simulation suggests that it would be valuable to reduce controllable causes of rework. While the simulations were based on general estimates of the strength of the relationships between activities, using historically-based parameters would make the results of the simulations more representative. The collection of real project delay data to determine these parameters was the subject of section 4.2.

- Results: Efficacy of Recovery Actions -

Given the likelihood of a delay in the project, the next questions to answer were:

- What activities are likely to be delayed, and
- What is the probability that schedule recovery actions will be able to recoup the delay and keep the project on schedule?

Chart 6 summarizes the range of delay (minimum, mean, and maximum observed in the simulation runs) in each of the project activities from a 300-run simulation. The simulation was run with parameters set for the benchmark project (feedforward and feedback probabilities 50% and 10% respectively, feedforward and feedback impacts each 10%). Recall that activity delay equals the work remaining (in time units) in the activity when a dependent or interdependent downstream activity is scheduled to start.

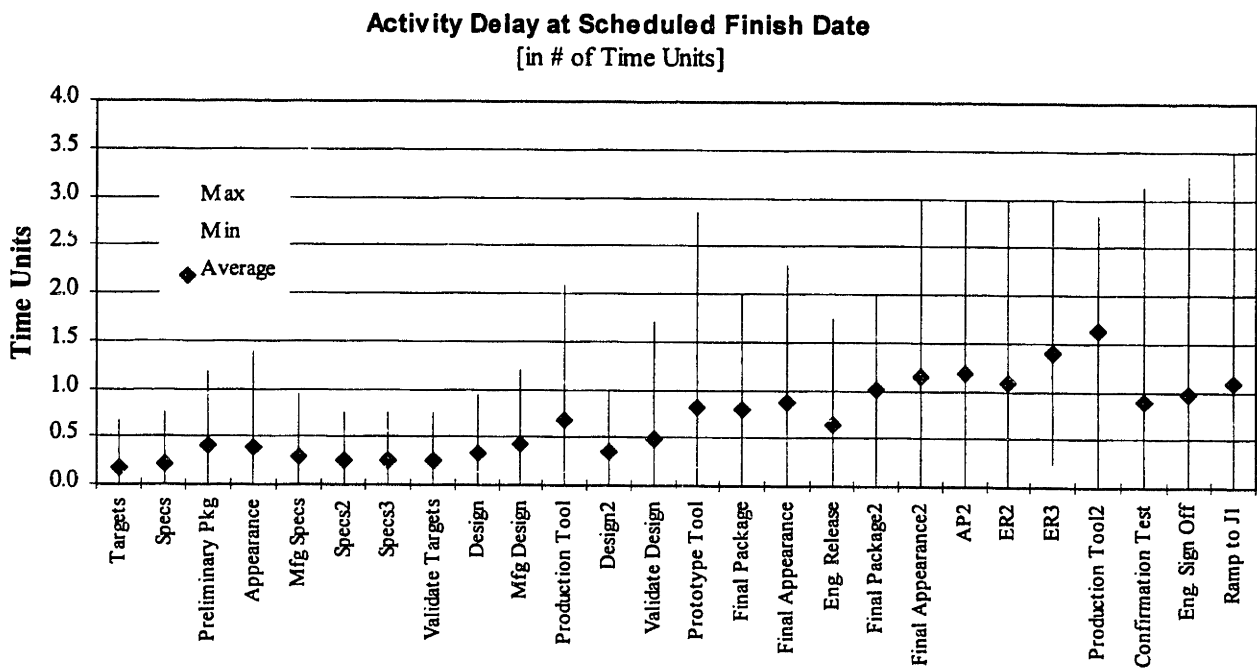


Chart 6: Delay in activities at their scheduled finish dates [in units of time]

As expected, the mean and maximum delay increases with original duration and the number of predecessors. As the chart shows, the net effect is that the delay is “pushed” toward the end of the program, which is clearly not the ideal situation. Delay late in the program means there is less time to take remedial action and make contingency plans.

The second observation from the chart is that the means for each activity are skewed slightly toward the minimum rather than the maximum of the delay range. This is consistent with the skewed variation in the initial activity durations used in the simulation which was described at the beginning of this section.

Chart 7 shows the same data expressed as a fraction of the original activity durations. Normalizing the delay in this way shows more clearly the effect of dependencies (feedforward and feedback) on the various activities. Without dependencies, one would expect the delay ranges for each activity to be equal since the same percentage variations were set for each activity in the simulation. The more direct dependencies an activity has, or the more dependencies it has on activities which in turn have many direct dependencies, the greater the expected delay.

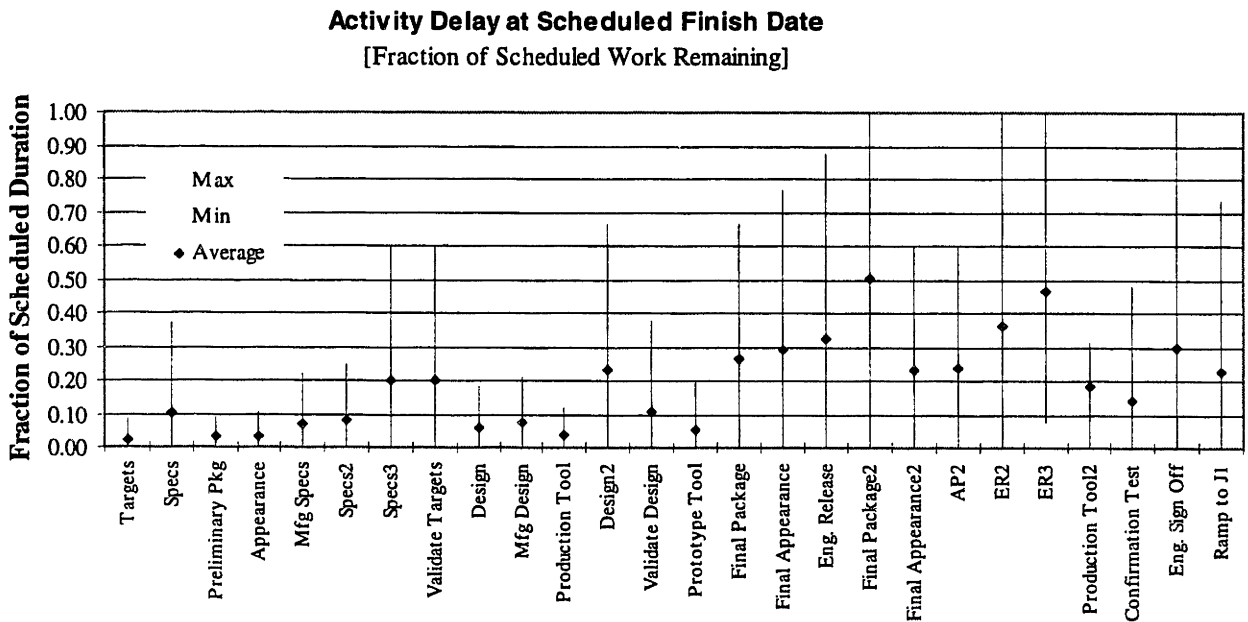


Chart 7: Delay in activities at their scheduled finish dates [as fraction of original work]

To assess their ability to maintain scheduled project completion date, three different types of recovery strategies were simulated: compression only, overlapping only, and combination strategies. The efficacy of each strategy was determined by considering its ability to increase the likelihood of completing the project on time, and the manpower costs associated with using it.

Charts 8 and 9 summarize the effect of various compression efficiencies on project completion date and manpower cost. Recall from section 4.3.2 that maximum allowable compression refers to the fraction by which a delay in an activity can be reduced using overtime. Simulations were run for maximum allowable compressions of 10%, 15% and 20%. As a practical note, it is generally accepted that the duration of an activity cannot be reduced by more than 20% using compression. The benchmark project (which, as defined at the beginning of this section, does not allow any schedule recovery action) is shown for comparison.

Effect of Compression Recovery on Project Duration
Probability of Being Earlier than...

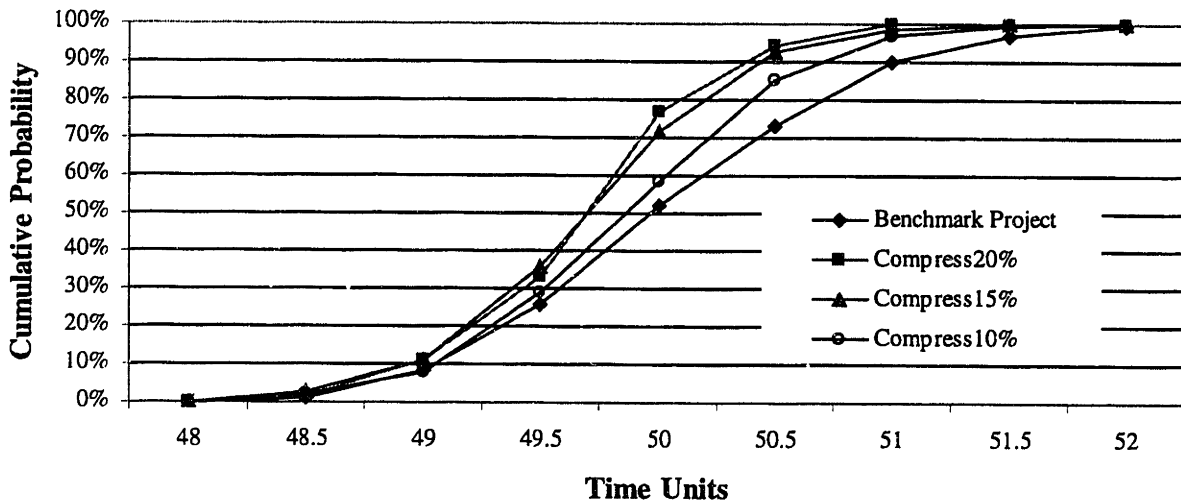


Chart 8: Effect of compression recovery on project duration

Effect of Compression Effectiveness on Overtime Cost

Probability of Costing Less than...

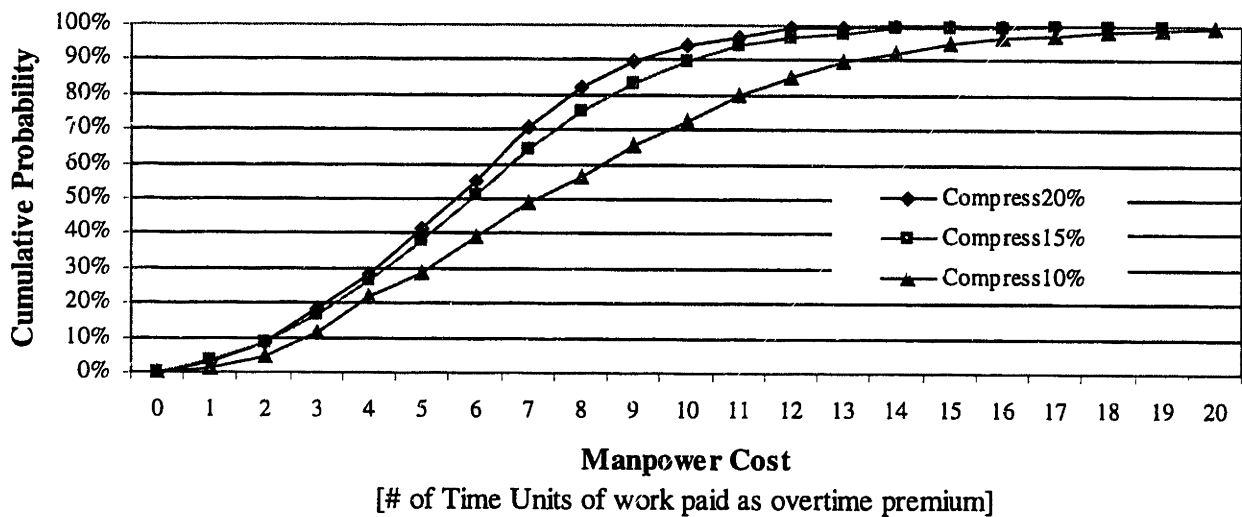


Chart 9: Effect of compression on overtime cost

Chart 8 shows that the probability of late project completion decreases with compression efficiency, which was expected since the greater the fraction of delay that can be eliminated using compression, the shorter the delay in the finish of a given activity. The shorter the delay in an activity, the lower the impact on project duration and the shorter the time interval over which overtime premium is paid. Increasing maximum allowable compression from 10% to 20% does not change the probability of completing the project on schedule (8%) but reduces the 90% confidence completion date from 51 to 50.5 time units.

It is obvious that manpower cost will decrease as maximum allowable compression increases. Recall from section 4.3.2 that the incremental manpower cost (in units of time) of using compression is:

$$\text{Cost}_{\text{compression}} = \text{Overtime premium} \times (\text{Delay} - \text{Fraction of delay reduced by compression}).$$

Focussing on the relationship between project duration and manpower cost, we find that while the mean (and maximum) project durations and manpower costs decrease as maximum allowable compression increases, the percent change in project duration is relatively insignificant compared to the percent change in manpower cost.

For example, increasing maximum allowable compression from 10% to 20% decreased mean and maximum project durations by less than 1%, while mean and maximum manpower cost decreased by 27% and 31% respectively. As a reference point, the time step used in the simulation was 0.25 time units. Thus the possible error associated with being off one time step would be 0.5% (0.25 ÷ standard project duration of 49 units). Clearly the maximum amount of compression that can be achieved has a significant effect on manpower cost.

Charts 10 and 11 summarize the effect of various amounts of overlapping on project completion date and manpower cost. Recall from section 4.3.2 that the amount of overlapping is specified as a percentage of the downstream activity duration by which the start of that activity can be overlapped with an incomplete upstream activity. The higher the percentage of overlapping, the more concurrent the execution of the activities. Simulations were run for overlap percentages of 5%, 10%, and 15%. It was felt that overlap of more than 15% for activities not originally planned to be overlapped would not be feasible. The benchmark project (which, as defined at the beginning of this section, does not allow any schedule recovery action) is shown for comparison.

Effect of Overlap Recovery on Project Duration
Probability of Being Earlier than...

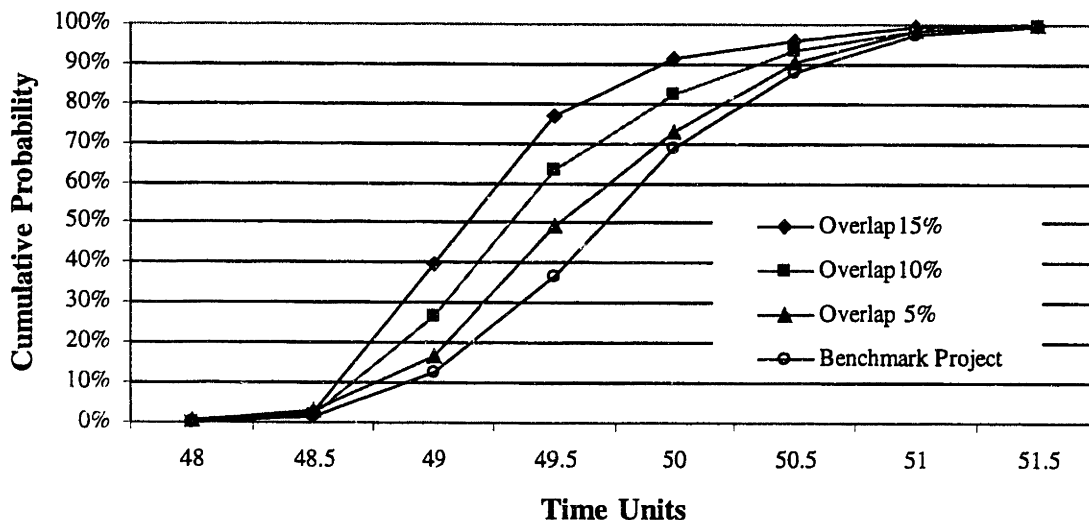


Chart 10: Effect of overlap recovery on project duration

Effect of Overlapping Recovery on Rework Cost

Probability of Costing Less than...

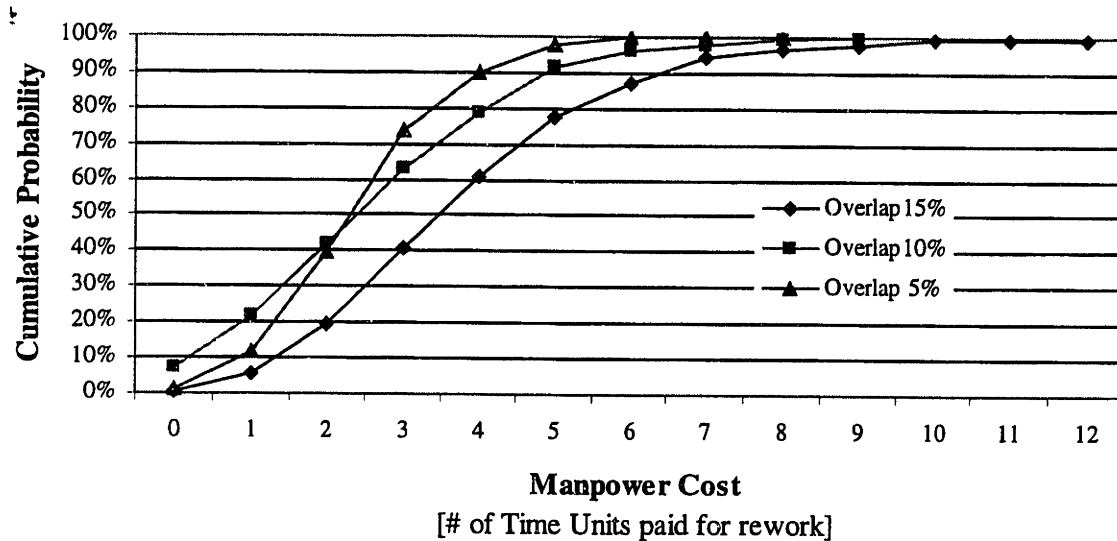


Chart 11: Effect of overlap recovery on rework cost

As seen in Chart 10, the effect of various amounts of overlapping on the project is more interesting than the compression case. This is because the use of overlapping carries with it a probability for rework. In cases where rework is required, the model incurs the manpower cost but the delay is not reduced. Thus, unlike the model of compression, it is not always true that increasing the amount of overlapping will reduce project duration. Since this rework is probabilistic, simulation is a valuable tool for predicting results. If it were possible to increase percent overlap from 5% to 15%, the probability of completing the project on time would be increased from 16% to 40%. The 90% confidence completion date would be reduced from 50.5 to 50 time units.

Recall from section 4.3.2 that the incremental manpower cost (in units of time) of using overlapping is:

$$\text{Cost}_{\text{overlap}} = P(\text{rework required for downstream activity}) \times \text{amount of time overlapped.}$$

Considering the relationship between project duration and manpower cost, while mean project duration decreases slightly for increasing amounts of overlap, the mean and maximum manpower cost increase monotonically. For example, increasing overlap from 5% to 15% decreases mean and maximum durations by less than 1% but increases mean and maximum manpower costs by 58% and 125% respectively.

Further increasing overlap to 25% did not reduce project duration but mean and maximum manpower costs continued to increase.

In selecting the best overlapping strategy, the maximum and expected lost profit were compared to maximum and expected manpower cost. The best strategy was defined as the one with the lowest expected sum of lost profit and cost of recovery and an acceptable maximum. An example calculation can be found in Appendix B, but the results are summarized in the Chart 12. The largest portion of each bar is the expected profit loss. The mean manpower costs associated with using the particular amount of overlapping is stacked on top of (added to) the lost profit. The dollar value labels on top of each bar refer to the manpower costs (in millions of dollars).

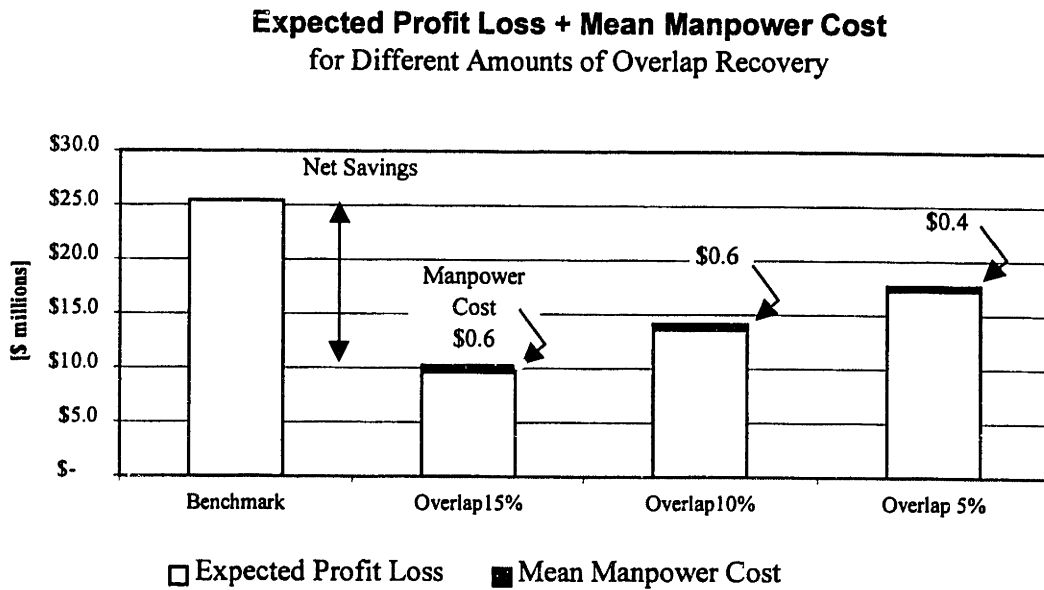


Chart 12: Expected cost outcomes for different amounts of overlap recovery

Having tested the two “pure” strategies, several combinations were also simulated. Taking the best (lowest project duration and lowest cost) feasible compression strategy which was 20% compression, various amounts of overlapping were added to see if the duration and/or cost could be further improved.

Charts 13 and 14 summarize the effect of increasing amounts of overlap (5%, 10% and 15%) added to 20% compression on project duration and cost. The benchmark project (which, as defined at the beginning of this section, does not allow any schedule recovery action) is shown for comparison.

Effect of Combined Recovery on Project Duration

Probability of Being Earlier than...

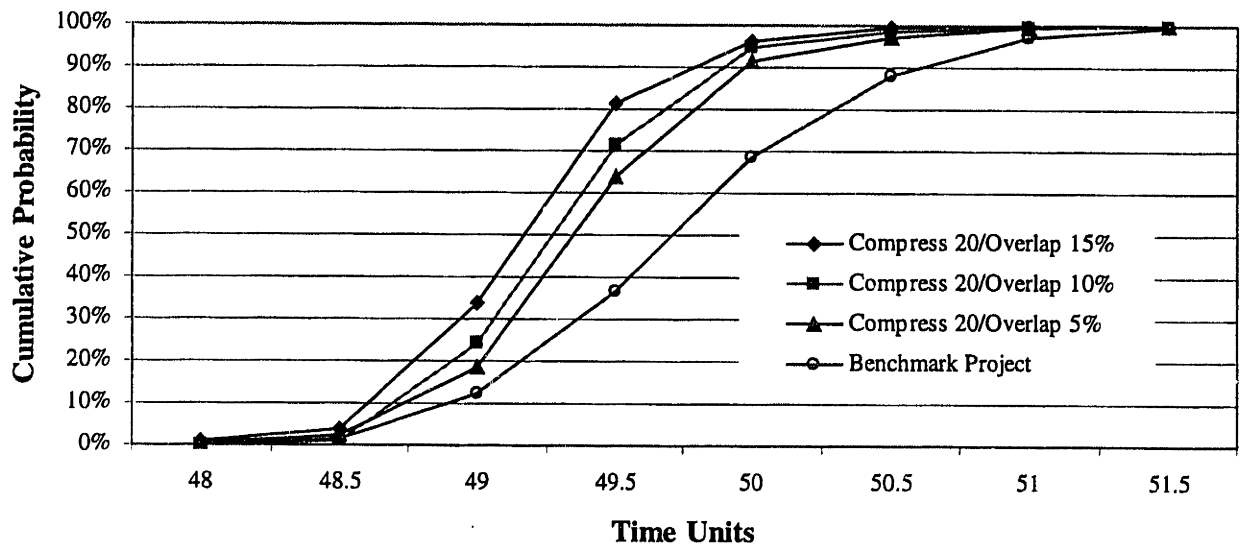


Chart 13: Effect of combined recovery on project duration

Effect of Recovery Combination on Cost

Probability of Costing Less than...

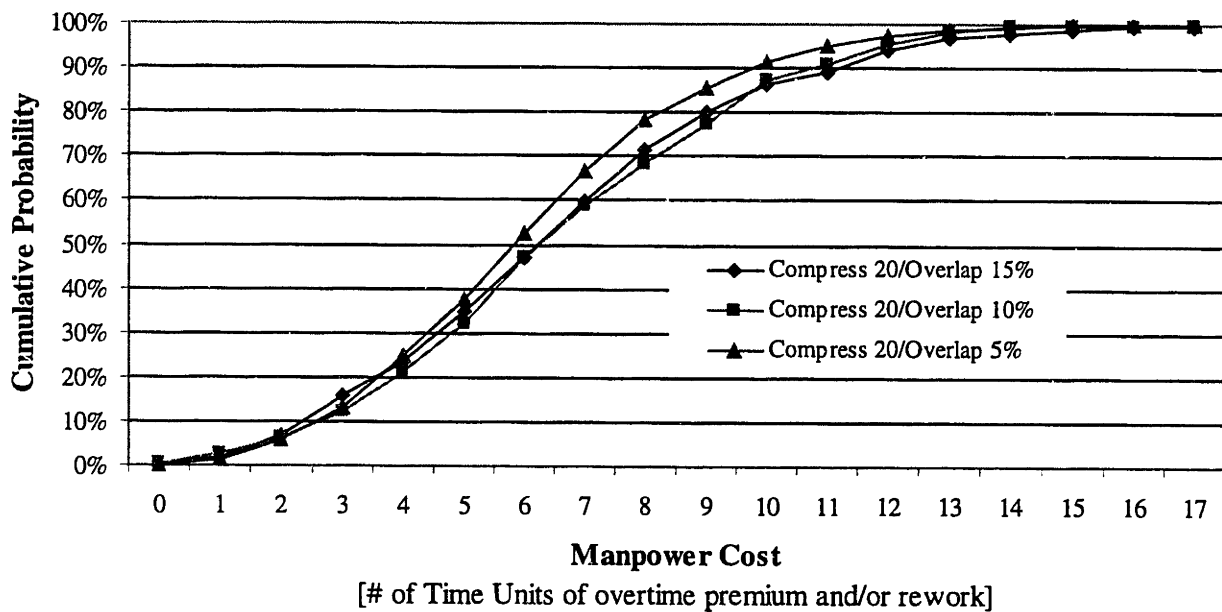


Chart 14: Effect of recovery combination on cost

Given 20% compression, increasing overlap from 5% to 15% increased the probability of completing the project on time from 19% to 34% but did not change the 90% confidence completion date. And, consistent with the “pure” recovery strategies discussed earlier, we see that the percent decrease in mean and maximum project duration (less than 1%) is much smaller than the increase in mean and maximum manpower cost (7% and 19% respectively). In selecting the best combination strategy, the maximum and expected lost profit were compared to maximum manpower cost.

Chart 15 compares the best of each of the three strategies, showing the expected lost profit and recovery costs. The maximums are shown in Table 2. Defining the optimal recovery strategy as the one with the lowest sum of mean lost profit and manpower cost, and with an acceptable maximum, the optimal strategy for the project described by the benchmark DSM is use of 15% overlapping and compression (20% maximum allowable compression). Of course, with historically-determined parameters from the delay analysis (see section 4.2) and activity-specific recovery strategies (see section 5.3.1), more representative conclusions could be drawn using the approach outlined here.

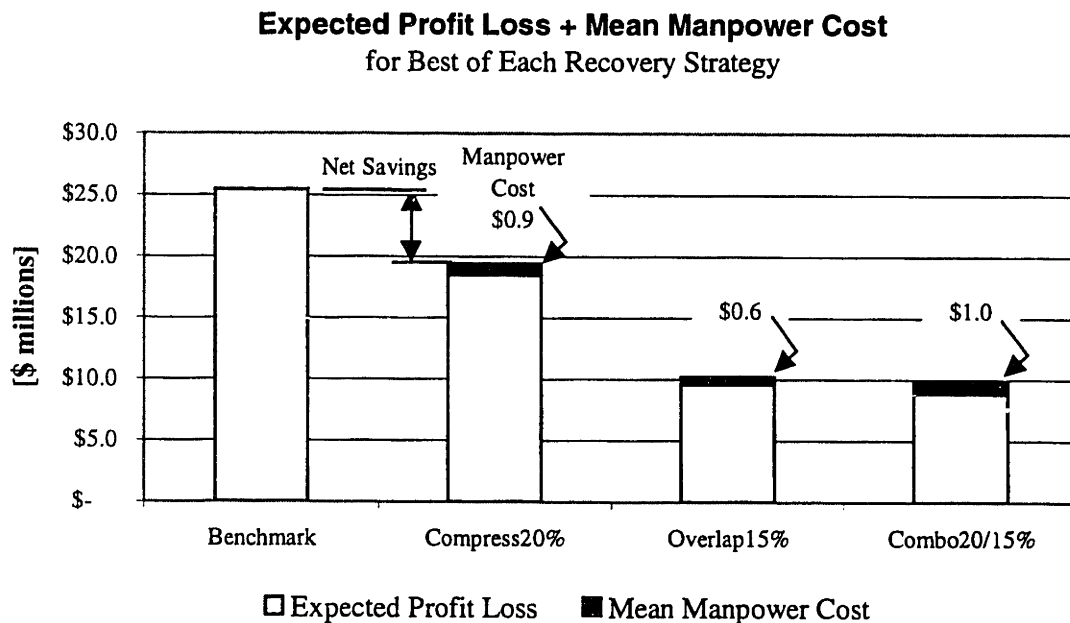


Chart 15: Expected cost outcomes for best of each recovery strategy

RECOVERY STRATEGY	MAXIMUM PROFIT LOSS [\$ millions]	MAX. MANPOWER COST [\$ millions]
Benchmark Project: No Recovery	\$70	n/a
Compress 20%	\$50	\$2.5
Overlap 15%	\$50	\$2.0
Combination: 20% Compression 15% Overlap	\$40	\$2.9

Table 2: Maximum profit loss and manpower costs for various recovery strategies

There were two key observations from this analysis of recovery strategies. Most significantly, even the best strategy cannot achieve on-time project completion more than 34% of the time. This would suggest that given the parameters used for the simulation, relying on recovery actions to keep a project on schedule is a high-risk bet.

Whether the low probability of completing on time is due to unrepresentative relationship estimates, poor activity duration estimates, or the contribution from controllable delay causes – most likely a combination of these – the first step to improving the process is to understand what the primary causes of delay are. This is the objective of the delay analysis tool described in section 4.2. Another factor that contributes to the low probability of completing on time is the delay in implementing recovery action. In the model, as is typically the case in reality, tracking only activity starts and finishes mean that recovery action is not taken until an activity start is already delayed. The development of a more-leading measure of project progress is described in section 5.3.2.

The second observation was that given the approximate costs of lost sales and extended manpower use, the costs using recovery were much smaller than the costs of lost sales in the simulated projects. The size of this difference early depends on the strategic importance of completing on time and the relationship parameters in the model. The key point is that given validated parameters for a particular company, the model allows the trade-off analysis to be made quickly and easily.

- Results: Effect of Rework on Project Progress -

Because it accounts for potential rework, another use for the project simulation would be to make the “percent complete” measure of project progress a more representative metric. As described in section 4.3.2, the simulation can be used to estimate the amount of expected rework in the project activities, which can in turn be used for a more accurate determination of percent complete.

Chart 16 illustrates the difference in the “percent complete” measure of project progress when the potential for rework is ignored and when it is included. The curves show the cumulative probability of completing the project within a given number of time units for the project as scheduled (no potential rework), and for the benchmark project (feedback and feedforward rework probabilities 10% and 50% respectively, feedback and feedforward rework impacts 10% each). In either case, to isolate the effect of rework on project progress, no variation in the initial activity durations were allowed (see section 4.3.2).

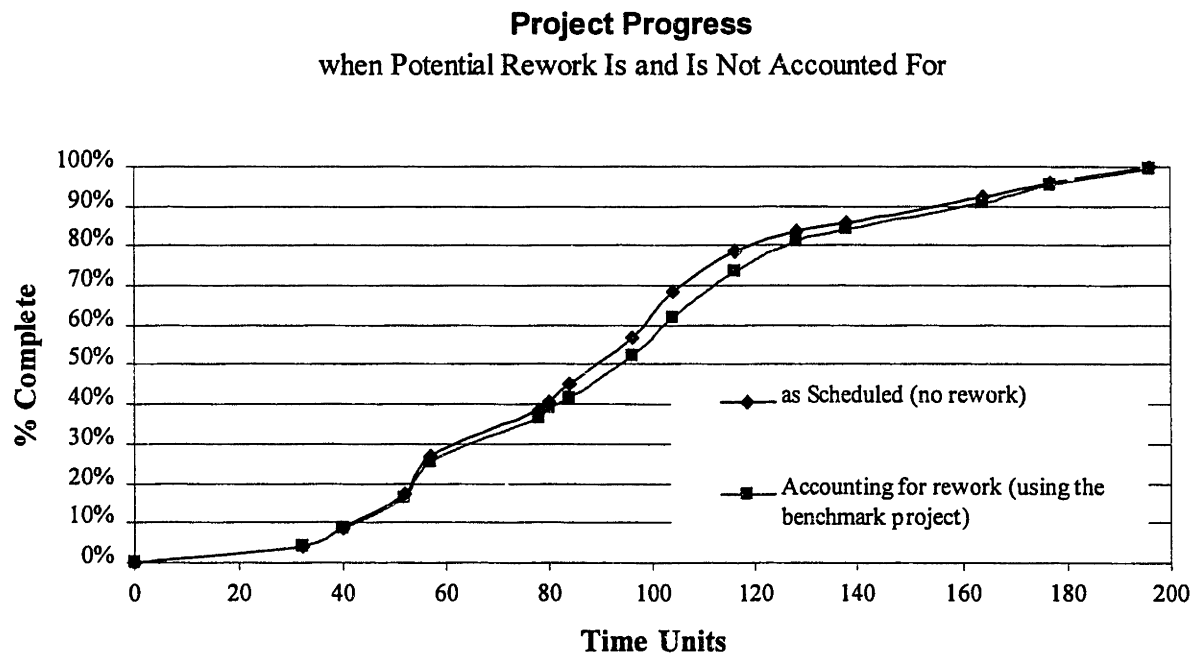


Chart 16: Project Progress when potential rework is and is not accounted for

Note that the benchmark project represents a relatively small potential for rework (only 10% probability of feedback, and “tight” feedback loops, i.e. ones that do not jump back very many activities). But even

with this small potential of rework, there was a significant effect on project progress – as much as 9% (at time step 116). It is interesting to note that for this project DSM, the difference in project progress was only in the middle of the project, indicating that most of the rework occurred in the middle activities. Given larger feedback loops (such as a change in design requirements), the difference would be even greater.

Thus it can be seen that the effect of potential rework on project progress is significant. Ignoring potential rework results in an inflated assessment of project progress which can lead to project management decisions which would not have been made had the actual (lower) project progress been known. Examples of such decisions would be accepting a change in design requirements under the impression that there is more time available than there is, or deciding that recovery action is not necessary, when in fact it is.

Another way in which the effect of potential rework on project duration can be assessed is by dividing the mean project duration for a given scenario by the scheduled project duration to compute a “shadow factor” for the scenario. Recall from section 4.3.2 that a project’s shadow factor (simulated project duration ÷ scheduled project duration) captures the amount of rework or iteration in the project. Table 3 summarizes the shadow factors for a selection of the various scenarios simulated. The effects of the three types of recovery strategies are also included.

SCENARIO	MEAN PROJECT DURATION [time units]	SHADOW FACTOR
Scheduled Project Duration: <ul style="list-style-type: none"> No duration variation No rework 	49	1
Rework Effect: <ul style="list-style-type: none"> No duration variation Benchmark rework 	49.24	1.005
Variation Effect: <ul style="list-style-type: none"> Duration Variation No Rework 	49.85	1.017
Variation+Rework: <ul style="list-style-type: none"> Duration Variation Benchmark Rework Probability (Pf=50%, Pb=10%) and Impact (Ib=If=20%) 	50.19	1.024
Poor Give/Get: <ul style="list-style-type: none"> Duration Variation Increased Rework Probability (Pf=60%, Pb=20%) 	50.47	1.031
Design Requirement Change: <ul style="list-style-type: none"> Duration Variation Change in "Specs" requiring 25% rework in "Appearance Concepts" 	51.11	1.043
Late Problem Discovery: <ul style="list-style-type: none"> Duration Variation Increased Rework Impact (If=Ib=20%) 	50.40	1.029
Weak Design Freeze: <ul style="list-style-type: none"> Duration Variation Increased Rework Probability (Pb=20%) and Impact (Ib=20%) 	50.76	1.037
Use of Compression Schedule Recovery: <ul style="list-style-type: none"> Duration Variation Benchmark Rework 20% Compression Efficiency (activity delays reduced by 20%) 	49.79	1.016
Use of Overlapping Schedule Recovery: <ul style="list-style-type: none"> Duration Variation Benchmark Rework 15% Overlap (15% of downstream activity overlapped with incomplete upstream activity) 	49.34	1.007
Use of Compression + Overlapping Schedule Recovery: <ul style="list-style-type: none"> Duration Variation Benchmark Rework 20% Compression Efficiency (activity delays reduced by 20%) 15% Overlap (15% of downstream activity overlapped with incomplete upstream activity) 	49.30	1.006

Table 3: "Shadow factors" for various simulated project scenarios

To put the significance of the differences in project duration into perspective, consider that an estimate of the profit loss associated with one time unit was \$20 million.

This page is intentionally blank.

5. Conclusions & Recommendations

5.1 Conclusions

Speed and flexibility in product development are becoming increasingly important bases for competition in the automotive industry – as well as many others. Concurrent engineering has been accepted as an approach to reducing development lead time while set-based design is being advanced as an approach to maintaining flexibility during development.

Regardless of the stage of implementation of either or both of these approaches, control over development lead time is key to being able to increase the efficiency of the development process. It provides direction for reducing the probability of delay and reduces the uncertainty in speed-flexibility trade-off decisions.

Two broad issues that inhibit control of development lead time were found at the company:

- Design churning and the difficulty in accurately measuring activity progress predispose the program toward reacting to schedule delay – as opposed to taking proactive action to avoid delay.
- Lack of schedule credibility creates a lack of urgency that perpetuates the need for reactive schedule recovery

Based on the relationship between these issues, three improvement efforts were recommended:

- track and address the controllable causes of delay
- optimize schedule recovery actions, and
- develop a more representative measure of project progress.

Collection of data that describes the performance of the process and identifies the root causes of problems is the first step in process improvement. It is the basis for the above improvement efforts. In the context of the product development process, the proposed metric of interest was delay in activity starts. A tool for collecting and analyzing this delay was developed.

A project simulation that includes the potential for iteration/rework in its determination of project completion date was used to illustrate the expected effects of implementing the three improvement efforts. All comparisons were made relative to a benchmark vehicle development project. The parameters for this benchmark project were chosen conservatively: 10% chance of a given activity

causing rework in a related upstream activity (probability of feedback rework), and activity must repeat 10% of original work if rework occurs (impact of rework). Nominal or most likely activity durations were taken from the company's standard development schedule. These durations were allowed to vary from most likely duration minus 5% to most likely duration + 10%. The key findings were:

- Allowing variation but no rework, the project had only a 12% chance of finishing on schedule
- The benchmark project (which allows for variation in activity duration and potential rework) had only an 8% chance of completing on time
- Increasing the probability of rework (20% probability of feedback rework), due for example to poor communication of required deliverables, reduced the probability of completing on time to 4%
- Increasing the impact of rework (rework impact 20%), due for example to late communication of problems, reduced the probability of completing on time to 6%
- Of the various recovery strategies simulated, a combination strategy (overlap and compression) increased the chance of completing on time to 34% and had a slightly lower expected profit loss than a "pure" overlap strategy
- The maximum difference between project progress determined by assuming no rework and by accounting for potential rework was 9%.

The Role of Project Management in Process Improvement

From the recommended improvement efforts it is clear that project management plays a key role in achieving control over development lead time. Given its role in maintaining project schedules, Program Management has the network and activity status information required to find and track delay causes. More significantly, there are controllable causes of delay associated with the functions of program management: inaccurate assessment of the effects of a change in design requirements, poor communication, poor planning, and weak design freezes. Thus Program Management would be a logical choice to champion these improvement efforts.

This role in improving control of development lead time provides another approach to answering a question that many companies have struggled with: "What is the value of project management?" This has proven to be a difficult question to answer. The largest difficulty is the lack of a direct cause and effect relationship between the degree of project management and the outcome (schedule, budget) of a project. Despite the most conscientious planning and control efforts, there are factors outside the control of Program Management that can affect the completion date and cost of a project. Thus comparing the

outcomes of one project with program management and another without is not necessarily an “apples-to-apples” comparison.

Given Program Management’s role in implementing the improvement tools described in this thesis, the value of program management can be assessed at two levels. At the activity level, the historical reduction in cases of controllable delay and the improved efficacy of schedule recovery actions (compression and overlapping) represent value. The data collected by the delay tracking tool facilitates this assessment.

At the project level, the value of program management can be assessed using the simulation tool. As described in this thesis, the findings from the simulation scenarios outlined above can be considered examples of effective and ineffective program management functions (evaluating changes in design requirements, coordinating communication, planning activities and resources, and coordinating design freezes). The simulation determines the probabilities of completing a project within various time frames. As shown in the thesis, these can be translated into expected profit loss associated with late projects.

Simulating a project with the delay probabilities and schedule recovery efficacy determined from the delay tracking tool data estimates the time required to complete the project. When the process improvement efforts, (driven by Program Management) reduce delay probabilities and recovery efficacy, the simulation can be rerun to determine the new, lower project duration. The simulation provides an “apples-to-apples” comparison since the factors outside the control of Program Management can be kept the same for both simulations. The difference between the earlier and later simulations can be considered value added by Program Management.

5.2 Recommendations

1. Implement delay tracking and analysis in a web-based tool.

The inputs, outputs and processing of this data are described in section 4.2. The tool proposed in this thesis was developed based on some conversation with the stakeholders (program management and engineering groups). However, for effective and successful implementation, stakeholders should be explicitly involved in developing the actual tool. Specifically, some of the parameters that should be agreed upon include:

- What activities to track
- What give-gets/deliverables define those activities

- How to normalize this schedule for larger/smaller projects so that all programs can contribute to and use the database
- What the causes of delay should be used to classify delays
- Who should be responsible for coordinating the input to the tool
- Who should be responsible for coordinating action based on the output of the tool

Collecting data across all future programs, this data can be used to target the reduction of controllable causes of delay. It can also be used to determine representative parameters (rework probabilities and impacts) for the project simulation described in Recommendation 2.

2. Use project simulation to evaluate project management decisions.

The simulation adapted¹ in this thesis determines project completion time for projects where the potential for rework exists. By changing input parameters, it can be used to evaluate the effect on project completion time of:

- Changes in design requirements
- Different schedule recovery strategies (overlap, compression or combination)

The simulation can also be used to determine a more representative basis against which to measure project progress, i.e. project progress (in terms of percent complete) should be measured against an estimate of project duration that accounts for potential rework.

Possible Extension:

3. Utilize project simulation for real-time project management.

Currently the project model simulates the evolution of a project by using activity durations with randomly determined variations and incrementing work done in equal steps as time progresses. In the simulation, delays are *detected* after they have occurred, and then acted on. The software could be adapted to track a current project in real time. The simulation could then be used to:

- *predict* delays based on current project progress, and
- determine optimum recovery strategy based on current project progress.

This idea is discussed more fully in section 5.3.1.

¹ A DSM-based simulation developed by Browning (Browning, 1998) was extended for application in this thesis. See section 4.3.1.

5.3 Further Work

5.3.1 Improving the Simulation

It was proposed that the first step to improving the product development process is to implement a delay analysis tool as described in section 4.2. Once this is in place and has collected the first significant amount of activity delay data, the development of a more representative project simulation is facilitated. And as outlined in 4.3, this project simulation can be used for project control analysis. Beyond increasing the accuracy of the model parameters though, there are several opportunities for extending the model. These are outlined briefly below.

Allow for activity-specific compression efficiencies

Recall that maximum allowable compression refers to the fraction of activity delay that can be reduced by using overtime. Currently the model uses a general maximum allowable compression across all activities. This is clearly a simplification given that different activities can be compressed by different amounts – and some, like testing may not be “compressible” at all. The model could be made more representative by allowing the input of different maximum compressions for each activity. The compression recovery efficacy data from the delay analysis tool can then be used to determine these input parameters.

Allow for amounts of overlap specific to each activity couple

Recall that the amount of allowable overlap is the amount by which the start of a downstream activity can be overlapped with its incomplete upstream activity. Note that unlike compression which is associated with a particular activity, overlap is associated with a pair or couple of activities, i.e. one specifies the overlap *between* two activities. Currently the model uses a general allowable overlap across all activity couples. Like the case with general maximum allowable compressions, this simplification ignores the fact that different activity couples can be overlapped by different amounts. Allowing the input of overlap amounts specific to each of the activity couples would make the model more representative. These parameters could be specified in a $n \times n$ matrix (where n = the number of activities in the project model), i.e. matrix element $_{ij}$ = maximum allowable overlap between upstream activity i and downstream activity j , expressed as a percentage of the downstream activity. The overlap recovery efficacy data from the delay analysis tool can be used to determine these input parameters.

Account for additional costs associated with recovery actions

Currently the model only accounts for incremental manpower costs associated with compression and overlapping. Obviously there are other costs associated with these recovery actions that could be significant and would therefore be valuable to incorporate into the model. For example, the compression of a tooling build activity typically carries with it an increased cost for any design changes due to accelerated reduction in flexibility. And, these cost of these changes increase as the tool build progresses. As another example of additional cost, the use of compression typically means that the activity is carried out with less-thorough checks than normal. This could be interpreted as an increase in the probability of rework. Yet another example of costs not accounted for are any facility or equipment costs incurred when rework due to overlapping is required. As a related note, while the simulation developed by Browning (Browning, 1998b, chapter 6) allows for the specification of costs incurred when rework (not associated with overlapping), they were not included in the simulations run for this thesis.

Make Overlap Recovery Model more Representative

Currently the model assumes that the probability of rework when overlap recovery is used is the same as the “normal” probability of rework, i.e. the probability specified in the DSM probability matrix. However, given a situation where the start of an activity is delayed and recovery action is taken in the form of overlapping, the probability of rework is likely to be higher since the activities are being overlapped more than they were originally intended to be. The accuracy of the model could be improved by using a separate probability matrix to specify the rework probabilities due to the use of overlap recovery. The overlap recovery efficacy data from the delay analysis tool can be used to determine these input parameters.

Yet a further extension would be to determine the probability of rework (for the downstream activity) based on how far along the upstream activity was when the overlap was implemented. The concept of activity “evolution” proposed by Krishnan et al. (Krishnan et al., 1997) could be used to model this. Similarly the amount or impact of the rework could be determined by how far along the downstream activity is when the rework occurs. The “sensitivity” of an activity to change in a related upstream activity could be used to model this (Krishnan et al., 1997). The probabilistic models for three activity execution strategies (sequential, partial overlapping and concurrent) proposed by Yassine et al. may provide an alternative approach to modeling the effects of rework due to overlap in greater detail (Yassine et al., 1995).

Extend delay checking capability

Currently the model only checks for the delayed start of an activity once – at the activity’s scheduled start date. Since recovery actions are only applied when a start delay is recognized, the model is conservative in that it only allows one round of recovery action per activity. In reality however, if a delayed start is found, on-going checks and recovery action would likely continue. Thus another extension of the model could be to incorporate periodic checks for start delay beyond the activity’s scheduled start date. Secondly, proactive delay checking could be incorporated into the model. In this case, at some time before the scheduled start date of an activity, the model should predict whether the start will be on time, based on work remaining in the related upstream tasks and the rate at which the work is being done. If a delay is *predicted* then recovery action can be implemented earlier, improving the chance that the delayed start of the downstream task can be avoided.

Convert the simulation for functional real-time project control

Thus far it has been suggested that the role of the simulation in real-time project control is for *analysis* of a change in design requirements, various recovery strategies, and/or project progress for a hypothetical project. However, with a conceptually simple modification, the software could be used to track the project with real/current activity progress data (start date, work remaining). In other words, rather than simulating the evolution of a project by incrementing work done in equal steps as time progresses, actual activity progress data could be used up to the present date. The software would then be able to alert the project manager of actual delays in activity starts. Then by simulating project evolution beyond the current date, the three aforementioned analyses can be carried out. An even greater benefit could be achieved by having the software alert the project manager of a potential delays in the start of upcoming activities, as determined by simulation. To implement this last functionality, the proactive delay checking described in the previous opportunity for improving the simulation would be required.

5.3.2 A More-Leading Indicator of Project Progress

The criteria for an effective project progress metric were outlined in section 3.2.3. It was suggested that “percent complete”² was an attractive candidate for measuring project progress. In section 2.3.2 it was suggested that the primary shortcoming of this metric was that it did not account for the possibility that rework is required. The use of the project simulation to estimate the potential rework and incorporating

² “percent complete” is a project progress measure based on the weighted sum of the “percent complete” of each of the activities that comprise the project – see description in 2.3.2

this into the determination of “percent complete” was discussed in section 4.3. With this extension, “percent complete” is a simple and useful progress metric.

However, there is opportunity to further develop this metric to make it even more useful. One of the key criteria of an effective metric is that it functions as a leading indicator of the behavior that one seeks to control (Reinertsen, 1997, p. 204). Having a leading indicator or prediction of the behavior allows proactive action to be taken. While “percent complete” is somewhat leading in that it monitors the achievement of intermediate deliverables rather than focussing only on activity starts and finishes, it can be made even more leading by recognizing that the progress of a design activity is driven by the development of design parameters.

Consider a design activity as being the action of evolving the various design parameters of the component or product. In other words, the objective of a design activity is to find values for the design parameters that define the product or component. The action of the design activity is thus to progressively narrow the range of values for a given parameter to a nominal design point. It is changes in these design parameters that drive iteration in the development process. Thus if project progress is based on the status of key design parameters (as opposed to design activities), a more direct and therefore more leading assessment of progress can be made.

The discussion below outlines how project progress based on the status of the key design parameters might be determined. Consider the following example of a concurrent design project with three design parameters (A, B, and C). It is represented in the format of a Design Structure Matrix or DSM.³

	A	B	C	% complete:	Potential rework:
A	*			a	
B	X	*	X	b	b'
C	X		*	c	c'

The development of each parameter can be characterized by:

- Its “percent complete” assuming no changes occur in interdependent parameters, where percent complete can be measured in one of several ways: the current width of a narrowing range of dimensions, time spent as a fraction of estimated total duration, # of open design concerns, % of required studies completed, or % of intermediate milestones completed (e.g. concept, detail drawing, tests passed, final drawing release)

³ See section 4.3 for a description of the Design Structure Matrix (DSM) framework

- Potential for rework caused by interdependent parameters (denoted by X's in the row of the activity under consideration). For example, the potential rework for activity b is b' where:

$$b' = P(\text{change in A}) \times P(\text{change affects B}) \times \text{impact (setback in B)} \\ + P(\text{change in C}) \times P(\text{change affects B}) \times \text{impact}$$

The probability of a change occurring in a parameter's "percent complete", depends on the degree to which the parameters on which it depends has been defined or evolved⁴. Thus as various parameters become more narrowly defined, the potential for rework in the project is reduced. The impact of a change on a parameter, i.e. how much rework is required, is described by its sensitivity⁴ to parameters it is dependent on.

Potential rework creates uncertainty in the completion date of the project. Project progress could thus be defined as the reduction in the sum of potential rework across all parameters (Progress = decrease in b'+c').

As an aside, an alternative determination of project progress that could be considered would be to simply count the total number of potential changes associated with the design parameters that have not been completely developed: Progress = 3 since parameter B can cause 2 changes (there are 2 "x's" in row B of the matrix) and parameter C can cause 1 change.

Having defined project progress, the project simulation can be used to develop a control strategy by simulating the effects of various strategies on project progress. More specifically it can be used to identify the "levers" of the project, i.e. the design parameters on which the progress of the project depend heavily on. A change in one of these design parameters has a significant effect on the project progress metric. These "levers" suggest that the development of design parameters could be prioritized in the order that accelerates project progress.

The interesting point here is that the "levers" at the design parameter level are more likely to change within a project and between projects than those at the activity level. This would suggest that the value of simulating a project to determine these "levers" is higher at the design parameter level than at the activity level. The order in which parameters are developed, and therefore the design "levers" could be expected to vary within a project and between projects for two reasons. Design parameter "levers" are a function of the status of all the design parameters at any given moment in the project. Thus it is likely that they

⁴ See section 1.1 for discussion of the concepts of design "evolution" and "sensitivity" proposed by Krishnan et al.

would change as the project proceeds since the various design parameters will evolve at different rates. Secondly, due to different carryover decisions (re-use of component or systems from existing products), even projects of similar scale with the same design parameters are likely to start with different degrees of definition in these parameters.

In contrast, the order in which *activities* should be executed is not likely to change between projects of similar scale since it would be expected that these projects would have similar activities that have a more constrained order of execution due to dependencies between the activities.

Given the progress metric and the concept of design parameter “levers”, two strategies for project control arise: i) identify and use the “powerful” design parameters to drive project convergence, where a parameter’s “power” is determined by the number of other parameters it could affect and the amount of rework it could cause in those parameters, and ii) identify and use the “vulnerable” design parameters to minimize design churning, where a parameter’s “vulnerability” is determined by the number of other parameters that affect it, and how much rework they could cause for it.

Other questions related to establishing this control policy include:

- What relationship do these “powerful” and “vulnerable” parameters have to the “design modes” identified by Smith and Eppinger (Smith and Eppinger, 1997)?
- Do these “powerful” and “vulnerable” parameters change as the project evolves?
- Are they sensitive to initial conditions, i.e. does the identification of these parameters depend on the order in the project’s design parameters are developed?

References

- Browning, Tyson R. "Sources of Schedule Risk in Complex System Development," *Proceedings of 8th Annual International Symposium of INCOSE*, Vancouver, July 26-30, 1998a.
- Browning, Tyson R. "Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development," PhD Thesis, Massachusetts Institute of Technology, 1998b.
- Carrascosa, Maria, Steven D. Eppinger, and Daniel E. Whitney. "Using the Design Structure Matrix to Estimate Product Development Time," 1998 ASME Design Engineering Technical Conferences, *Proceedings of DETC'98*, Atlanta, Georgia, September 13-16, 1998.
- Center for Quality Management (CQM). The Language Processing (LP) Method: A Tool for Organizing Qualitative Data and Creating Insight. Cambridge, Massachusetts: Center for Quality Management, 1996.
- Clark, K. B., G. Chew, and T. Fujimoto, "Product Development in the World Auto Industry: Strategy, Organization and Performance," *Brookings Papers on Economic Activity*, 3, 1987.
- Eppinger, Steven D., Daniel E. Whitney, Robert P. Smith, and David A. Gebala. "Organizing the Tasks in Complex Design Projects," *ASME Design Theory and Methodology Conference*, Chicago, September 1990.
- Eppinger, Steven D., Daniel E. Whitney, Robert P. Smith, and David A. Gebala. "A Model-Based Method for Analyzing Tasks in Product Development," *Research in Engineering Design*, Vol. 6, 1994, pp. 1-13.
- Goldratt, Eliyahu M. Critical Chain. Great Barrington, Massachusetts: North River Press, 1997.
- Krishnan, V., Steven D. Eppinger, and Daniel E. Whitney. "A Model-Based Framework to Overlap Product Development Activities," *Management Science*, Vol. 43, No. 4, April 1997.
- Project Management Institute Standards Committee (PMI). A Guide to the Project Management Body of Knowledge. Upper Darby, Pennsylvania: Project Management Institute, 1996.
- Reinertsen, Donald G. Managing the design Factory: A Product Developer's Toolkit. New York: The Free Press, 1997.
- Shiba, Shoji, Alan Graham, and David Walden. A New American TOM: Four Practical Revolutions in Management. Portland, Oregon: Productivity Press, 1993.
- Smith, Robert P., and Steven D. Eppinger. "Identifying Controlling Features of Engineering Design Iteration," *Management Science*, Vol. 43, No. 3, March 1997.
- Sobek, Durward K., Allen C. Ward, and Jeffrey K. Liker. "Toyota's Principles of Set-Based Concurrent Engineering," *Sloan Management Review*, Winter 1999, pp. 67-83.
- Steward, Donald V. "The Design Structure System: A Method for Managing the Design of Complex Systems," *IEEE Transactions on Engineering Management*, Vol. 28, No. 3, 1981, pp. 71-74.

Ward, Allen, Jeffrey K. Liker, John J. Cristiano, and Durward K. Sobek II. "The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster," *Sloan Management Review*, Spring 1995, pp. 43-61.

Wheelright, Steven C., and Kim B. Clark. Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality. New York: The Free Press, 1992.

Yassine, Ali, Kenneth Chelst, and Donald Falkenburg. "A Decision Analytic Framework for Evaluating Concurrent Engineering," *IEEE Transactions on Engineering Management*, January 1998.

Appendix A: Source Code for Schedule Recovery Extension

Variable Definitions:

Dim MAXCOMP As Single	>>>>> Variables for Schedule Recovery Subroutine: 'Amount by which activity duration can be reduced using compression/overtime (as % of predecessor delay)
Dim MAXOVERLAP As Single	'Amount by which successor activity can be overlapped with incomplete predecessor activity (as % of successor duration)
Dim STARTDATE() As Single	'Array of scheduled start dates for all activities (time units)
Dim OTPREMIUM As Single	'Overtime premium (wage multiplier)
Dim OVERLAP As Single	'Number of time units remaining in predecessor task when successor task is allowed to proceed
Dim OLREWORK As Single	'Amount of rework for successor task as a result of overlapping with an incomplete predecessor activity (time units)
Dim cCOMPRESS As Single	'Cumulative incremental cost of using compression (overtime) in time units; reinitialized below (at beginning of each run)
Dim cOVERLAP As Single	'Cumulative incremental cost of using overlapping (in time units); reinitialized below (at beginning of each run)
Dim COST_SAMPLES()	'Dynamic array (2,r) containing compression and overlap cost samples for all runs
Dim x As Integer	'Counter
Dim y As Integer	'Counter

Schedule Recovery Subroutine:

Sub Recovery()

Application.StatusBar = "Recovery: Run #" & r
If r = 1 And t = 1 Then

'Display module progress
'on first run only...

ReDim STARTDATE(n)

For x = 1 To n

'Loop through all activities and...

STARTDATE(x) = Worksheets("Recovery").Cells(4 + x, 11) 'Load start dates (in time steps) from Recovery worksheet

Next x

MAXCOMP = Worksheets("Recovery").Cells(2, 2) 'Get maximum compression (percentage of delay that can be eliminated by compression)

MAXOVERLAP = Worksheets("Recovery").Cells(3, 2) 'Get maximum overlap (amount by which successor activity can be overlapped with incomplete predecessor as % of successor duration)

OTPREMIUM = Worksheets("Recovery").Cells(4, 2) 'Get overtime premium (wage multiplier)

End If

For x = 1 To n

'Loop thru all activities to find any that should start now but have not (because predecessors delayed):

```

Application.StatusBar = "Recovery: Run #" & r & "t=" & t & "Act=" & x    'Display module
                                progress
If (t - STARTDATE(x)) < delta_t And (t - STARTDATE(x)) >= 0 And W(x) = 1 Then
                                'If within one time step of the activity's start
                                date...

For y = 1 To (x - 1)                'Loop thru all possible predecessors

    If DSM(x, y, 1) <> 0 And W(y) <> 0 Then    'Check for unfinished predecessor
        If MAXCOMP > 0 Then                'If compression (of unfinished predecessor) is
                                            allowed...
            W(y) = W(y) - W(y) * MAXCOMP    'Reduce amount of work for the delayed
                                            predecessor W(y) by maximum compression
            cCOMPRESS = cCOMPRESS + OTPREMIUM * W(y) * ActS(y) * delta_t
                                            'add cost to cumulative compression cost
        End If

    If W(y) > 0 And MAXOVERLAP > 0 Then    'If predecessor still delayed and overlapping is
                                            allowed...
        OVERLAP = min(W(y) * ActS(y), MAXOVERLAP * ActS(x))
                                            'Amount of overlap (in time steps) cannot
                                            exceed the delay
        W(x) = W(x) - OVERLAP * 1 / ActS(x) 'Reduce duration of successor by
                                            amount of overlap
        If Rnd <= DSM(x, y, 1) Then        'If rework in successor is required
                                            (randomly determined)...
            W(x) = W(x) + OVERLAP * 1 / ActS(x)    'add overlap back
            cOVERLAP = cOVERLAP + OVERLAP * delta_t    'add cost for this rework to
                                                        cumulative overlap cost
        End If
    End If
End If
Next y                'Check next predecessor for delay
End If
Next x                'Check for delay in start delay of next activity

End Sub

```


Appendix B: Calculations

B.1 Expected Profit Loss

One approach to attaching a cost to late project completion is to determine the lost profit due to forgone sales. Lost profit per day may not be a particularly easy quantity to assess, but placing a value on time is important for making informed project management decisions. Lost profit per day will depend on whether a “fixed” or “open window” sales model is used, and on estimated sales rates. In an open window sales model, the assumption is that late entry into the market does not affect total sales, only the timing of them, i.e. the lifecycle of the product is shifted forward in time. In a fixed window sales model, the assumption is that late entry costs market share and total sales and product lifecycle are reduced. Based on the significant degree of competition and the relatively low purchase frequency (automobiles are durable goods), the automobile industry could be represented by the latter sales model.

Sales rates are obviously company and vehicle specific. However, for the purposes of determining comparative profit loss between various scenarios simulated in this thesis, a profit loss of \$1 million/day was used.⁵ Having determined the profit loss/time unit, the output of many runs of the project simulation (probabilities of completing the project within increasing lengths of time) can be used to calculate the expected or mean profit loss for the project:

$$\text{Expected Profit Loss} = \sum_{\text{all late projects}} \begin{matrix} (\text{Probability that project is completed after } y \text{ time units}) \times \\ (\text{\# of time units late}) \times \\ (\text{Profit loss per time unit}) \end{matrix}$$

where: # of time units late = (actual project duration) – (scheduled project duration).

Note that the sum is taken only over late projects, i.e. a profit loss is only incurred if the project takes longer than scheduled.

Rather than treating the results of each run of the simulation separately, the project durations were grouped into a histogram (ranges of project durations). The following spreadsheet illustrates the

⁵ Note that this value also corresponds to an estimate by Clark et al. that the marginal cost of development lead time for a vehicle is at least \$1 million/day (Clark et al., 1987).

calculation of expected profit loss from the histogram of simulated project durations. The data is from the simulation of the benchmark project (defined in section 4.3.3).

Simulation Output:		Calculated Values:	
<i>Range</i>	<i>Frequency</i>	<i>Probability</i>	<i>Expected Profit Loss</i>
[in time units]			[\$ millions]
a	b	c	d
48.0	1	0.1%	n/a
48.5	8	0.8%	n/a
Scheduled Duration: 49.0	74	7.4%	0.0
49.5	174	17.4%	1.7
50.0	264	26.4%	5.3
50.5	212	21.2%	6.4
51.0	166	16.6%	6.6
51.5	70	7.0%	3.5
52.0	27	2.7%	1.6
52.5	3	0.3%	0.2
53.0	1	0.1%	0.1
53.5	0	0.0%	0.0
54.0	0	0.0%	0.0
More	0	0.0%	
Totals:	1000		25.4

$$c = b / \text{total \# of runs}$$

$$d = (a - \text{scheduled duration}) \times c \times \text{profit loss per time unit}$$

B.2 Cost of Recovery

The two schedule recovery actions are compression and overlapping. As discussed in section 4.3.2, the incremental manpower cost of using these actions were modeled in the project simulation as follows:

$$\text{Cost}_{\text{compression}} = \text{Overtime premium} \times (\text{Delay} - \text{Fraction of delay reduced by compression})$$

$$\text{Expected Cost}_{\text{overlap}} = P(\text{rework required for downstream activity}) \times \text{amount of time overlapped}$$

These costs are in terms of the incremental amount of work (in number of time units) required to implement the recovery and/or overlapping. To convert time units to dollar terms, the number of time units was multiplied by a representative wage rate (\$40/hr) and an average number of people involved in the recovery (25 people), i.e.:

$$\text{Total Cost of Recovery} = (\text{Cost}_{\text{compression}} + \text{Cost}_{\text{overlapping}}) \times \frac{\$40}{\text{hr}} \times \frac{8\text{hr}}{\text{day}} \times \frac{20\text{days}}{\text{month}} \times 25 \text{ people}$$

THESIS PROCESSING SLIP

FIXED FIELD: ill. _____ name _____

index _____ biblio _____

► COPIES: Archives Aero Dewey Eng Hum
Lindgren Music Rotch Science

TITLE VARIES: ► _____

NAME VARIES: ► William

IMPRINT: (COPYRIGHT) _____

► COLLATION: 119p

► ADD: DEGREE: S. M. ► DEPT.: M. E.

SUPERVISORS: _____

NOTES:

cat'r:

date:

page:

► DEPT: Mgt J139211

► YEAR: 1999 ► DEGREE: S. M.

► NAME: MAR, Carey W.