An Integrated Method for Managing Complex Engineering Projects Using the Design Structure Matrix and Advanced Simulation

by

Soo-Haeng Cho

Bachelor of Science in Engineering
Division of Aerospace and Mechanical Engineering
Seoul National University, 1999

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

at the
MASSACHUSETTS INSTITUTUE OF TECHNOLOGY

June 2001

Signature of Author . . . . . .

Soo-Haeng Cho
Department of Mechanical Engineering
May 11, 2001

Certified by . . . . . .

Steven D. Eppinger
General Motors LFM Professor of Management
Thesis Supervisor

Accepted by . . . . . .

Ain. A. Sonin
Chairman, Department Committee on Graduate Students

An Integrated Method for Managing Complex Engineering Projects Using the Design Structure Matrix and Advanced Simulation

by

Soo-Haeng Cho

Submitted to the Department of Mechanical Engineering
on May 11, 2001 in partial fulfillment of the
Requirements for the Degree of

Master of Science

ABSTRACT

This thesis proposes an integrated project management framework for complex engineering projects such as the development of an automobile. The integrated method streamlines project planning and control using three modules: structuring, modeling, and scheduling. In the structuring module, the design structure matrix (DSM) method is used to structure the information flows among tasks and capture the iteration loops. By classifying various types of information dependencies, a critical dependency path is identified and redundant constraints are removed for the modeling and scheduling analyses. In the modeling module, a generalized process model predicts complex behaviors of iterative processes using advanced simulation techniques such as the Latin Hypercube Sampling and parallel discrete event simulation. The model computes the probability distribution of lead time and identifies critical paths in a resource-constrained, iterative project. Using the results of analyses from the structuring and modeling modules, a network-based schedule in the form of a PERT or Gantt chart is developed in the scheduling module. The schedule is used as the basis for monitoring and control of the project.

The primary goal of this work is to develop an integrated method that guides project management efforts by improving the effectiveness and predictability of complex processes. The method can also be used for identifying leverage points for process improvements and evaluating alternative planning and execution strategies. Better project management will ultimately result in a better quality product with timely delivery to customers. Two case studies are performed to illustrate the utility of the method.

Thesis Advisor: Steven D. Eppinger, General Motors LFM Professor of Management

Acknowledgements

First of all, I would like to thank my advisor, Professor Steven Eppinger for the opportunities he has given me and for his invaluable guidance throughout the years. I have learned much from him about the attitudes and skills for conducting research, collaborating with other colleagues, presenting ideas, and being open-minded. I will keep these lessons in my mind wherever I go.

I am grateful to the Center for Innovation in Product Development at MIT for providing great research settings and funds for this research. I am also thankful to the Singapore-MIT Alliance for supporting the research in the second year.

I am indebted to several people for helping me complete this work. I am thankful to Dr. Tyson Browning for providing me a basic foundation for this work. I'd like to thank Dr. Ali Yassine for his encouragement and friendship. I am also grateful to the people who broadened my vision to see a "real world": Norm Lewicki, Bill Kennedy, Keith Schmidt, Ron Williams, and Bernard Spycher.

I would like to thank my friends, Taesik Lee, his wife Haeyun Oh, and Yong-Suk Kim who have made my life at Boston exciting and memorable. I am sincerely grateful to Seonghwan Cho, Jongyoon Kim, Sungbae Park, and all other friends at MIT, for sharing joys and hands with me.

Lastly, I would like to thank my parents for their trust and support.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Today's competitive market has created a highly challenging environment for product development. Companies are under increasing pressure to sustain their competitive advantages by reducing product development time and cost while maintaining a high level of quality. These needs drive companies to focus on developing a well-coordinated development plan to organize their processes and resources (Takeuchi and Nonaka (1986), Clark and Fujimoto (1991), Wheelwright and Clark (1992), Ulrich and Eppinger (2000)).

A complex product development project involves a large number of tasks executed by a network of professionals from various disciplines. As complexity increases, it becomes more difficult to manage the interactions among tasks and people; it may be impossible to even predict the impact of a single design decision throughout the development process (Eppinger et al. 1994).

Since the introduction of network scheduling techniques such as PERT (Malcolm et al. 1959) and CPM (Kelley and Walker 1959), many researchers have made extensions to make these standard techniques more powerful. These improvements include the following:

- Allocation of resources across tasks
- Modeling of uncertainty in estimated task durations using Monte Carlo methods
- Prediction of cost as well as project duration
- Analysis of iterative processes including concurrent and/or stochastic rework of tasks

- Developing measures for criticalities in both task and project levels
- Structuring information flows using a diagram or matrix
- Inserting buffers to manage variance in durations

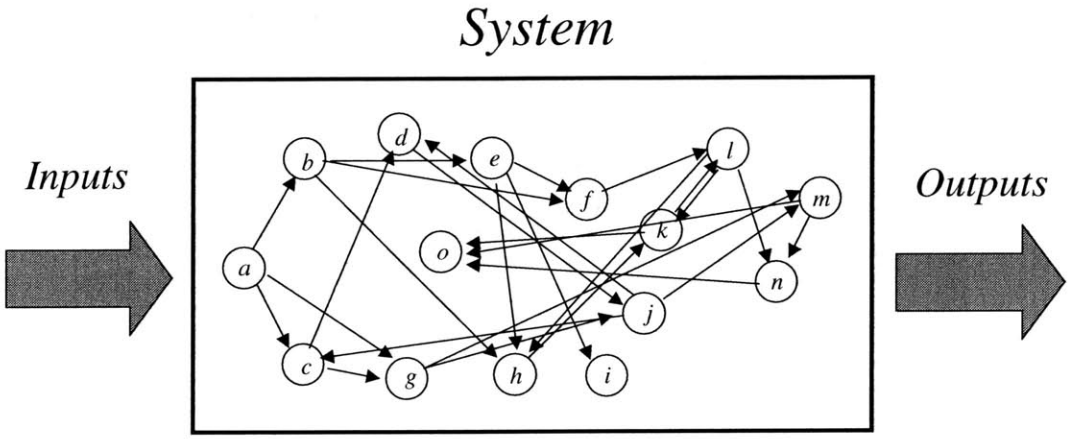
As listed above, there have been tremendous efforts in academia and in the field to improve the project management methods and tools that are currently available. However, there has not been any integrated framework that utilizes all of these improvements.

The goal of this work is to develop an integrated method that utilizes positive aspects of individual methods while resolving compatibility problems among them. This thesis proposes a method that streamlines project planning using the design structure matrix (DSM) method and advanced simulation techniques such as the Latin Hypercube Sampling (LHS) and parallel discrete event simulation. The first step of planning processes is to structure information flows among tasks using the DSM. The second step is to model development processes using advanced simulation based on the characteristics of a project, tasks, and interfaces between tasks. The third step is to construct a network-based schedule in the form of a PERT or Gantt chart based on the analyses of structuring and modeling. This streamlined planning approach can be extended to an integrated project management framework for planning, monitoring and control. This framework can be used to improve the effectiveness and predictability of complex processes, accelerate communications among people, and guide project management efforts throughout the development process.

The proposed method is based on the theory of systems modeling and simulation, and adapts it to the project management context. It views a large scale development project as a complex system within which its component tasks interact through information exchanges in order to produce a high-quality product within targeted schedule and budget. Figure 1-1 illustrates the systems view of a project adopted in the thesis.

In order to test the utility of the proposed method, two industrial cases are used. The first case models the uninhabited aerial vehicle (UAV) preliminary design process at an

aerospace company. The data are from Browning (1999) and extended to include additional parameters that can be incorporated in the proposed model. The second case demonstrates the streamlined project planning in a logistics project at a medical equipment company. The data were collected through the interview of a project manager. The results and validity of the method are discussed later in the thesis.



System objective: a high-quality product within targeted time and cost

○ : *a component task* → : *information flow*

FIGURE 1-1. SYSTEMS VIEW OF A PROJECT

1.2 Literature Review

The CPM/PERT approaches have been widely acknowledged as industry standard techniques for project management for the past fifty years. However, these basic techniques are not suitable for managing complex engineering projects due to their limited analytical capabilities. The integrated method that the thesis proposes built upon two areas of research that complement the weaknesses of those basic techniques. The first area is related to *process-structuring* methods that structure complex information flows and represent complex interactions among tasks. The second area is related to *process-*

modeling methods that characterize behavioral aspects of processes along the time line such as stochastic models using a Markov chain etc. Throughout the thesis, the terms of process-structuring and process-modeling methods denote those methods described above.

1.2.1 Process-Structuring Literature

In the network-based models such as the PERT/CPM, a project network is usually built by identifying precedence relationships among the tasks that are close to one another in sequence from experience. However, this approach is not easily scalable for complex engineering projects where a large number of tasks interact with one another in a complex way. In addition, they do not represent iterative relationships very well.

Steward (1981) developed the design structure matrix (DSM) to model the information flows of design tasks and to identify their iterative loops. The DSM method is based on earlier work in large-scale system decomposition (Sargent and Westerberg (1964), Steward (1965), Ledet and Himmelblau (1970), Warfield (1973)). Eppinger et al. (1994) extended Steward's work by explicitly modeling information coupling among tasks and investigating different strategies for managing entire development procedures. Rogers (1999) developed a computer package as a design-supporting tool for a design manager. He used the Generic Algorithm for partitioning and presented hierarchical decomposition method similar to Warfield (1973).

The extended DSM methods have been widely applied to various manufacturing industries as a project-planning tool. It complements the traditional project management tools by showing a structural view of a complex project in a compact, square matrix. However, there are also limitations in this method as a *stand-alone* project management tool such as:

- Limited capability showing a time scale (time aspects):

A square matrix form in the DSM is very useful for the compact visualization of a project structure. However, it does not show time aspects of a project graphically as in the Gantt chart for monitor and control purposes.

- Limited compatibility with the network-based scheduling methods and tools (compatibility issues):

  Using the DSM method, we can identify and represent iterative loops (cycles) more effectively than other methods. In order to increase the compatibility of the DSM method with other schedule methods, further research is necessary in the following two areas: (1) how iteration loops in the DSM can be analyzed and illustrated in the Gantt chart, associated with time aspects of iterations, and (2) how information dependencies can be transferred to precedence task relationships in a PERT or Gantt chart.

Austin et al. (1999) proposed using a 'rolled-up' task for construction projects to represent a loop in the Gantt chart within which constituent tasks are shown in parallel without precedence relationships. However, the analytical features such as predicting the durations of those rolled-up tasks, computing criticalities etc. are not included in their method.

1.2.2 Process-Modeling Literature

Two streams of research in this area are related to this work: one is resource allocation and the other is iteration.

Many models related to resource allocation seek to find the optimal allocation of resources to tasks, minimizing lead time subject to resource constraints. Early efforts concentrated on two areas: one is the formulation and solution of the problem as a mathematical (usually integer) programming problem and the other is the development of heuristic methods to obtain approximate solutions (Patterson 1984). Given the NP-hard nature of the underlying problem, enumerative approaches (Pritsker et al. 1969, David and Heidorn 1971, Patterson and Roth 1976, Talbot and Patterson 1978, Patterson et al. 1990) and branch and bound solution approaches (Stinson et al. 1978, Christofides et al.

1987) were developed for solving this problem optimally. However, the exact approaches are not computationally viable because the magnitude of combinatorial problem grows too big, as more tasks are involved in the network. Cooper (1976) proposed heuristic priority rules based on experimental results for assigning resources when tasks are competing for limited resources. Woodworth and Shanahan (1988) proposed the method identifying the critical sequence (Weist 1964) but did not provide a rigorous set of heuristic rules. Bowers (1995) presented a set of heuristics for calculating a resource-constrained slack and discussed the measures of tasks' importance. More recently, Joglekar et al. (2000) have formulated the resource allocation problem with a deadline constraint, seeking to maximize design quality.

Iteration is a fundamental characteristic of development processes (Alexander (1964), Eppinger et al. (1994), Browning and Eppinger (2000)). Several analytical models have been developed to explain iterative design processes. These include sequential iteration models, parallel iteration models, and overlapping models.

In the sequential iteration model, tasks are repeated one after the other by a probabilistic rule. GERT network is a generalized PERT network that allows probabilistic routing and feedback loops, and Q-GERT is its simulation package for large-scale projects (Neumann (1979), Pritsker (1979), Taylor and Moore (1990)). Smith and Eppinger (1997a) developed a model based on a reward Markov chain using the DSM representation for repetition probabilities and task durations. Ahmadi and Wang (1994) extended the sequential iteration model by taking into account dynamic iteration probabilities and learning effects. Eppinger et al. (1997) used Signal Flow Graphs to compute the probability distribution of lead time. Andersson et al. (1998) extended the SFG model to include learning effects and other non-linearities.

In the parallel iteration model, multiple interdependent tasks work concurrently with frequent information exchanges. AitSahlia et al. (1995) compared sequential and parallel iteration strategies in terms of time-cost tradeoff. Hoedemaker et al. (1995) discussed the limit to parallel iteration due to increasing communication needs. Smith and Eppinger

(1997b) developed a model for fully parallel iteration processes which predict slow and rapid convergence of parallel iteration. Carrascosa et al. (1998) included the probability and impact of changes in stochastic processes.

In the overlapping model, two development tasks are overlapped to reduce total lead time. Ha and Porteus (1995) built a model of concurrent product and process design processes and explored the optimal review timing minimizing total expected completion time. Krishnan et al. (1997) developed a framework for overlapped sequential tasks. They explained appropriate overlapping strategies based on upstream information evolution and downstream iteration sensitivity. Loch and Terwiesch (1998) presented an analytical model for optimal communication policy between two sequentially overlapped tasks.

While each of the resource and iteration models presented in the literature captures some aspects that earlier methods do not explain very well, each new model also has its own limitations. Resource models have considered optimal resource allocation and heuristic resource priority rules. However, there has not been any significant work resolving resource over-allocation issues in the *overlapped* and *coupled* project networks where tasks repeat by a probabilistic rule. In addition, no algorithm is computationally practical when the model takes into account the uncertainty inherent to estimated durations of tasks. The sequential iteration models present the closed-form solutions for the development time of an iterative network with a limited number of tasks. However, due to the limitation of analytical approaches, they do not handle resource constraints as well as more general project networks having parallel tasks (or paths) and overlapping. The parallel iteration models analyze important aspects of concurrent engineering but use highly simplified assumptions. The two-task overlapping models provide the optimal way to reduce the time of two sequential tasks having the interface of unidirectional information transfer. However, the concept does not easily apply to multiple tasks, in particular, having multiple paths with iteration.

Adler et al. (1995) developed a simulation-based framework using queuing principles for a multi-project, shared-resources setting. The model incorporates simple iterative effects

on development time but neglects many characteristics of iteration including overlapping (or preemptive) iteration. Unlike most other resource models, it uses the average amount of time that each resource group spends on each process of multiple projects and assumes that work can be reallocated between resources with perfect efficiency, which is very difficult to achieve. Thus, resources are not treated as constraints, so that the model is not able to calculate criticalities of tasks and identify critical sequences.

Browning and Eppinger (2000) used simulation to analyze iterative processes based on a DSM model as well as to account for normal variance of development task durations. However, this first DSM simulation method is based on rather restrictive assumptions regarding task concurrency and rework. Roemer et al. (1999) discussed time-cost tradeoffs in multiple overlapped tasks. However, his model is limited to a single path, assuming that sequential iterations take place among sub-tasks within a task.

1.3 Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2 presents the process-structuring method using the design structure matrix. This chapter also discusses the procedure to construct a matrix model for a complex engineering project.

Chapter 3 presents the process-modeling method using advanced simulation such as the Latin Hypercube Sampling (LHS) and parallel discrete event simulation. Based on the underlying structure of the project, complex behaviors of tasks are modeled and analyzed for process improvements.

Chapter 4 presents the integrated project management framework based on the methods presented in Chapters 3 and 4. This framework can be used for streamlined project planning, monitoring, and control.

Chapter 5 presents the case studies performed using the methods explained in the previous chapters. The results and findings from the case studies are also discussed.

Chapter 6 concludes the research work in this thesis and discusses the limitations and extensions of the proposed method for further work.

CHAPTER 2

Process-Structuring Methods Using the Design Structure Matrix

2.1 Introduction

This chapter discusses the procedure to construct a matrix model for mapping information flows among tasks. It also presents the method for structuring complex engineering processes using the design structure matrix (DSM). It. The method views a large-scale development project as a complex system as described in the previous chapter. Using the partitioning procedure based on Boolean matrix operations, hierarchical decompositions in the DSM are performed based on the as-late-as-possible rule as well as the as-early-as-possible rule in the schedule. Process hierarchies are used to analyze the structure of a project. The method that classifies various types of information dependencies is presented. It characterizes dependencies and identifies a critical dependency sequence from a structural view.

2.2 Constructing a Matrix Model for a Complex Project

From the systems view, it is necessary to identify and define the component tasks and informations exchanged among them in order to understand the structure of a complex project. This section explains the procedure to construct a matrix model for a complex engineering project in the following three steps: (1) decompose a project into manageable tasks, (2) identify sources of inputs and information flow types for each task, and (3) map information flows among tasks into a square matrix.

2.2.1 Decompose a Project into Manageable Tasks

Simon (1973) made an argument about the necessity of decomposing or dividing a project into smaller tasks as follows:

> "From the information-processing point of view, division of labor means factoring the total system of decisions that need to be made into relatively independent subsystems, each one of which can be designed with only minimal concern for its interactions with the others. The division is necessary because the processors that are available to organizations, whether humans or computers, are very limited in their processing capacity in comparison with the magnitude of decision problems that organizations face. The number of alternatives that can be considered, the intricacy of the chains of consequences that can be traced – all these are severely restricted by the limited capacities of the available processors."

Following his perspective, the successful accomplishment of a complex project requires decomposing a project into smaller tasks that are (Kerzner 1995):

- manageable so that specific authorities and responsibilities can be assigned
- independent or with minimum dependence on other tasks
- integratable so that the total project can be seen when combined
- measurable in terms of progress

Figure 2-1 illustrates one simplified method of decomposition processes. First, set the milestones and identify deliverables in each milestone. Second, create a list of informations needed to produce each milestone deliverable. Third, draw causal diagrams among the informations, if possible. Fourth, define a task that describes the process to generate one or a few informations in the lowest hierarchy of the Work Breakdown Structure (WBS). Fifth, group the tasks to have simple hierarchies in the WBS. Note that the hierarchies in the WBS might draw unnecessary boundaries between tasks, decreasing flexibility in processes. DSM partitioning explained later in this chapter may help cluster the groups of tasks that are tightly coupled.

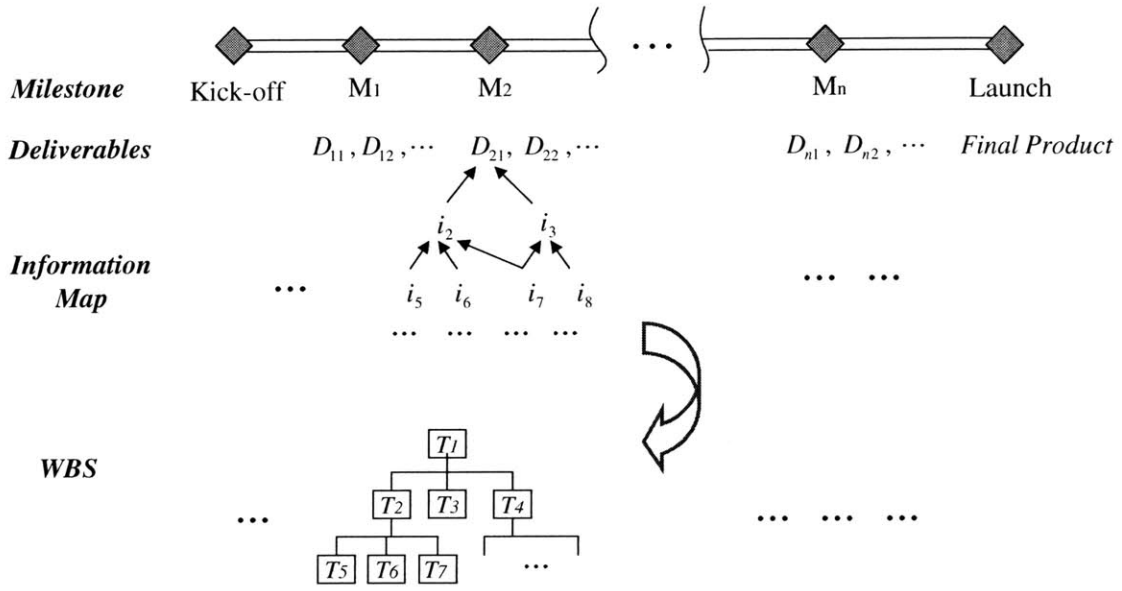


FIGURE 2-1. AN EXAMPLE OF PROJECT DECOMPOSITION PROCESSES

As Browning (1999) noted, the level of detail (or abstraction) at which tasks represent processes is an important aspect when constructing a process model. While building a model with sufficient details can improve the understanding of processes, there may be the adverse effects using the project plan based on ill-defined processes, which include:

- increasing 'paper' complexity of a project with a larger number of tasks, which is opposite to the objective of structuring processes
- prohibiting buy-ins from the project members who have not participated in describing/defining their 'as-is' processes. (There might be more chances of this phenomenon when a project plan is built upon a standardized template.)
- decreasing reusability of the processes
- decreasing flexibility of individual performers

Therefore, it is important to decompose a project into a set of tasks to the appropriate level of detail.

When a project is very novel, it is very difficult to specify tasks as well as interim deliverables. In this case, we can still improve project performance by structuring and organizing *emerging* tasks as the project unfolds (von Hippel 1990).

2.2.2 Identify Sources of Inputs & Information Flow Types for Each Task

Throughout the thesis, the author follows the information-based view in which a task is the information-processing unit that receives information from other tasks and transforms it into new information to be passed on to subsequent tasks. The information exchanged between tasks includes both tangible and intangible types such as parts, part dimensions, bill of materials etc.

From a schedule perspective, there exist two types of *information flow in a task*: (1) information flow at the beginning or at the end of the task, and (2) information flow in the middle of the task, as illustrated in Figure 2-2.

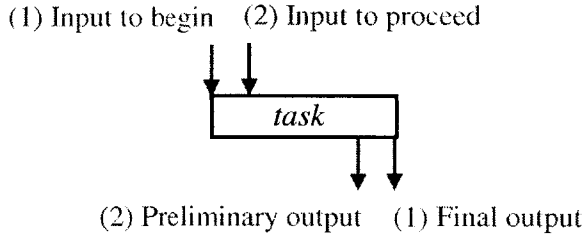
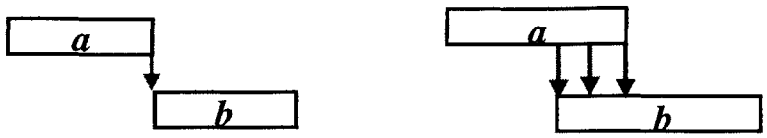


FIGURE 2-2. TWO TYPES OF INFORMATION FLOW IN A TASK

Based on the above observation, the author defines two types of *information flow between two tasks*. The first type represents the case that a downstream task requires final output information from an upstream task to begin its work. The second type represents the case that a downstream task uses final output information in the middle of its process and/or begins with preliminary information but also receives a final update from an upstream task. The second type of information flow is very common unless a project

planner were to decompose all tasks to the level of detail such that all information exchanges take place at the end of tasks. Figure 2-3 illustrates the types of information flow from upstream task *a* to downsream task *b*.



(a) First Type *(b) Second Type*

FIGURE 2-3. TYPES OF INFORMATION FLOW BETWEEN TWO TASKS

The second step of constructing a matrix model is to identify sources of inputs and information flow types for each task following the above rules. The rationale behind inquiring inputs instead of outputs is that people performing a task are better aware of the characteristics of inputs than those of outputs. The tasks that require outputs of each task can be easily identified after mapping all information flows for task inputs. Figure 2-4 illustrates the simple example in which tasks *e*, *f* and *i* produce outputs i_e, i_f and i_i, respectively. Task *i* requires inputs from tasks *e* and *f*. i_e is transferred to task *i* by the second type of information flow while i_f is transferred by the first type (as indicated by the numbers on the arrows).

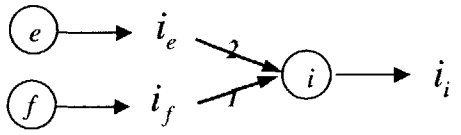


FIGURE 2-4. SOURCES OF INPUTS AND INFORMATION FLOW TYPES

2.2.3 *Map Information Flows among Tasks into a Square Matrix*

A simple methodology to map information flows among tasks is using a square matrix. First, list the tasks down the left side in their rough sequence of execution with unique indices. The same indices are listed in the row above the square matrix. When there is the first type of information flow from task *a* to task *b*, then enter "1" into (i, j) of the square matrix where i and j are the unique indices representing tasks *b* and *a*, respectively. For the second type of information flow, enter "2". Reading across a row reveals the tasks where the inputs of the task corresponding to the row come from. Reading down a specific column reveals the tasks receiving outputs from the task corresponding to the column. If the sequence of tasks in the matrix is the sequence of execution, a nonzero element in the upper diagonal represents a feedback. Figure 2-5 illustrates a simple example of this matrix.

Task Name		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	1															
b	2	1														
c	3	2									1					
d	4			2							1					
e	5		1													
f	6		1			2										
g	7	1		1												
h	8		2			1							1			
i	9					2	1									
j	10				1			1								
k	11								2				2			
l	12											1				
m	13							1			1					
n	14									1			1	1		
o	15											1		1	1	

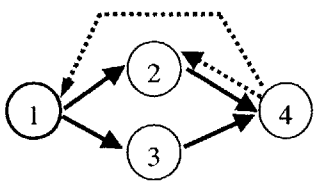
FIGURE 2-5. A SQUARE MATRIX MAPPING INFORMATION FLOWS AMONG TASKS

2.3 Hierarchical Decomposition of Processes

Partitioning is the process of reordering tasks such that information flow marks in the matrix are placed in the lower diagonal or grouped together within square blocks on the diagonal (Steward 1981). It identifies iteration loops (cycles) and moves them to the diagonal as close as possible. Steward (1981) called a square matrix obtained after partitioning as the design structure matrix (DSM). This section reviews basic aspects related to partitioning and hierarchical decomposition of processes in systems modeling. The thesis basically adopts the partitioning method that has been developed using the Boolean matrix manipulation based on the graph theory. Improvements are made such that systems modeling technique can be better utilized in project management.

2.3.1 Graph Theory and Boolean Matrix Manipulation

A *graph* is a collection of *vertices* (*nodes*) and *arcs* (*edges*) connecting the vertices. A *directed graph* or *digraph* is a graph in which the arcs have directions. A vertex represents a system element while an arc represents information flow between the elements. The *adjacency matrix R* is a Boolean matrix representing a graph uniquely. *R* is a square matrix with the same number of vertices where $r(i, j) = 1$ if there is an arc from vertex j to vertex i, and $r(i, j) = 0$ otherwise. Figure 2-6 illustrates a diagraph and its adjacency matrix.



(a) Digraph

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

(b) Adjacency Matrix

FIGURE 2-6. A DIGRAPH AND ITS CORRESPONDING ADJACENCY MATRIX

The *reachability matrix* R^* is the element-by-element Boolean union of all of the powers of the adjacency matrix up to the *n*th, where *n* is the number of rows in the *R*. It is also called the transitive closure of the adjacency matrix. An important property of the adjacency matrix is that *i*th power of this matrix gives all the *i* step paths between vertices including loops. Then, the reachability matrix represents all of the paths of any length between vertices (Ledet and Himmelblau 1970). Steward (1962) has shown that *n*th power of $(R \cup I)$ matrix is equivalent to the reachability matrix, *i.e.* $(R \cup I)^n = R^n \cup R^{n-1} \cup \cdots \cup R \cup I = R^*$. This property saves computation time significantly when computing the reachability matrix by using the second, fourth, eighth powers, and so on. Figure 2-7 shows the reachability matrix of the digraph in Figure 2-6.

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

FIGURE 2-7. A REACHABILITY MATRIX

2.3.2 Hierarchical Decomposition Based on As-early-As-Possible rule

The thesis basically adopts the partitioning method developed by Warfield (1973) and improves it to be better applicable to project management. This method uses Boolean matrix manipulation and decomposes a project into hierarchical levels that tasks belong to. Process hierarchies are also used in information dependency classification in the next section. There is an alternative partitioning method called "path searching" which does not use Boolean matrix manipulations. Steward (1965) proposed this method based on the earlier work of Sargent and Westerberg (1964). The method identifies a loop (circuit) by tracing information flows until a node is encountered twice. However, this method does not provide information about process hierarchies.

Warfield used the reachability matrix to identify system elements belonging to each hierarchical level. A *level* was defined in the way that any two elements at the same level are either not connected to each other or else there are two-way connections between the two elements. The latter is the case that two elements are part of a *loop* (*cycle*). Once the hierarchical levels and their elements are determined, the original Boolean matrix is rearranged to show the separate blocks of the partition formed by the sets at the various levels.

In the context of project management, a *system* in systems modeling is a *project* and an *element* within a system is a *task* of a project. An information flow mark – hereafter referred to as a *dependency mark* – in the lower diagonal of the matrix represents *feedforward* information flow from an upstream task to a downstream task. A dependency mark in the upper diagonal of the matrix represents a *feedback* information flow to an upstream task from a downstream task. The term of a *coupled block* (Eppinger et al. 1994) is defined as a set of tasks consisting of a *loop* (*cycle*), within which there is at least one dependency mark in the upper diagonal of the matrix. Due to its cyclic relationships among constituent tasks, iterations may take place potentially among the tasks within a coupled block. Tasks in a coupled block belong to the same process *level* while tasks outside coupled blocks are located in the same level if they are independent of each other. It is also possible that multiple coupled blocks and/or single tasks are located in the same level when there is no information flow between them. Thus, tasks and/or coupled blocks in the same level can work in parallel.

In order to apply the two-type scheme of information dependencies, the second type of information flow in the matrix are replaced by "1" before partitioning and recovered after partitioning which is based on Boolean matrix manipulations. After this partitioning process, coupled blocks are identified and tasks are rearranged based on the *as-early-as-possible (AEAP) rule* which assumes that a task starts immediately after all the inputs necessary to begin are available. In the same level, a coupled block is located before a single task and, with more than one coupled block, the one with more constituent tasks is located upfront. However, tasks within a coupled block follow the sequence in the matrix

input due to its cyclic relationship. Figure 2-8 illustrates the algorithm that computes the DSM based on the AEAP rule.

(1) For every task i (i = 1, 2, ..., n) within a project, compute the following sets:

$$R_i = \{ j \mid R^*(i, j) = 1 \; ; \; j = 1, \ldots, n \}$$
$$A_i = \{ j \mid R^*(j, i) = 1 \; ; \; j = 1, \ldots, n \}$$

where, R^* : reachability matrix

n : number of tasks in the project

(2) Compute the set L_k for k = 1, ..., m , where m is the maximum level in the project, as follows:

(i) $L_1 = \{ i \mid R_i \cap A_i = R_i \; ; \; i = 1, \ldots, n \}$

(ii) for k = 2, ..., m,

$$L_k = \{ i \mid R_i' \cap A_i' = R_i' \;\; s.t. \;\; R_i' = R_i - (L_1 \cup \cdots \cup L_{k-1}) \;\; and \;\; A_i' = A_i - (L_1 \cup \cdots \cup L_{k-1}) \; ; \; i = 1, \ldots, n \; and \; i \notin (L_1 \cup \cdots \cup L_{k-1}) \}$$

where $m = \min k \;\; s.t. \;\; (L_1 \cup \cdots \cup L_k) = \{1, \ldots, n\}$

(3) Construct the DSM from the task i, of which n(R_i') (n(R_i) for L_1) is higher, in L_k for k = 1, ..., m.

<Note>

(i) L_k is the set having the tasks in level k.

(ii) If n(R_i') (n(R_i) for L_1) is higher than one, the tasks in R_i' (R_i for L_1) constitute a coupled block.

(iii) '(3)' determines the ordering of tasks in the same level such that a coupled block is located before a single task and, with more than one coupled block, the one with more constituent tasks is located upfront.

FIGURE 2-8. ALGORITHM COMPUTING DSM BASED ON AEAP RULE

When the size of a coupled block after partitioning is too big, the partitioning result may not provide useful information about the structure of a project. *Tearing* (Steward 1965 and 1981) is the technique that makes a coupled block smaller and finds a suitable ordering of tasks within a coupled block. It removes a dependency mark in the upper diagonal where task performers can make good estimates for its outputs, and reorders tasks by partitioning again. The torn mark is recovered after reordering. Several researchers (Steward (1965), Weinblatt (1972), Rogers (1989), Kusiak and Wang (1993)) proposed the methods for efficient tearing that minimizes number of tears or confines tears to the smallest block along the diagonal (Yassine et al. 1999). However, an optimal sequence of tasks within a coupled block depends the characteristics of individual tasks as well as the characteristics of interfaces among tasks. Thus, while the tearing technique (which is based solely on the characteristics of interfaces among tasks) suggests an efficient order of execution from a structural view, it may not provide an optimal sequence. This issue is further discussed in Chapter 4.

It is generally accepted that iteration improves the quality of a product in a design project while increasing development time. Compromise must be made to optimize the tradeoffs between these conflicting project goals (Eppinger et al. 1994). Thus, when constructing a matrix model, a project planner should be careful not to mark all feedback flows unless they represent iteration significant enough to affect development processes. After partitioning, for this purpose, Eppinger et al. proposed artificial decoupling which actually removes one or more 'less important' task dependencies from the matrix. This can also be achieved by distinguishing *unplanned* iterations from *planned* iterations indicated by marks in the upper diagonal of the DSM.

Planned iteration represents possible iteration between overlapped tasks or tasks within coupled blocks, of which the impacts are accounted in a project plan. For *overlapping iteration* between the tasks linked by "2" dependency mark in the DSM, the planned iteration takes place when the downstream task receives new information from the overlapped upstream task after starting to work with preliminary inputs. For *sequential iteration* among the tasks within a coupled block, the planned iteration takes place when

the upstream task needs to incorporate the change of the initial estimate that it has started with for the output from the downstream task. Also, the iteration may be caused by the failure to meet the established criteria in testing or reviewing.

Unplanned iteration represents a possible iteration in a system level or between major development phases (usually a long feedback loop in the DSM), which is not accounted in a project plan. The loop usually feeds back from the task such as major testing or reviewing at the end of each phase. This can also be caused by unexpected market changes, technology innovation etc. in the middle of processes. The unplanned iteration is often regarded as a failure mode and requires managerial decision about whether to continue or abandon the project. When it is decided to pursue another iteration, it is common to re-plan the project. The proposed method represents this type of iteration flow with 'warning sign' (◈) in the DSM after partitioning. If this loop is initially marked in the DSM and determined as unplanned after partitioning, smaller coupled blocks can be obtained by replacing the dependency mark with the warning sign. Even though unplanned iteration is not accounted in the schedule, it is very important to be aware of the existence of those iterations so that appropriate contingency plans can be made as part of project planning.

Figure 2-9 illustrates the results of hierarchical decomposition of the matrix input in Figure 2-5, following the AEAP rule (the DSM obtained from this procedure is referred to as the *AEAP DSM* hereafter). Note that a new index is assigned to each task. Two coupled blocks are identified and different color schemes are applied to the blocks. The tasks within each block are expected to do *planned* iterations. The example also shows that it has chances to do *unplanned* iterations after finishing task *n* which might be the task performing major reviews. In level two, *block 1* and task *b* are independent of each other, which implies that they can be worked in parallel. The same interpretation can be made for the tasks in level three and four.

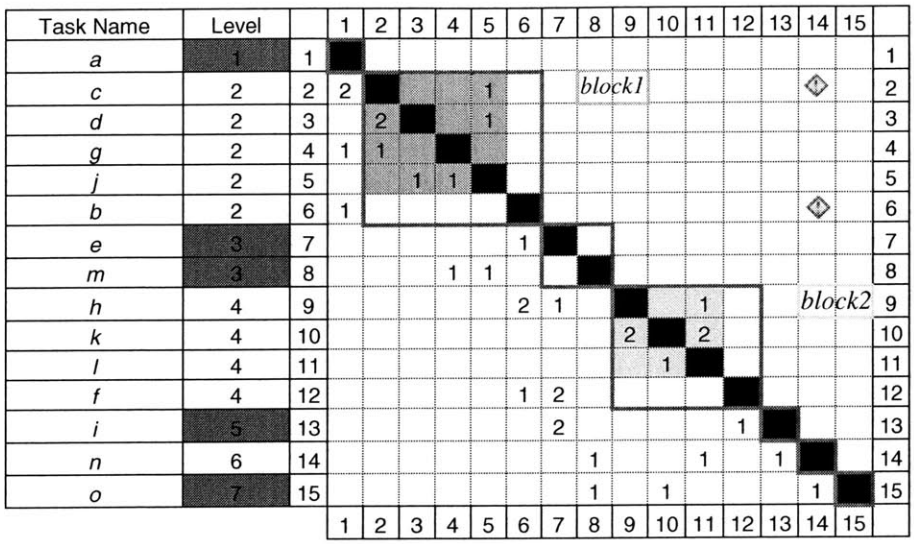


FIGURE 2-9. HIERARCHICAL DECOMPOSITION BASED ON THE AEAP RULE

Another view of the DSM can be obtained by *collapsing* coupled blocks into block tasks as suggested by Eppinger et al (1994). This type of the DSM can provide the structure of a project more concisely and can also be used in multi-tier DSM's (Sabbaghian et al. 1998). In the collapsed view, information flow from a single task to a block task is marked as "2" when there exists at least one second-type flow between the single task and any constituent task within the block in the DSM. In contrast, any information flow from a block task to any single or block task is regarded as the second-type flow as it is very likely that the preliminary outputs of tasks within the block are transferred to downstream tasks before converging to their final forms at the end of iterations. Figure 2-10 illustrates the collapsed view of the DSM shown in Figure 2-9 (hereafter referred to as *AEAP Collapsed DSM*). Note that a new index is assigned to each *single* or *block* task. Two coupled blocks are collapsed into block tasks of which the diagonal are filled with the same color used for coupled blocks in the *AEAP DSM*. Notice that information flows to and from block tasks are changed according to the above rules.

Task Name	Level		1	2	3	4	5	6	7	8	9	10	
a	1	1											1
block1	2	2	2										2
b	2	3	1										3
e	3	4			1								4
m	3	5		2									5
block2	4	6			2	1							6
f	4	7			1	2							7
i	5	8				2			1				8
n	6	9					1	2		1			9
o	7	10					1	2			1		10
			1	2	3	4	5	6	7	8	9	10	

FIGURE 2-10. COLLAPSED VIEW OF AEAP DSM

2.3.3 Hierarchical Decomposition Based on As-Late-As-Possible Rule

In the previous section, the DSM's obtained by hierarchical decomposition are based on the AEAP rule. However, it is not always true that tasks are planned to start as early as possible when they have *slack* (*float*). Thus, we may have different DSM's which essentially show the same structure of a project but with different task sequences. In order to find those alternative DSM's, the method to obtain the DSM based on the *as-late-as-possible (ALAP)* rule is explained in this section (hereafter referred to as *ALAP DSM*). Under the ALAP rule, the starting time of a task is delayed to the extent that it does not delay entire project, *i.e.* it can be delayed within the slack of the task.

The author proposes two alternative methods to obtain the *ALAP DSM*. The first method takes a similar approach to Warfield (1973) by using the reachability matrix. After computing the *ALAP DSM* using the algorithm in Figure 2-11, the collapsed DSM can also be obtained in the same way as in the *AEAP DSM*.

(1) For every task i (i = 1, 2, ..., n) within a project, compute the following sets:

$$R_i = \{ j \mid R^*(i, j) = 1 \; ; \; j = 1, ..., n \}$$
$$A_i = \{ j \mid R^*(j, i) = 1 \; ; \; j = 1, ..., n \}$$

where, R^* : reachability matrix

n : number of tasks in the project

(2) Compute the set L_k for k = 1, ..., m , where m is the maximum level in the project, as follows:

(i) $L_1 = \{ i \mid R_i \cap A_i = A_i \; ; \; i = 1, ..., n \}$

(ii) for k = 2, ..., m,

$$L_k = \{ i \mid R_i' \cap A_i' = A_i' \;\; s.t. \;\; R_i' = R_i - (L_1 \cup \cdots \cup L_{k-1}) \;\; and$$
$$A_i' = A_i - (L_1 \cup \cdots \cup L_{k-1}) \; ; \; i = 1, ..., n \;\; and \;\; i \notin (L_1 \cup \cdots \cup L_{k-1}) \}$$

where $m = \min k \;\; s.t. \;\; (L_1 \cup \cdots \cup L_k) = \{1, ..., n\}$

(3) Construct the DSM from the task i, of which $n(A_i')$ (for L_1, $n(A_i)$) is higher, in L_k for $k = m, ..., 1$.

<Note>

(i) L_k is the set having the tasks in level '$m - k + 1$'.

(ii) If $n(A_i')$ ($n(A_i)$ for L_1) is higher than one, the tasks in A_i' (A_i for L_1) constitute a coupled block.

(iii) '(3)' determines the ordering of tasks in the same level such that a coupled block is located before a single task and, with more than one coupled block, the one with more constituent tasks is located upfront.

FIGURE 2-11. ALGORITHM COMPUTING DSM BASED ON ALAP RULE

The second method uses the *AEAP Collapsed DSM*. Since coupled blocks are collapsed into block tasks, there exists no feedback dependency in this collapsed DSM. After computing the *ALAP Collapsed DSM* using the algorithm in Figure 2-12, the *ALAP DSM* can be constructed by de-collapsing block tasks in the collapsed DSM to coupled blocks.

(1) For every task i (i = 1, 2, ..., p) in the *AEAP Collapsed DSM*, compute the following sets:

$$A_i = \{\, j \mid AEAP(j,\ i) \neq 0\ ;\ j = i,\ldots, p\ \}$$

where, *AEAP*: *AEAP Collapsed DSM* of which (i, i) for i = 1, ..., p is "1"
p: number of tasks in the *AEAP Collapsed DSM*

(2) Compute the set L_k for k = 1, ..., m , where m is the maximum level in the project, as follows:

(i) $L_1 = \{\, i \mid A_i = \{\, i \,\}\ ;\ i = 1,\ldots, p\ \}$

(ii) for k = 2, ..., m,

$$L_k = \{\, i \mid A_i' = \{\, i \,\}\ \ s.t.\ A_i' = A_i - (L_1 \cup \cdots \cup L_{k-1});\ \ i = 1,\ldots, p\ \ and\ \ i \notin (L_1 \cup \cdots \cup L_{k-1})\ \}$$

where $m = \min k\ \ s.t.\ \ (L_1 \cup \cdots \cup L_k) = \{1, \ldots, p\}$

(3) Construct the DSM from block task(s) with more constituent tasks to single task(s) in L_k for $k = m, \ldots, 1$.

<Note>

(i) L_k is the set having the tasks in level '$m - k + 1$'.

(ii) '(3)' determines the ordering of tasks in the same level.

FIGURE 2-12. ALGORITHM COMPUTING ALAP COLLAPSED DSM FROM AEAP COLLAPSED DSM

Figure 2-13 illustrates the results of hierarchical decomposition of the matrix input in Figure 2-5, following the ALAP rule. Note that the levels of tasks as well as the sequence of tasks are changed from the *AEAP DSM*. For example, tasks *b* and *e* are on the second and third in the *ALAP DSM* while they are on the sixth and seventh, respectively, in the *AEAP DSM*. Also, *block1*, which is on level two of the *AEAP DSM*, is located in level four of the *ALAP DSM*. Figure 2-14 illustrates the *ALAP Collapsed DSM*.

Task Name	Level		1	6	7	2	3	4	5	12	9	10	11	8	13	14	15	
a	1	1																1
b	2	6	1													◇		6
e	3	7		1														7
c	4	2	2						1			*block1*				◇		2
d	4	3				2			1									3
g	4	4	1			1												4
j	4	5					1	1										5
f	4	12		1	2													12
h	5	9		2	1								1			*block2*		9
k	5	10									2		2					10
l	5	11										1						11
m	5	8						1	1									8
i	5	13			2					1								13
n	6	14											1	1	1			14
o	7	15										1		1		1		15
			1	6	7	2	3	4	5	12	9	10	11	8	13	14	15	

FIGURE 2-13. HIERARCHICAL DECOMPOSITION BASED ON ALAP RULE

Task Name	Level		1	3	4	2	7	6	5	8	9	10	
a	1	1											1
b	2	3	1								◇		3
e	3	4		1									4
block1	4	2	2								◇		2
f	4	7		1	2								7
block2	5	6		2	1								6
m	5	5				2							5
i	5	8			2		1						8
n	6	9						2	1	1			9
o	7	10						2	1		1		10
			1	3	4	2	7	6	5	8	9	10	

FIGURE 2-14. COLLAPSED VIEW OF ALAP DSM

2.3.4 Level Slack and DSM with Customized Task Sequence

Level slack of a task is defined as the difference between the level in the *ALAP DSM* and that in the *AEAP DSM*. Figure 2-15 illustrates tasks with nonzero level slack based on the *AEAP Collapsed DSM* in Figure 2-10. Dotted boxes are wrapped around the columns of those tasks between the levels they can be located in. For instance, *block1* has level slack of two and can be located between levels two and four. Tasks *m* and *block2* have nonzero level slack as well.

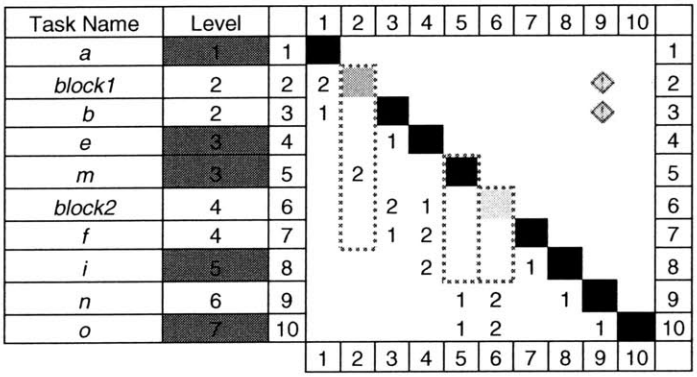


FIGURE 2-15. LEVEL SLACK IN AEAP COLLAPSED DSM

A task with nonzero level slack can be placed in a different hierarchy between its level in the *AEAP DSM* and that in the *ALAP DSM* without affecting the structure of the project. If the task is planned to start with some *lag* (*i.e.* later than the as-early-as-possible schedule), the DSM can show the better structure of the project by moving it to the level where tasks in the same level are possibly performed in parallel with it. Figure 2-16 illustrates the collapsed DSM with *customized* task sequence in which task *m* is moved to level five from level three in the *AEAP Collapsed DSM* in Figure 2-10.

Task Name	Level		1	2	3	4	6	7	5	8	9	10	
a	1	1											1
Block1:	2	2	2										2
b	2	3	1										3
e	3	4			1								4
Block2:	4	6			2	1							6
f	4	7			1	2							7
m	5	5		2									5
i	5	8				2		1					8
n	6	9					2		1	1			9
o	7	10					2		1		1		10
			1	2	3	4	6	7	5	8	9	10	

FIGURE 2-16. COLLAPSED DSM WITH CUSTOMIZED TASK SEQUENCE

2.4 Information Dependency Classification

The method presented in this section classifies information dependencies by analyzing the results from hierarchical decompositions. Through this classification process, a smaller number of *precedence relationships* between tasks are drawn from *information dependencies* between tasks. Thus, it reduces the number of constraints, hence complexity.

2.4.1 Definitions of Various Dependencies

The method first classifies information dependencies into two categories: *binding* and *non-binding*. The underlying purpose of this classification is to identify redundant information dependencies and to reduce complexity by eliminating redundant constraints. A *binding dependency* represents the dependency which is regarded as a constraint between two dependent tasks. The delay of information transfer from the upstream task *directly* causes the downstream task to slip. This information dependency is translated to a precedence task relationship of "finish-to-start" for scheduling analyses in the following chapters. Note that all dependencies within coupled blocks are regarded as binding. A *non-binding dependency* represents the dependency which is *not* regarded as a constraint

between two dependent tasks. The delay of information transfer does not directly impact the schedule even though there is information flow between two tasks. Thus, it is regarded as a redundant constraint and omitted from further scheduling analyses.

Figure 2-17 illustrates binding and non-binding dependencies in a simple example. Two feasible scenarios are given in (b) and (c) that correspond to the DSM in (a). The scenario one is the case that all three tasks generate single outputs and the scenario two is the case that task *a* generates multiple (two) outputs while other two tasks generate single outputs. The following examples in part designs would explain such scenarios in which task *a* is the design of part A, task *b* is the design of part B, and task *c* is the design of part C:

> *Scenario one: part A has interfaces with both parts B and C while parts B and C also have a common interface. The design of part B needs dimensions of part A before starting, and the design of part C needs dimensions of part B while it also requires dimensions of part A.*
>
> *Scenario two: part A is connected with part B while parts B and C have a common interface. The design of part B needs dimensions of part A before starting, and the design of part C needs dimensions of part B while it also requires information about the material of part A to analyze structural performance.*

In both scenarios, even though task *c* needs information from task *a*, it does not give a precedence constraint because task *a* must have been completed by the time task *c* starts after task *b* is finished. Therefore, information dependency between tasks *a* and *c* is *non-binding* while other two dependencies are *binding*. However, this non-binding dependency is also significant because people who perform task *c* should know from whom or where they can find inputs to work.

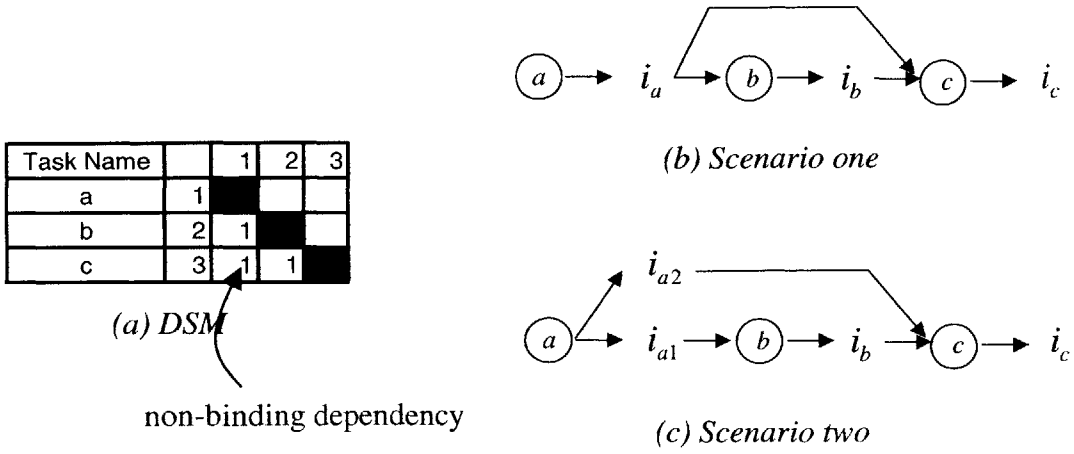


FIGURE 2-17. AN EXAMPLE OF BINDING AND NON-BINDING DEPENDENCIES

Next, the method classifies two types of a binding dependency. A *critical binding dependency* is a binding dependency between the tasks with zero level slack. A *non-critical binding dependency* is a binding dependency which is not critical-binding. Following critical binding dependencies along the tasks, the path goes through the entire process hierarchies of a project from the first level to the last level. Kusiak et al. (1995) called this path as a *critical dependency path* (however, they did not provide the method or algorithm to identify this path). This path is different from a critical path because it is determined without considering time aspects. However, this path provides guidance for process improvements in the early stage of planning processes when detailed data for durations are not available. Particular attention needs to be paid to critical binding dependencies in order to shorten the lead time.

Figure 2-18 illustrates various dependencies in the collapsed DSM in Figure 2-15. The marks at (7, 3), (10, 5), (6, 3), (8, 4) and (10, 6) represent non-binding *dependencies.* The marks in latter three cells are converted from "2" in the *AEAP Collapsed DSM* to "1" marks since there are no benefits from overlapping between the tasks linked by non-binding dependencies.

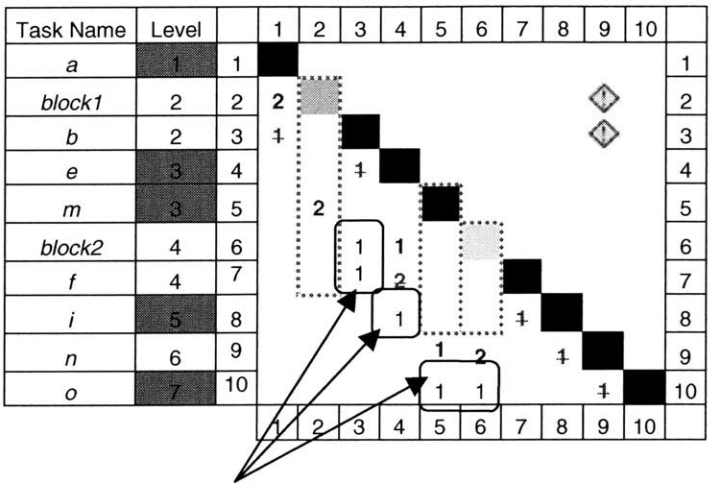
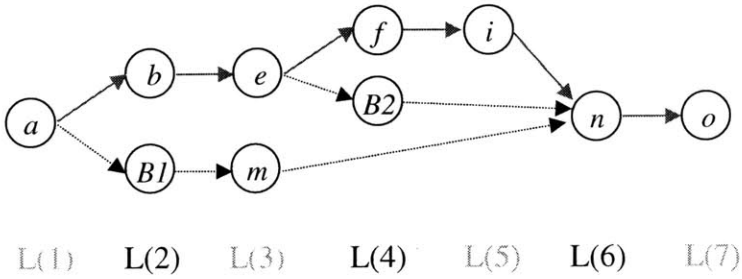
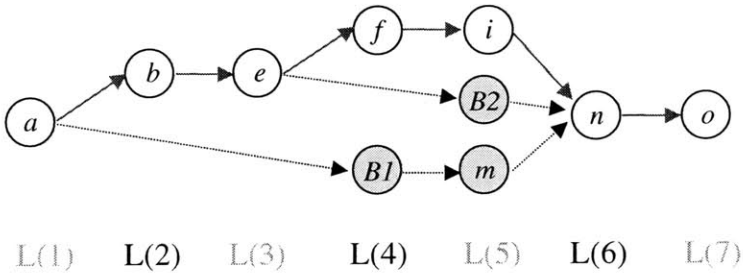


FIGURE 2-18. AN EXAMPLE OF INFORMATION DEPENDENCY CLASSIFICATION

The network diagrams in Figure 2-19 show the equivalent structure of the above collapsed DSM except that non-binding dependencies are omitted. It is noticed that block tasks *B1* and *B2* as well as singe task *m* can be placed in different levels since they have nonzero level slack. The path along the tasks *a-b-e-f-i-n-o* is a critical dependency path of the project linked by critical binding dependencies. The dotted arrows indicate non-critical binding dependencies between tasks. The dependency between tasks *b* and *f* in the collapsed DSM, for instance, is omitted since it is non-binding. Note that the coupled blocks collapsed in these analyses are expanded in the modeling and scheduling analyses in the following chapters.



(a) Based on the AEAP Rule



(b) Based on the ALAP Rule

FIGURE 2-19. NETWORK DIAGRAMS CORRESPONDING TO COLLAPSED DSM'S

2.4.2 Algorithms for Information Dependency Classification

Since all dependencies within coupled blocks are regarded as binding, the algorithm receives initial values from the *AEAP Collapsed DSM* and classifies into the categories explained earlier. For complete classification of information dependencies, the method follows two steps: (1) classifies all dependencies into two groups – binding and non-binding, (2) classifies binding dependencies into two groups – critical binding and non-critical binding. Two alternative algorithms for the first step of classification are proposed. The first one uses matrix manipulations while the second uses process levels. The algorithm for the second step of classification also uses process levels.

2.4.2.1 *Algorithms Classifying Binding and Non-Binding Dependencies*

The method assumes that, the downstream task cannot be completed before the upstream task finishes when tasks are overlapped, as illustrated in Figure 2-20. Figure 2-21 and Figure 2-22 show the algorithms that classify binding and non-binding dependencies.

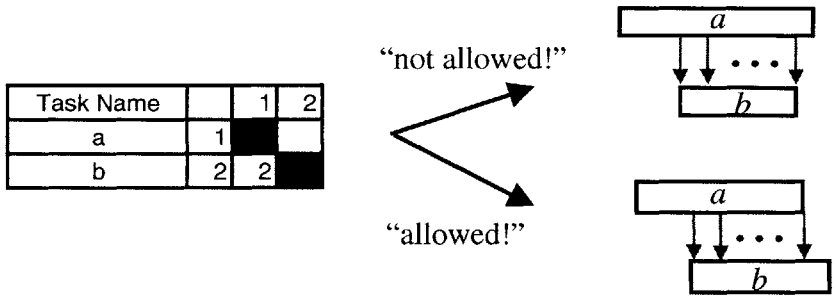


FIGURE 2-20. ASSUMPTION FOR OVERLAPPED TASKS

For $i = 2, 3, \ldots, n$, where n: number of tasks in the *AEAP Collapsed DSM*,

(1) Set $M(i, j) = DSM(i, j)$ for $j = 1, \ldots, i - 1$

where, $DSM(i, j)$: number in the *AEAP Collapsed DSM* representing the type of dependency between tasks i and j

(2) For $i > 2$, compute the matrix M by executing the following loop:

```
for ii = 2 to i
   for j = i –2 to 1 decreasing by 1
      for k = j + 1 to i – 1
         M(ii, j) = M(ii, j) + M(k, j)  × M(ii, k)
      next k
   next j
next ii
```

(3) For $j = 1, \ldots, i - 1$ *s.t.* $DSM(i, j) \neq 0$,

if $M(i, j) > DSM(i, j)$ for $j = 1, \ldots, n$,

$(i, j) \in Q_1$: a set of non-binding dependencies

otherwise, $(i, j) \in Q_2$: a set of binding dependencies

FIGURE 2-21. ALGORITHM <1> TO CLASSIFY BINDING AND NON-BINDING DEPENDENCIES

For $i = 2, 3, \ldots, n$, where n: number of tasks in the *AEAP Collapsed DSM*,

(1) Identify the following two sets:

$$E_i = \{ j \mid AEAP(i, j) \neq 0 \text{ and } L_{AEAP}(j) + 1 = L_{AEAP}(i) \ ; \ j = 1,\ldots,i-1 \}$$

$$L_{seq(i)} = \{ seq(j) \mid ALAP(i, j) \neq 0 \text{ and } L_{ALAP}(j) + 1 = L_{ALAP}(i) \ ; \ j = 1,\ldots,i-1 \}$$

where, $AEAP(i, j)$: number in the *AEAP Collapsed DSM* (0, 1 or 2)

$ALAP(i, j)$: number in the *ALAP Collapsed DSM* (0, 1 or 2)

$L_{AEAP}(i)$: level of task i in the *AEAP Collapsed DSM*

$L_{ALAP}(i)$: level of task i in the *ALAP Collapsed DSM*

$seq(i)$: function that maps the index i in the *ALAP Collapsed DSM* to that in the *AEAP Collapsed DSM*

(2) For $j = 1, \ldots, i - 1$ *s.t.* $j \in B_i' = (E_i \cup L_i)$,

$(i, j) \in Q_2' \subset Q_2$: a set of binding dependencies

FIGURE 2-22. ALGORITHM <2> TO CLASSIFY BINDING AND NON-BINDING DEPENDENCIES

Note that the algorithm <2> does not give a complete set of binding and non-binding dependencies. Figure 2-23 illustrates the example of the binding dependency which is *not* identified by the algorithm <2>. In this example, the dependency between tasks *b* and *g* is not counted as binding even though it actually is. These exceptions rarely happen when two tasks, of which the process levels differ by more than one, are placed on parallel paths that have the same length of process levels. Thus, this algorithm can be used in less computation time with marginal errors. When the difference between the levels of two tasks is two as in the figure, the exceptions belong to the following set Q_2'':

$$Q_2'' = \{ (i, j) \mid i = 1, 2, \ldots, n \ ; \ j = 1, \ldots, i - 1 \text{ and } j \in B_i'' \}$$

where, $B_i'' = \{ j \mid L_{AEAP}(j) + 2 = L_{AEAP}(i) \ ; \ j \notin B_k' \text{ for } \forall k \in B_i' \text{ and } j = 1,\ldots,i-1 \}$

When the difference of the levels is over two, a set has a more complex form. However, the dependencies in this set rarely constrain the project if considering durations of tasks.

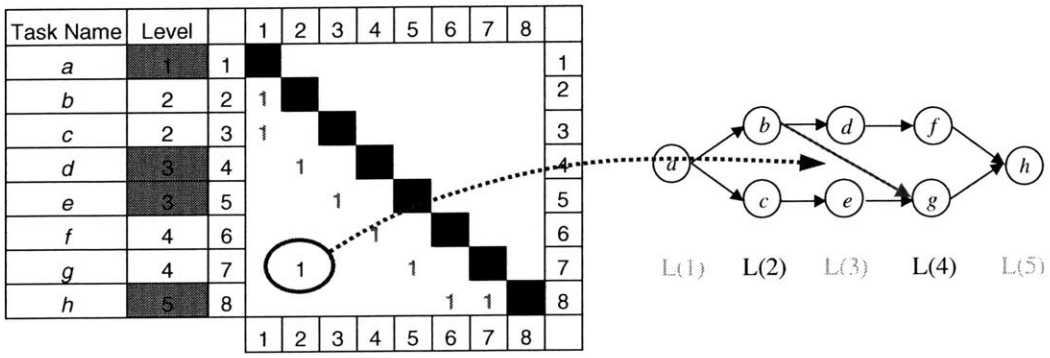


FIGURE 2-23. AN EXCEPTION OF ALGORITHM <2>

2.4.2.2 Algorithm Classifying Critical and Non-Critical Binding Dependencies

$Q_3 = \{ (i, j) \in Q_2 \mid L_{AEAP}(i) = L_{ALAP}(seq^{-1}(i)) \text{ and } L_{AEAP}(j) = L_{ALAP}(seq^{-1}(j)) \}$

: a set of critical binding dependencies

$Q_4 = \{ (i, j) \in Q_2 \mid (i, j) \notin Q_3 \}$

: a set of non-critical binding dependencies

where, $L_{AEAP}(i)$: level of task i in the *AEAP Collapsed DSM*

$L_{ALAP}(i)$: level of task i in the *ALAP Collapsed DSM*

$seq^{-1}(i)$: function that maps the index i in the *AEAP Collapsed DSM* to that in the *ALAP Collapsed DSM*

FIGURE 2-24. ALGORITHM <2> TO CLASSIFY BINDING AND NON-BINDING DEPENDENCIES

2.5 Chapter Summary

In this chapter, the process-structuring methods are presented using the design structure matrix. The methods structure complex information flows in the project and decompose processes into hierarchies. And they identify iteration loops and distinguish planned iteration from unplanned iteration. Through the information dependency classification process, a critical dependency path is identified and redundant dependencies are eliminated in further analyses in the following chapters. Improvements can be made from a structural view using the results of analyses in the early stage of planning. The methods are used as the structuring module in the integrated project management framework.

CHAPTER 3

Process-Modeling Methods Using Advanced Simulation

3.1 Introduction

In this chapter, a product development process modeling and analysis technique using advanced simulation is presented. The model is a second-generation, DSM-based, dynamic process simulation model that can incorporate many more general characteristics of complex product development processes, including the following features:

- modeling dynamic iteration characteristics among sequential, parallel and overlapped tasks
- resolving resource confliction in the iterative project network
- taking into account the normal variance of development time
- assessing schedule risks of iterative processes as they progress
- calculating slack and identifying critical sequences in a resource-constrained iterative project

The model uses the design structure matrix representation to capture the information flows between tasks. In each simulation run, the expected durations of tasks are initially sampled using the Latin Hypercube Sampling method and decrease over time as the model simulates the progress of dynamic stochastic processes. It is assumed that the rework of a task occurs for the following reasons: (1) new information is obtained from overlapped tasks after starting to work with preliminary inputs, (2) inputs change when other tasks are reworked, and (3) outputs fail to meet established criteria.

In order to analyze such a rich process model, numerical simulation methods are employed. The model differs from the Virtual Design Team framework (Levitt et al. 1994) in that the objective of the VDT simulation is to predict organizational breakdowns

in performing activities while the goal of this model is to predict dynamic behavior of iterative processes.

3.2 Model Inputs

Model inputs explained in this section characterize behaviors of individual tasks and interactions among the tasks from a schedule perspective. The duration of a task is used to model uncertainty and complexity within the domain of the task. Precedence and resource constraints are used to determine the boundaries of tasks along the time line. Iterations are modeled to predict the patterns of workflows caused by dynamic information exchanges among the tasks.

3.2.1 Task Durations

The model uses the triangular probability distribution to represent the characteristic of a task duration since it offers comprehensibility to a project planner (Williams 1992a). For each task ($i = 1, \ldots, n$), the model receives three estimated durations – optimistic $(d_{opt})_i$, most likely $(d_{likely})_i$ and pessimistic $(d_{pess})_i$ as in some PERT-based analyses – for the expected duration of *one-time* execution. The expected duration is the duration between the start and end of its continuous work even though the task may iterate more than once afterwards. Remaining duration, d_i of task i decreases over time as the model simulates the project's progress. The model also receives actual duration $(d_{act})_i$ if the task has been in progress. The original duration is defined as $(d_{ori})_i \equiv (d_{act})_i + d_i$ representing the duration of task i at the start of the simulation.

It has been found that assessing the 10th and 90th percentiles of the expected duration is more reliable than the extremes of the PDF which are typically outside the realm of experience (Williams 1992a, Keefer and Verdini 1993). The model uses the Latin Hypercube Sampling (LHS) method (McKay et al. 1979) to incorporate the uncertainty of the expected duration of a task based on three estimated durations. After calculating

the extreme values of the PDF, it divides the range between them into N strata of equal marginal probability $1/N$ where N is the number of random values for the expected duration representing a known triangular PDF. Then, it randomly samples once from each stratum and sequences the sampled values randomly. Figure 3-1 illustrates the LHS procedure.

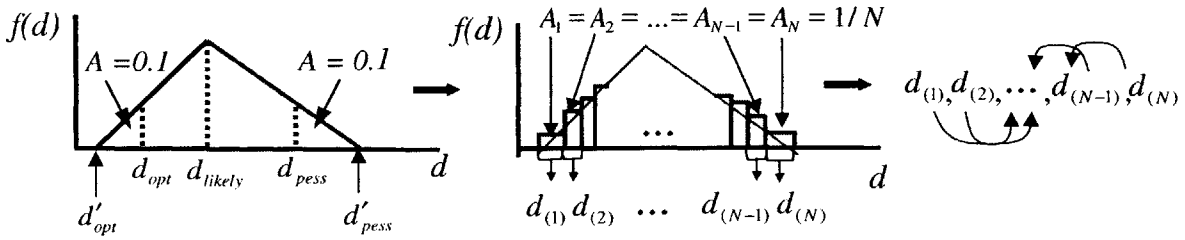


FIGURE 3-1. LATIN HYPERCUBE SAMPLING

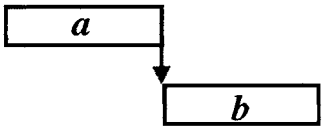
3.2.2 Precedence Constraints

In the previous chapter, two types of information flows are defined from a schedule perspective. Using information flows and types of transfer patterns documented in the DSM, precedence task relationships are drawn in order to extend the information-based approach to the scheduling model.

The first type of information flow (representing the case that the downstream task requires final information from the upstream task to begin its work) is translated to a "finish-to-start" precedence constraint between two tasks. The second type of information flow (representing the case that the downstream task uses final information in the middle of its process and/or begins with preliminary information but also receives a final update from the upstream task) is translated to a "finish-to-start plus lead" constraint. The DSM is used to document these information flows and precedence constraints. The notation *DSM(i, j)* for $i, j = 1, \ldots, n$ is used to represent this two-type scheme in the DSM where:

- *DSM(i, j)* = 0 when there is no information flow from task *j* to task *i*
- *DSM(i, j)* = 1 when there is the finish-to-start type of information flow from task *j* to task *i*
- *DSM(i, j)* = 2 when there is the finish-to-start-plus-lead type of information flow from task *j* to task *i*

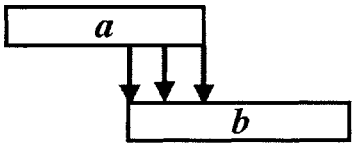
Figure 3-2 illustrates types of information flows and precedence constraints between tasks *a* and *b*. Note that each of three arrows in (b) represents the second type of information flow while it is modeled as the same precedence constraint.



Task Name		1	2
a	1		
b	2	1	

DSM(i, j) *(i, j = 1, 2)*

(a) Finish-to-Start



Task Name		1	2
a	1		
b	2	2	

DSM(i, j) *(i, j = 1, 2)*

(b) Finish-to-Start + Lead

FIGURE 3-2. INFORMATION FLOWS AND PRECEDENCE CONSTRAINTS

3.2.3 Resource Constraints

The model assumes that there exists a fixed resource pool throughout the entire project duration. It consists of specialized resources and/or resource groups of which constituents exhibit the same functional performance. Each task has its own resource requirement which is assumed to be constant over the entire period the task is processed. When two or

more tasks are competing for limited resources in a certain period of time, *i.e.* resources are over-allocated, the model determines priorities by the heuristic rules which are explained later in this chapter.

3.2.4 Iteration

Eppinger et al. (1997) defined iteration as the repetition of tasks to improve an evolving development process. In the thesis, iteration is referred to as rework caused by other tasks without including repetitive work within a single task (variance in duration). The model assumes that planned rework of a task is generated due to the following causes (similar to the explanations of Smith and Eppinger (1997a), Eppinger et al. (1997), Browning and Eppinger (2000)):

- receiving new information from overlapped tasks after starting to work with preliminary inputs
- probabilistic change of inputs when other tasks are reworked
- probabilistic failure to meet the established criteria

In the proposed model, the first cause gives rise to overlapping iteration, and the second and the third causes give rise to sequential iteration. Parallel iteration of a limited number of tasks is simulated in this model by combining overlapping and sequential iteration.

3.2.4.1 Overlapping Iteration

Overlapping has been described as a "core technique for saving development time" (Smith and Reinertsen 1995). It is generally acknowledged that overlapping tasks may save time, but is more costly than the traditional sequential approach. Following the definition of task relationships in this work, however, this may be true only when the overlapping implies the transfer of incomplete preliminary information. This is also related to how to define a task. If the tasks are defined such that all information exchanges take place at the end of tasks, the project network could be modeled using only a sequential iteration model.

Suppose two dependent tasks are overlapped sequentially and the downstream task starts to work with the preliminary information from the upstream task. As the upstream task evolves, its output information also evolves to its final form while the new information generated since its initial transfer of the preliminary information gets released according to its communication policy. The downstream task may repeat the part of its work to accommodate this new information which is unnecessary if it started to work with the final information from the upstream task. In this model, it is assumed that overlap amounts as well as expected rework impacts between the two tasks can be estimated in the planning stage.

The model uses the DSM representation as shown in Figure 3-3. The notation *OA(i, j)* is used for maximum overlap amount and *OI(i, j)* for overlap impact for $i, j = 1, \ldots, n$. The former represents the planned overlap amount between tasks i and j and it is a fraction of the expected duration of task i. This carries the assumption that the downstream task cannot be completed before the upstream task finishes (this is in accordance with the assumption in the previous chapter). The latter represents the expected overlap impact when task i is overlapped with task j by the amount $OA(i, j) \times d_i$ and it is a fraction of that amount. $OI(i, j) = 1$ implies no benefit from overlapping. To implement overlapping strategy, it should be reasonably less than 1 considering additional risk due to the evolution of volatile preliminary information.

In Figure 3-3, task b starts with preliminary information from task a. It is planned to begin earlier with preliminary information and expected to finish 20% of its work before task a gives a final update. However, it is also expected to rework half of work done through overlapping to incorporate updated information from task a. Thus, lead time is reduced by 10% of d_2 from this overlapping.

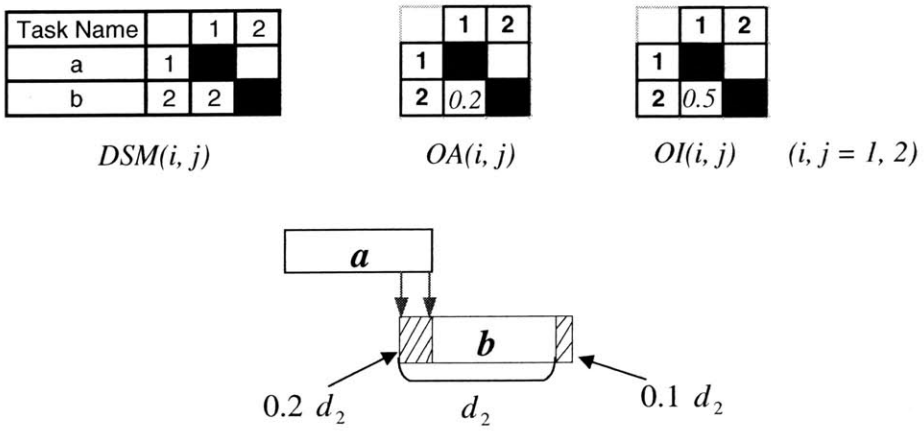


FIGURE 3-3. OVERLAP AMOUNT AND IMPACT BETWEEN TWO TASKS

3.2.4.2 Sequential Iteration

As explained in the previous chapter, the model distinguishes unplanned iteration from planned iteration, and incorporates the effects of only *planned* iterations when computing the lead time of a project (while it also allows for assessing the effects of *unplanned* iteration). Thus, the model accounts for sequential iteration caused by feedback marks only within coupled blocks. It takes an approach similar to Browning and Eppinger (2000) by explaining sequential iteration using rework probability, rework impact and the learning curve.

Rework probability is a measure of uncertainty in sequential iteration. *RP(i, j, r)* represents the probability that task *i* does rework affected by task *j* in *r*th iteration for *i, j* = 1, ..., *n* and *r* = 1, 2, ... In the case of $i < j$, it represents the *feedback* rework caused by the change of information from downstream task *j* or by the failure of downstream task *j* to meet the established criteria. In the case of $i > j$, it represents the *feedforward* rework that downstream task *i* needs to do since upstream task *j* has generated new information after it has done its own rework. As the development processes converge to their final solutions with iterative rework, there are less chances that new information is generated

and errors are discovered. Therefore, rework probability tends to decrease in each iteration.

Rework impact is a measure of the level of dependency between tasks in sequential iteration. *RI(i, j)* represents the percentage of task *i* to be reworked when rework is caused by task *j* for *i, j* = 1, …, *n*. Rework impact is assumed to be constant in each iteration.

The learning curve measures a characteristic of a task when it repeats. $(L_{ori})_i$ for $i = 1, \ldots, n$ represents the percentage of original duration when task *i* does the same work for a second time. The model assumes that the learning curve decreases by $(L_{ori})_i$ percent in each repetition until it reaches $(L_{\max})_i$ which is the minimum percentage of original duration when task *i* does the same work repeatedly. Thus, rework amount is calculated as the original duration multiplied by rework impact and learning curve. Figure 3-4 shows the rework probability and impact for sequential iteration using the DSM representation, and Figure 3-5 illustrates the learning curve.

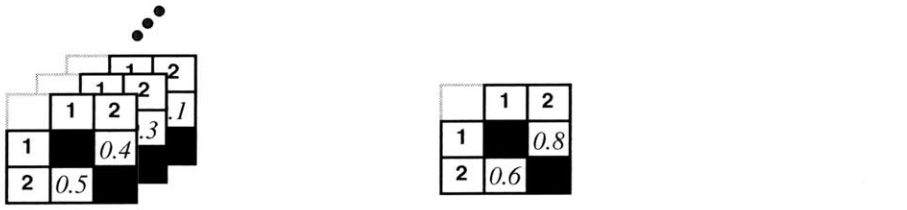


FIGURE 3-4. REWORK PROBABILITY AND IMPACT

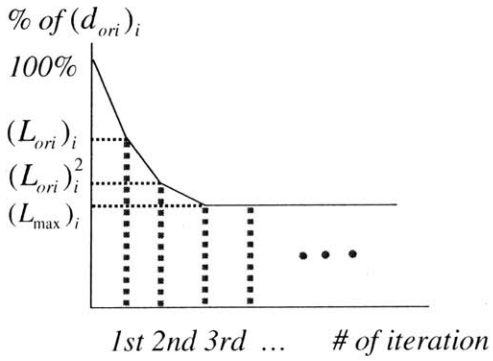


FIGURE 3-5. LEARNING CURVE

3.3 Description of the Basic Model

The model employs the *parallel discrete event simulation* (Pritsker and O'Reilly (1999), Zeigler et al. (2000)) to compute the distribution of lead time. Analytical features are included so that the model can describe the complex behavior of development processes having overlapped tasks and sequential iterations. This section explains the underlying structure of the simulation-based model. The detailed procedure and model variables are presented in the section *A1* of the *Appendix.*

3.3.1 Model Structure and Algorithm

In discrete time systems, the model assumes a discrete mode of execution. It is in a particular state at a particular time instant and advances to the next state in a sequential manner along the time line. In the discrete event simulation, events trigger state transitions and time advances in discrete steps by the time elapsed between events. The distinctive feature of discrete *event* simulation is that no components within a system need to be scanned during times between events (in contrast, the discrete *time* simulation requires that scanning must be performed at each time step). A parallel simulation allows multiple model components to be active and to send their outputs to other components.

The model uses different expected durations of tasks in each simulation run which are initially sampled using the Latin Hypercube Sampling method. With those durations of tasks, it simulates a series of sequential state transitions incorporating iterations in multiple paths. States are determined dynamically based on all the inputs of tasks and task interfaces explained earlier. In each state, it scans all tasks and determines a set of active tasks satisfying both precedence and resource constraints. If the amount of resources required by the tasks satisfying precedence constraints exceeds the resource capacity of the project, resource assignments are made by the heuristic priority rules. It assumes that a task begins to work as early as possible when it has all the necessary inputs from upstream tasks and all the required resources.

An event is defined as the completion of a task instead of any information transfer. Thus, when any active task in the current state q is completed, the model makes a transition to the next state 'q+1'. The duration of state q, D_q (q = 0, 1, 2, ...) is defined as the minimum remaining duration of active tasks in the state. Before making a transition to the next state, the model subtracts the duration of the current state from the remaining durations of all active tasks. If all the remaining durations of tasks are zero, one simulation run is complete and the lead time is calculated as the sum of all the state durations. After N simulation runs, the probability distribution of lead time can be displayed.

Figure 3-6 summarizes the algorithm to compute the lead time in one simulation run. A simulation run starts with initializing model variables from the model inputs at STEP1. It simulates time advance of tasks by following STEP2 through STEP7 in each state until it satisfies the termination condition.

For each simulation run,

STEP1. Initialize model variables from the inputs at state *0*.

STEP2. Initialize model variables in the current state q.

STEP3. Identify a set of concurrent active tasks in the current state satisfying precedence and resource constraints based on the priority rules.

STEP4. Adjust overlapping iteration.

STEP5. Adjust the durations of the active tasks and the lead time.

STEP6. Generate sequential iteration rework.

STEP7. Make a transition to the next state q+1 or finish the simulation run if satisfying the termination condition.

FIGURE 3-6. ALGORITHM FOR COMPUTING LEAD TIME IN THE BASIC MODEL

3.3.2 *Overlapping Iteration in Multiple Resource-Constrained Paths*

In each state, the model identifies a set of active tasks which have started to work in the current state. For each task in this set, the model simulates that its overlapped part of work has been performed in prior state(s) and the expected impact due to iterations has been added to the projection in the current state. The overlap amount of a task is dynamically determined by both precedence and resource constraints with other tasks in multiple paths. The model assumes that overlapped part of work has a lowest priority for limited resources and does not start to do unless resources are secured during its processing time. When the amount of overlap is different from the planned amount with any information-providing task, it computes its overlap impact, assuming that it is linear to the overlap amount. If a task is overlapped with multiple tasks, the overlap impact is between the maximum of single impacts and the sum of them depending on the amount of duplicate rework caused by those tasks. In this case, the model takes the latter as a default. When a task is overlapped with any upstream task that has been reworked, the model assumes that the overlap amount cannot exceed the rework amount of the upstream task. Finally, the overlap amount is subtracted from the remaining duration of the active task and the overlap impact is added to it.

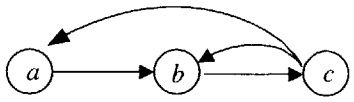
3.3.3 *Rework Generation for Sequential Iteration in Multiple Paths*

In each state, the model identifies a set of active tasks which are supposed to be completed in the current state. It is those tasks that cause state-transition events. For each task in the set, the model determines whether it causes feedback and/or feedforward rework to other tasks. Rework decisions are simulated by comparing each rework probability with a random number. When rework occurs, the amount of rework is computed by the original duration of a task doing rework multiplied by a rework impact adjusted for learning curve benefit. Finally, rework amounts are added to the remaining durations of those tasks that are determined to rework. Note that feedback rework in an upstream task can also cause successive feedforward rework in subsequent states to the downstream tasks that have been in process or completed before. The model also

simulates that a rework decision can be made before final output information is produced through overlapping. The section *A2* in the *Appendix* explains the details of how the proposed method differs from the analytical methods using a Markov chain (Ahmadi and Wang 1994 and 1999), a reward Markov chain (Smith and Eppinger 1997a), and a signal flow graph (Eppinger et al. 1997), and from the first DSM-based simulation method (Browning and Eppinger 2000).

3.3.4 Rework Concurrency

When tasks iterate sequentially, a choice of rework policy may allow rework concurrency to shorten the lead time. For illustration, consider the simple example given in Figure 3-7. The information flow diagram and its corresponding DSM show three sequential tasks with feedback loops. In all other process models for sequential iterations surveyed in the literature, when both tasks *a* and *b* require new information from task *c*, the rework of task *b* cannot begin before the completion of the rework of task *a*. This is based on the underlying assumption that the precedence constraint between tasks *a* and *b* should also be respected when tasks iterate. However, a project manager may prefer a different policy when there is a small chance that task *a* will produce new information also causing task *b* to rework. By performing the rework of both tasks *a* and *b* concurrently, the lead time can be reduced with small additional risk.



(a) Information Flow Diagram

Task Name		1	2	3
a	1			1
b	2	1		1
c	3		1	

(b) DSM

FIGURE 3-7. AN EXAMPLE FOR TASK CONCURRENCY

We introduce the *Rework Concurrency (RC)* to model this strategic decision upon task concurrency during iteration. It represents total *direct* and *indirect* feedforward rework probabilities which control the level of concurrency in sequential iteration. The *Rework*

Concurrency is a lower-triangular matrix which takes *direct* rework probabilities from *RP(i, j, 1)* ($i = 2,\ldots, n; j = 1,\ldots, i\text{-}1$) and adds them with *indirect* rework probabilities. The *indirect* probability *(i, j)* represents the probability of task *i* doing rework caused by task *j* through the intermediary of other tasks between *i* and *j*. For example, *RC(5, 2)* in Figure 3-8 is computed as the sum of indirect rework probabilities between tasks 2 and 5 through tasks 3 and 4 as intermediaries ($0.5\times0.1 + 0.5\times0.4 = 0.25$).

RC(i, j) ($i > j$) is computed at STEP3 of the simulation algorithm when determining the concurrency of tasks *i* and *j* when task *j* is reworked. The model assumes that a task can be performed even though there exists an upstream dependent task being reworked if the total rework probability between the two tasks in the *RC* is less than the probability $P_{tolerance}$, a pre-determined *rework risk tolerance*. The algorithm to compute the *Rework Concurrency* is included in the algorithm explained in the section *A1* of the *Appendix*.

In the above example, if $RC(2, 1) < P_{tolerance}$, the model simulates that both tasks *a* and *b* are reworked concurrently when both must be reworked. Otherwise, task *b* waits until the rework of task *a* is completed, at which time, new information from task *a* becomes available. In some cases, it may not be necessary for task *b* to use any of the new information from the rework of task *a*. However, if the rework of task *a* does create additional rework for task *b*, the total amount of rework of task *b* is between the maximum and the sum of reworks generated by tasks *a* and *c*. The model uses the latter as the default amount of rework required for task *b* and assumes that this quantity cannot exceed the task's original duration diminished by the learning curve effect.

	1	2	3	4	5	6
1						
2	0.5		0.4			
3		0.5				
4		0.5			0.1	
5			0.1	0.4		
6					0.4	

RP(i, j, 1)

	1	2	3	4	5	6
1						
2	0.50					
3	0.25	0.50				
4	0.25	0.50	0.00			
5	0.13	0.25	0.10	0.40		
6	0.05	0.10	0.04	0.16	0.40	

RC(i, j) $(i, j = 1,\ldots, 6)$

FIGURE 3-8. AN EXAMPLE OF REWORK CONCURRENCY

3.3.5 Measures and Rules for Resource Priorities

The heuristic priority rules that Cooper (1976) proposed are no longer applicable in the project network where tasks iterate sequentially in a probabilistic manner. In this paper, a *rework-adjusted rank positional weight* is proposed as a good measure that can help a project manager to determine task priorities. The rank positional weight is one of the measures that Cooper found among the best toward minimizing total lead time in the project network without iteration. (See the section *A3* in the *Appendix* for the definition.) We define the *rework-adjusted duration* of a task as the expected value of the sum of the duration of its first execution and the total amount of successive rework it creates for its predecessors, assuming no resource constraints in the project network. Then, the *rework-adjusted rank positional weight* is computed by replacing the deterministic duration of a task in Cooper's definition with the rework-adjusted duration. This measure of task priority is calculated before computing lead time *with* resource constraints.

The model determines priorities by the heuristic rules whereby a task has a higher priority if:

(1) it has been in process
(2) it has a higher user-specified priority
(3) it has a higher rework-adjusted rank positional weight
(4) it is sequenced more upstream
(from (1) to (4) in order of significance)

The above priority rule is toward minimizing lead time of a coupled block. Rule (1) implies that a task cannot be interrupted once it has started (non-preemption). Thus, the model does not allow splitting of a task due to its resource constraints. The user-specified rule in (2) can be used when resource priorities should be determined considering different project objectives. Rule (3) stipulates that a higher priority is given to the task that exposes the project to higher schedule risk.

3.4 Description of the Extended Model

In the CPM/PERT method, precedence relationships between tasks and durations of tasks are the sufficient inputs for analyses. Slack (float) – hereafter referred to as conventional slack – and a critical path are defined based on those two inputs. Combined with Monte Carlo methods, it can also take into account normal variance of task durations. However, as Weist (1964) pointed out, the definition of conventional slack is no longer valid when a project network is determined by resource constraints as well as precedence constraints. Thus, Bowers (1995) defined resource-constrained slack – hereafter referred to as RC slack – as follows:

> *"the time by which a single task can be extended or delayed without affecting the project duration, assuming that the resource allocation is unchanged".*

Using this definition, a *critical sequence* (Weist 1964) is defined as the sequence of tasks that have zero RC slack following both precedence and resource links.

However, the above definitions cannot be directly used in the iterative project network where a *probabilistic* rule is applied. The following questions illustrate this restriction: if a task has no slack in its first execution while having some slack in its first iteration, would we use the average slack as a measure of task importance? What if the task repeats twice in another scenario? For this reason, slack is not considered in all the process models for iteration presented in the literature. In this section, the extended model is presented to compute the RC slack of tasks as well as the lead time in a resource-constrained iterative network. Critical sequences are also identified after computing RC slack using this extended model.

3.4.1 Model Structure and Algorithm

In the extended model, the model assumes that sequential iteration takes place *only* among the tasks within the coupled blocks that are identified by the structuring analyses in the previous chapter. Then, the slack and criticality of tasks can be computed by

treating those coupled blocks as if they were single tasks having separate resource requirements. (Hereafter, the coupled blocks that are collapsed in the extended model are referred to as block tasks while the tasks not belonging to any coupled block is referred to as single tasks.)

Figure 3-9 summarizes the procedure and interim outputs of this extended model. The simulation starts with computing the probability distributions for expected durations of coupled blocks. The basic model presented in the previous section is applied m times individually where m is the number of coupled blocks in the project. Then, the model computes the lead time and RC slack in the network without sequential iteration. Figure 3-10 summarizes the algorithm in one simulation run of STEP4 in Figure 3-9, including forward and backward pass computation of the lead time.

While the extended model has the same simulation-based underlying structure of the basic model, there are several differences, as discussed below.

In order to compute RC slack, the extended model performs both forward and backward pass computations. Since it assumes no sequential iteration in the project where coupled blocks are collapsed, the procedure in the basic model that generates rework for sequential iteration is omitted. The model takes separate overlap inputs between block and single tasks (or between block tasks when two coupled blocks can work in parallel). Without specifying these amounts, a downstream task that is dependent upon any task within a coupled block can start only when *all* constituent tasks in the coupled block finish iterations. Thus, if $P_{tolerance} > 0$, the computed lead time from the extended model is always equal to or longer than that from the basic model unless the overlapping is specified between block and single tasks.

When adjusting overlapping iteration, the extended model modifies original durations of tasks as well as remaining durations of tasks. And it uses those adjusted original durations as remaining durations at the start of the backward pass computation. This procedure eliminates the overlapping adjustment step in the backward pass computation.

For each coupled block,

STEP1. Forward pass computation *without* resource constraints, having one of N LHS-sampled durations for constituent tasks

Output: Measures for task priorities for resources among the tasks within the coupled block

STEP2. Forward pass computation *with* resource constraints, having one of N LHS-sampled durations for constituent tasks

Output: Probability distribution for the expected duration of the coupled block based on N simulation results

For the entire project with coupled blocks collapsed into block tasks having the probability distributions for the expected durations of those blocks computed in STEP2,

STEP3. Forward and backward pass computation *without* resource constraints, having the *most likely* durations of single tasks and *average* durations of block tasks

Output: Measures for task priorities for resources among the single and block tasks

STEP4. Forward and backward pass computation *with* resource constraints, having one of N LHS-sampled durations for single tasks and one of N simulated durations for block tasks

Output:

(i) Probability distribution of the lead time based on N simulation results
(ii) Average and standard deviation of RC slack for each task
(iii) Major critical sequences with percentage of chances that they are critical

FIGURE 3-9. PROCEDURES AND INTERIM RESULTS OF THE EXTENDED MODEL

For each simulation run,

<Forward pass computation>

STEP1. Initialize model variables from the inputs at state *0*.

STEP2. Initialize model variables in the current state q.

STEP3. Identify a set of concurrent active tasks in the current state satisfying precedence and resource constraints based on the priority rules.

STEP4. Create temporary precedence constraints for backward pass computation to the tasks that have all the inputs but are delayed due to lower priorities for required resources.

STEP5. Adjust overlapping iteration and modify original durations accordingly.

STEP6. Adjust the durations of the active tasks and the lead time.

STEP7. Make a transition to the next state $q+1$ or finish the forward computation of the simulation run if satisfying the termination condition.

<Backward pass computation>

STEP8. Initialize model variables from the inputs at state *0*.

STEP9. Initialize model variables in the current state q.

STEP10. Identify a set of concurrent active tasks in the current state from the most downstream task having nonzero remaining duration, which satisfy precedence (including additional constraints generated during the forward pass computation) and resource constraints.

STEP11. Adjust the durations of the active tasks and the lead time.

STEP12. Make a transition to the next state $q+1$ or finish the backward computation of the simulation run if satisfying the termination condition.

<RC slack computation>

STEP13. Compute RC slack for each task.

STEP14. Identify critical sequence(s) in the simulation run following the tasks having zero RC slack.

FIGURE 3-10. ALGORITHM FOR COMPUTING LEAD TIME IN THE EXTENDED MODEL

3.4.2 Computing Resource-Constrained Slack

Resource allocations made in the forward pass computation must be respected in the backward pass computation in order to compute the RC slack defined earlier. Woodworth and Shanahan (1988) used a resource sequence label to record allocation decision of resources to each task, and Bowers (1995) used a method to create explicit resource links in the forward pass computation by identifying all resource flows. The latter method has an advantage in that there is no need to account for resources during the backward computation. However, the resource link should be created whenever a resource allocation is made and this becomes complicated when tasks share resources having multiple units in a complex project (or multi-projects). Thus, the author proposes the new approach that uses a less number of explicit resource links and records of allocation decisions.

During the forward pass computation at STEP4 in Figure 3-10, the model creates explicit resource links from the active tasks which are supposed to be completed in the current state to the tasks which have all inputs but are delayed because of lower priorities for required resources. During the backward computation, the model regards these resource links as additional precedence constraints that determine the starting time of tasks. When resource over-allocation is found during the backward pass, the tasks with later start time during the forward pass have priorities following the allocation decisions made during the forward pass. In this way, only a small number of additional constraints are created without tracking all the resource flows across the tasks. However, it is also necessary to account for resources during the backward computation. The section *A4* in the *Appendix* discusses this method more in detail with comparison to the method proposed by Bowers.

The RC slack of each task is computed by subtracting earliest start time in the forward pass from latest start time computed in the backward pass. Then, one or more critical sequences are identified by following the tasks with zero RC slack. This process is repeated in each simulation run to assess the variance of RC slack and to identify critical sequences that may be changed in each simulation run due to the variance of task durations.

3.4.3 Measures and Rules for Resource Priorities

The model takes the resource requirements for block tasks, assuming that those resources are required over the entire period that all constituent tasks are processed. These requirements may give constraints to other tasks that can be executed in parallel with those block tasks.

Two measures for task priorities proposed by Cooper (1976) are used to determine resource priorities among tasks in the network without sequential iteration. One is a *conventional slack per successor* and the other is a *cumulative resource equivalent duration* (see the section *A3* of the *Appendix* for the definitions). Bowers (1995) suggested that the conventional slack might be better used in loosely resource-constrained networks while the resource equivalent duration might be better in tightly resource-constrained networks. Thus, either one can be used as a more important measure depending on a specific project.

These measures for task priorities are calculated with *most likely durations* of tasks during the forward and backward pass computation *without* resource constraints in the STEP3 of Figure 3-9. And they are used when computing the lead time *with* resource constraints in N scenarios in the STEP4. Thus, the model assumes that resources are allocated based on pre-determined measures without knowing the actual durations of tasks during the simulation (Bowers 1995).

When two or more tasks are competing for limited resources in a certain state, the extended model determines priorities by the heuristic rules whereby a task has a higher priority if:

(1) it has been in process
(2) it has a higher user-specified priority
(3) it has higher cumulative RED
(4) it has a smaller conventional slack per successor
(5) it is sequenced more upstream

(from (1) to (5) in order of significance; (3) and (4) are interchangeable)

The model simulates that different allocation decisions (represented by explicit resource links) can be made in each simulation run while the same pre-determined measures for task priorities are used. This is based on the assumption that a project manager can make resource-allocation decisions with the knowledge of *past* performance (*i.e.* actual durations of tasks prior to the current state) and the priority measures that are calculated with the best *estimates* for the expected durations of tasks (*i.e.* most likely duration estimates). Note that the priority measures are based on the characteristics of a task itself and successive tasks that have *not* been performed at the decision points. Bowers (1995) did his analyses under two different assumptions that a project manager has perfect knowledge of past and future performance or that all resource allocation decisions are made well in advance without changes. The assumption in the thesis is between these two extremes and represents more practical view as Bowers discussed in his work.

3.5 Chapter Summary

In this chapter, the process model is presented using advanced simulation. The model computes the probability distribution of lead time in a resource-constrained project network where iterations take place among sequential, parallel and overlapped tasks. The extended model computes resource-constrained slack and identifies critical sequences in a generalized project network having iteration. These models are used as the modeling module in the integrated project management framework presented in the next chapter.

CHAPTER 4

An Integrated Project Management Framework

4.1 Introduction

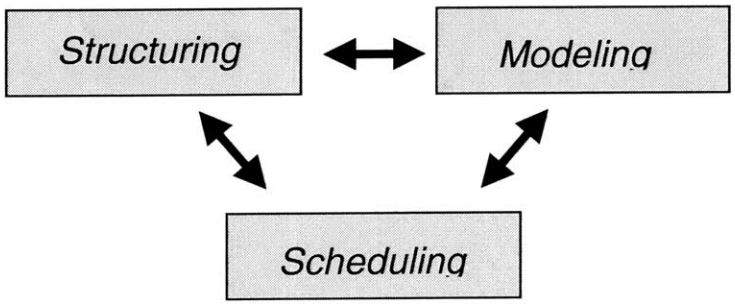


FIGURE 4-1. AN INTEGRATED PROJECT MANAGEMENT FRAMEWORK

Figure 4.1 illustrates the integrated project management framework that this thesis proposes. In the previous chapters, the methods for structuring and modeling modules were presented. This chapter discusses how each module functions and interacts with other modules within the integrated framework. And it presents some applications of the framework in project planning and control.

4.2 Applications of the Integrated Framework

4.2.1 Project Planning

Under the integrated framework, the basic sequence of planning processes is 'structuring – modeling – scheduling'. Figure 4.2 illustrates iterative information flows between the modules in project planning.

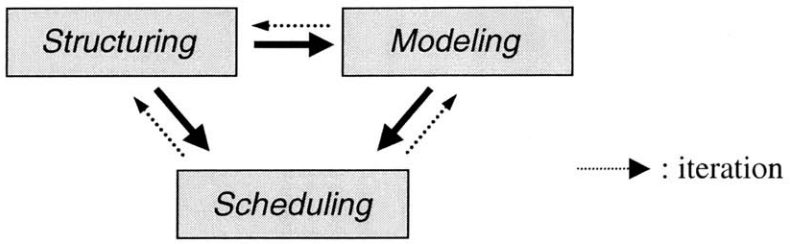


FIGURE 4-2. THE INTEGRATED FRAMEWORK IN PROJECT PLANNING

The structuring module analyzes complex information flows among tasks and decomposes the project into process hierarchies to which each task belongs. The DSM provides a compact visualization of information flows and helps define clear interfaces between tasks. Tasks are sequenced to have minimum feedback iterations from a structural view. The module also computes level slack of tasks and identifies the longest path through hierarchies. Non-binding dependencies are identified and eliminated for analyses in modeling and scheduling modules. Figure 4.3 illustrates the function of the structuring module from the systems view in which a complex project is seen as the system within which component tasks interact through information exchanges.

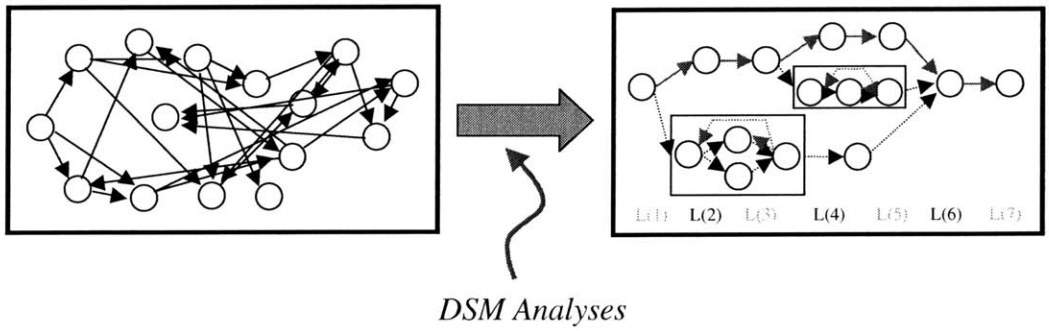


FIGURE 4-3. FUNCTION OF THE STRUCTURING MODULE FROM SYSTEMS VIEW

In the modeling module, dynamic iterative processes are simulated along the time line. With the model inputs for characteristics of tasks, iterations etc., it simulates execution of tasks and decision-making of project members in resource allocations, rework etc. Different execution strategies are evaluated by adjusting modeling parameters. The simulation outcomes include the probability distribution of lead time, criticalities of tasks and major critical paths (or sequences) with percentages of criticalities. Figure 4.4 illustrates the function of the structuring module from the systems view.

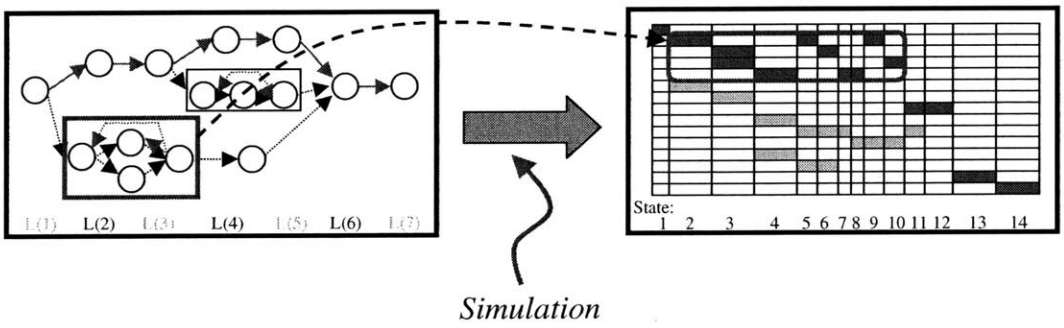
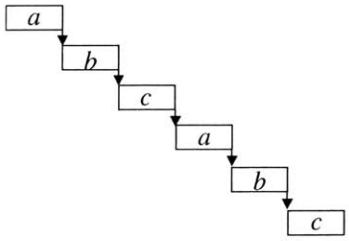


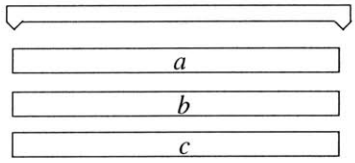
FIGURE 4-4. FUNCTION OF THE MODELING MODULE FROM SYSTEMS VIEW

In the scheduling module, the outcomes of structuring and modeling modules are used to construct a network-based schedule in the form of a PERT or Gantt chart with scheduled task durations. The tasks having slack may be scheduled to have some lag to the extent that they do not add more risk than the expected benefits from the late starts. Figure 4-5 illustrates different ways of representing coupled blocks in the Gantt chart. (a) is one example of a traditional way of representation where *all* tasks constituting a coupled block (loop) are listed k times where k is the expected number of iterations from experience. (b) is the way that Austin et al. (1999) represented a coupled block when transferring information in the DSM to the Gantt chart. In (b), all constituent tasks of a coupled block are listed in parallel without precedence relationships and have the same duration that is estimated directly for the coupled block. This method might be good when the tasks are tightly coupled so that the duration of the block cannot be predicted

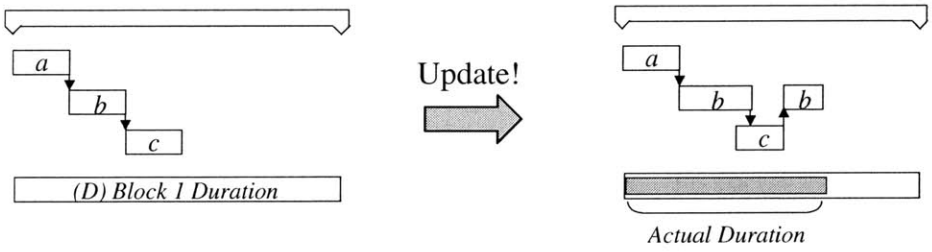
with accuracy by modeling information exchanges among tasks or when the tasks are loosely coupled so that they can work in parallel. Except these two extreme cases, the sequential progress of tasks with moderate overlaps can be simulated in the modeling module. In this case, a coupled block is represented as a 'rolled-up' task within which its constituent tasks are arranged without feedback information flows, as illustrated in (c). A dummy task is added at the end representing the duration of a coupled block that is computed in the modeling module. As uncertainties diminish while tasks progress, iterative execution of tasks can be updated under the rolled-up task. Note that the sequence of tasks in iteration may be changed due to the probabilistic nature of iteration.



(a) Using the Expected Number of Iteration



(b) Long Parallel Tasks without Relationships



(c) Using a Dummy Task Under a Rolled-Up Task

FIGURE 4-5. REPRESENTATION OF A COUPLED BLOCK

The deterministic durations of tasks (including rolled-up tasks) in a network schedule may be chosen from a certain percentile in the probability distribution, depending on a specific project. For instance, if the schedule target is given as 100 days from a start and the lead time is expected to be 110 days with 50th percentile durations of tasks, the

percentile might be adjusted lower to meet the target in schedule, but with a higher schedule risk. A project buffer may be added at the end representing aggregated safety. This can improve the Critical Chain method (Goldratt 1997) as well.

Table 4-1 summarizes the inputs and outputs of each module in project planning.

	Inputs	Outputs
Structuring	• A list of tasks • Information flows among tasks • Information flow patterns	• Design Structure Matrix (Identification of loops and process hierarchies among tasks) • Differentiation of planned & unplanned iterations • Identification of non-binding dependencies • A critical dependency sequence and level slack of tasks
Modeling	• Duration estimates of tasks • Resource requirements of tasks and capacity of the project • Dynamic characteristics of overlapping and sequential iterations (including the learning curve) • Rework risk tolerance	• Probability distributions of the durations of the project and coupled blocks • Resource-constrained (or conventional) slack and criticalities of tasks • Critical sequences (or paths) with percentages of criticalities • Simulated Gantt charts
Scheduling	• Scheduled durations of tasks and coupled blocks chosen from the probability distribution • Due date and/or project buffer size	• PERT or Gantt chart • Schedule risk that the project fails to meet the due date (or target)

TABLE 4-1. INPUTS AND OUTPUTS OF THE MODULES

4.2.2 Project Monitoring and Control

Both the DSM and the Gantt chart can be used to monitor and control the project. The DSM can accelerate communications between people by increasing the understanding of interactions. Thus, the right information can be made available at the right place at the right time (Browning 1999). In addition, the impact of any delay of a single task can be easily identified by tracing information flows from the task. The Gantt chart is used to track the progresses of tasks against a time scale. The tasks with high criticalities and/or small slack need particular attention to reduce chances that the project is overrun.

Using the modeling module, the schedule risk can be identified quantitatively and its assessment can accelerate proactive risk management efforts. The model inputs can be easily updated to incorporate current information as the project progresses. For instance, initial estimates for task duration, overlap amounts and impacts can be updated or replaced with actual values as they become available. Rework parameters may be updated to represent foreseeable iterations. As uncertainties diminish while the project advances, the variance of lead time becomes smaller.

Note that all the benefits of each module can be achieved by maintaining compatible interfaces between the modules. By following the streamlined processes within the framework, any control efforts incorporating latest decisions can be easily represented in the project plan. For instance, if there is a need to add a new task, the project plan may be updated as follows: (1) identify the tasks which it requires inputs from and delivers outcomes to – update the DSM, (2) enter its rework characteristics as well as its duration estimates and resource requirements – update the process model, and (3) update the Gantt chart using the results of analyses from the structuring and modeling modules.

4.3 Other Applications

The previous DSM-based process models (Smith and Eppinger (1997a and 1997b), Eppinger et al. (1997), Carrascosa et al. (1998), Browning and Eppinger (2000)) have provided practical insights for process understanding and improvements. This section discusses applications of the simulation-based model (which is used in the modeling module of the framework) that further enhance analyses that have been performed by the previous models. The distinctive feature of this model is that it has greater flexibility and generality to describe real development processes. Engineering management can benefit from this rich model through better project planning and control in several ways discussed in this section.

4.3.1 Finding an Optimal Task Ordering

The proposed model is a performance evaluation model, not an optimization model. However, the model can be easily utilized to find an optimal task sequence given the objective function that might be, for instance, a function of the average and standard deviation of lead time, and schedule risk. Note that this is an $O(n!)$ operation where n is the number of tasks (Browning and Eppinger 2000) if we allow tasks to be positioned anywhere in the sequence. By fixing some logical precedence relationships between tasks, computation time can be reduced.

4.3.2 Setting Appropriate Due Date

By analyzing the probability distribution of lead time, an appropriate due date can be chosen with predictable confidence (Browning and Eppinger 2000). When the schedule target is given, the risk that the project fails to meet the target can be assessed via the simulation. When the schedule risk is high, the model can be used to evaluate different improvement efforts such as adding resources, overlapping tasks, executing fewer or faster iterations etc. This perspective may facilitate decision making and communications between senior management and the project team.

4.3.3 Finding Areas for Process Improvement

The model can be used to evaluate various process improvements through simulation by adjusting model parameters. Below are examples of model applications for this purpose:

- ***Task criticality:*** When the size of a coupled block is large, it is very likely that it is on the critical path (or critical sequence). For the tasks within coupled blocks, three alternative measures may be used to represent relative importance of tasks within a coupled block where conventional and resource-constrained slacks cannot be defined. The first measure is the sensitivity of project lead time to variation in task duration (Williams (1992b), Bowers (1995), Christian (1995)). The second measure is the rework-adjusted rank positional weight defined earlier. The third measure is *rework-adjusted slack* as a substitute for conventional or RC slack that is computed by replacing nominal durations with rework-adjusted durations of tasks. When the size of a coupled block is small, its aggregate slack provides a broader measure of criticality in the entire network.
- ***Strategic work policy for concurrency:*** The rework risk tolerance defined earlier can be used as guidance for work policy during iterations. Lead time can be reduced by increasing concurrency level strategically, although this may result in increased development costs.
- ***Faster and/or fewer iterations:*** Smith and Eppinger (1997b) proposed two general strategies for accelerating the iterative processes. Faster iterations can be achieved by increasing learning curve and/or decreasing rework impact, *i.e.* reducing the amount of rework. For instance, the efficient use of information technology such as computer-aided tools could help accomplish this effect. Fewer iterations can be achieved by decreasing rework probabilities. Well-defined interfaces between tasks, and well-established coordination and communication routes between project members could reduce the number of iterations. The proposed model can be used to identify the most efficient points for such improvements.

4.3.4 Evaluating Multiple Projects

The model can also be applied in a multi-project environment by representing each project on a different path. In this manner, it is possible to consider resource allocation across the projects and the effect of such constraints on completion of all of the projects.

4.4 Chapter Summary

This chapter presented the integrated project management framework for complex engineering projects. The integrated method streamlines project planning and control using three modules that are structuring, modeling, and scheduling. Under this integrated framework, the positive aspects of the methods used in each module can be utilized while overcoming the limitations of stand-alone applications.

CHAPTER 5

Case Studies

5.1 Introduction

In this chapter, two industrial cases are presented. In the first case, the DSM-based process model (which is used in the modeling module in the integrated framework) is used to analyze the uninhabited aerial vehicle (UAV) preliminary design process at an aerospace company using the data from Browning (1999). In the second case, the streamlined project planning is demonstrated in a logistics project at a medical equipment company.

5.2 UAV Development Project[1]

The Boeing IS&DS Group was testing some novel design process techniques in an effort to develop complex system products faster and cheaper while maintaining high performance. One such effort was the UAV project, which utilized some new methods for organizing, managing, and implementing the vehicle's conceptual and preliminary design development phases. After the initial model and effort led to a faster and more efficient process, Boeing enlisted the aid of an outside perspective to learn still more. The data and analyses presented in Browning (1999) was this second phase effort. In this thesis, the data and analyses are extended to include additional parameters that can be incorporated in the proposed model.

[1] The background of the case was quoted with some modifications from Browning (1999).

5.2.1 Basic Inputs and Analyses Results

Figure 5-1 shows basic model inputs. Under the same conditions of Browning and Eppinger (2000) – (1) 0 and 100th percentiles for optimistic and pessimistic duration estimates (2) constant rework probabilities in all iterations, (3) learning curve benefit only in the first iteration, (4) zero rework risk tolerance $P_{tolerance}$, (5) no overlapping, and (6) no resource conflicts, the average of lead time is 146.8 days with 17.0 days of standard deviation after 2000 simulation runs. The probability distribution shown in Figure 5-2 is skewed to the right because the lead time becomes longer as more iterations take place. Both the average and standard deviation are higher than those obtained by Browning and Eppinger (avg. 141, s.d. 8). This is mainly because the new model accounts for all the successive feedforward rework while the earlier model does not. (See the section *A2-2* in the *Appendix* for the detailed explanation).

Task Name	ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Exp. Duration Opt.	Likely	Pess.	Learning
Prepare UAV Preliminary DR&O	1															1.9	2.0	3.0	0.35
Create UAV Preliminary Design Configuration	2	1								1						4.8	5.0	8.8	0.20
Prepare Surfaced Models & Internal Drawings	3		1		1											2.7	2.8	4.2	0.60
Perform Aerodynamics Analyses & Evaluation	4	1		1												9.0	10.0	12.5	0.33
Create Initial Structural Geometry	5	1		1			1		1				2	1		14.3	15.0	26.3	0.40
Prepare Structural Geometry & Notes for FEM	6	1				2										9.0	10.0	11.0	0.90
Develop Structural Design Conditions	7	1					2									7.2	8.0	10.0	0.35
Perform Weights & Intertias Analyses	8						1						1			4.8	5.0	8.8	1.00
Perform S&C Analyses & Evaluation	9	1		1					1							18.0	20.0	22.0	0.25
Develop Freebody Diagrams & Applied Loads	10				1		1	1	1			1				9.5	10.0	17.5	0.50
Establish Internal Load Distributions	11						1	1	1		2					14.3	15.0	26.3	0.75
Evaluate Structural Strength, Stiffness, & Life	12	1					1	1			2	2				13.5	15.0	18.8	0.30
Preliminary Manufacturing Planning & Analyses	13	1				1							1			30.0	32.5	36.0	0.28
Prepare UAV Proposal	14	1	1	1	1	1	1	1	1	1	1	1	1	1		4.5	5.0	6.3	0.70

(a) Project Table and DSM

	2	3	4	5	6	7	8	9	10	11	12	13
2								0.2				
3	0.5		0.4									
4		0.5										
5		0.5			0.1		0.1				0.3	0.1
6				0.4								
7					0.4							
8					0.5						0.5	
9		0.5	0.5				0.5					
10			0.1		0.5	0.2	0.1			0.4		
11					0.5	0.5	0.5		0.5			
12					0.4	0.5			0.5	0.4		
13				0.5							0.4	

(b) Rework Probability Matrix

	2	3	4	5	6	7	8	9	10	11	12	13
2								0.1				
3	0.3		0.5									
4		0.8										
5		0.1			0.1		0.1				0.3	0.1
6				0.3								
7					0.8							
8					0.5						0.5	
9		0.3	0.3				0.3					
10			0.1		0.5	0.4	0.3			0.3		
11					0.5	0.5	0.3		0.3			
12					0.3	0.5			0.5	0.5		
13				0.9							0.3	

(c) Rework Impact Matrix

FIGURE 5-1. MODEL INPUTS FOR UAV PROJECT

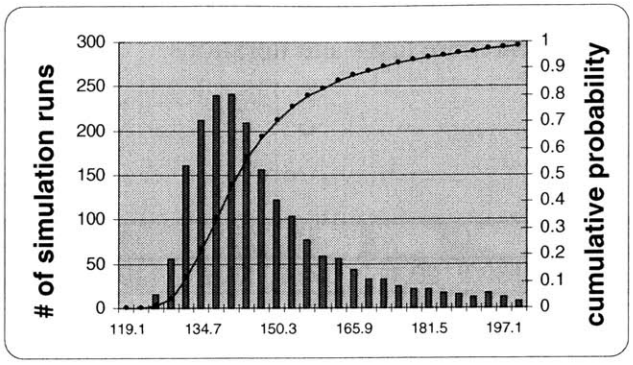


FIGURE 5-2. PROBABILITY DISTRIBUTION OF LEAD TIME WITH BASIC INPUTS

5.2.2 Analyses Results Using Additional Modeling Parameters

The simulation model allows great flexibility to account for more general features of dynamic development processes. Table 5-1 summarizes the results using additional modeling parameters as follows:

(1) Optimistic and pessimistic duration estimates are 10th and 90th percentiles, respectively.
(2) Rework probabilities decrease in each iteration by 50%.
(3) Maximum learning curve is 50% of that in the first iteration.
(4) Rework risk tolerance $P_{tolerance}$ = 0.3, i.e. more concurrency is allowed when tasks iterate.

Under the more rigorous assumption about task duration estimates in (1), both the average and standard deviation are greater than those with 0 and 100th percentile estimates. This is mainly due to the right-skewness tendency of task durations. Also, the existence of multiple paths in the project contributes to the increase of the lead time. Incorporating dynamic characteristics of sequential iteration in (2) and (3), and increased task concurrency in (4), the model predicts smaller averages and standard deviations of the lead time. The cumulative effect of the additional modeling parameters from (2) to (4)

is a 5.3% decrease from the lead time under the assumption (1). Note that this difference will be more significant with more tasks and iterations.

Basic	(1)	(1), (2)	(1) - (3)	(1) - (4)
(146.8, 17.0)	(152.2, 20.1)	(149.3, 15.8)	(148.5, 14.2)	(144.2, 13.1)

<Note>

(i) (#, #): (average, standard deviation)

(ii) "(1)-(3)" denotes the results of the model with assumptions (1) through (3), and so on.

TABLE 5-1. RESULTS USING ADDITIONAL MODELING PARAMETERS

Ignoring feedback marks in the DSM, *i.e.* assuming no sequential iterations, the path along the tasks 1-2-3-5-6-7-10-11-12-13-14 is a critical path when tasks have most likely durations. This implies that the lead time can be reduced by overlapping the tasks along this path such as the development of preliminary surfaced configuration (task 3) and structural design (tasks 5–7). Other important leverage points for reducing the lead time are the feedback marks. By transferring preliminary review decisions or testing results to upstream tasks, feedback rework can start earlier. This allows for the accelerations of iterative rework. Under the following overlapping scenario, the average lead time is reduced to 137.7 days:

(5) For the six information flows marked by "2" in the DSM in Figure 9 (b), tasks are planned to complete 25% of work with preliminary inputs before receiving final updates, and expected to redo 50% of work completed without final updates.

The above scenario includes the overlapping between tasks 12 and 5 in feedback rework. The following scenario, for example, would cause such overlapping:

As a result of structural evaluation in task 12, it may become necessary to redesign the structural geometry of specific subsystems before having evaluation results for the entire system. The need to redesign some subsystems (task 5) can be detected early in

a series of tests in task 12, whereas the need for additional weight and inertial analyses (task 8) can be determined only at the end of the tests.

All the above analyses are performed under the assumption that there is no resource conflict among the tasks in the UAV project. This assumption is reasonable because tasks that can be performed in parallel are related to different disciplines, therefore, no resource sharing is necessary between those tasks. In multiple project environments, however, the resources belonging to the same functional group may need to get involved in tasks in different projects during a certain period of time. In this case, optimal resource allocation toward project objectives becomes an important issue. For the purpose of illustration, an example situation is tested as follows:

(6) Tasks 7 and 8 compete for limited resources and one of them should be delayed.

Table 5-2 shows the rework-adjusted durations and rank positional weights of the tasks under assumptions (1)-(5). Since the rework-adjusted RPW of task 8 is higher than that of task 7, the model assigns resources to task 8 and delays task 7 that can work in parallel with task 8 without this resource constraint. Then, both tasks 7 and 8 become critical and the project is delayed by the duration of task 7. Figure 5-3 shows an example of a simulated Gantt chart. This is only one of many simulation runs under assumptions (1)-(6). The numbers inside the bars indicate the amounts of overlapped work performed in prior state(s). Even though the scenario shows that tasks rework during states 5 and 14-19, it delays the entire project only in states 5 and 14 since no rework is added to task 13 after starting earlier. This is due to the pre-determined work policy for increased task concurrency during iterations. Since total rework probabilities are less than 0.3 except for task 5 as shown in Figure 5-4, preliminary manufacturing planning and analyses (task 13) are simulated to work concurrently with functional performance analyses (tasks 8-11) after redesigning structural geometry (task 5).

ID	Task Name	Rework-Adjusted Duration	Rework-Adjusted RPW
2	Create UAV Preliminary Design Confinguration	6.4	184.4
3	Prepare Surfaced Models & Internal Drawings	3.3	178.0
4	Perform Aerodynamics Analyses & Evaluation	11.7	108.7
5	Create Initial Structural Geometry	19.6	163.0
6	Prepare Structural Geometry & Notes for FEM	10.5	143.4
7	Develop Structural Design Conditions	8.4	105.4
8	Perform Weights & Intertias Analyses	7.3	124.5
9	Perform S&C Analyses & Evaluation	20.2	20.2
10	Develop Freebody Diagrams & Applied Loads	13.0	97.0
11	Establish Internal Load Distributions	22.1	84.0
12	Evaluate Structural Strength, Stiffness, & Life	27.8	61.9
13	Preliminary Manufacturing Planning & Analyses	34.2	34.1

TABLE 5-2. REWORK-ADJUSTED DURATION AND RANK POSITIONAL WEIGHT

ID \ State	D (w/o rework)	D (w. rework)	1	2	3	4	6	7	8	9	10	11	12	13	20	21
1	2.0	2.0														
2	5.9	5.9														
3	2.9	3.8														
4	8.9	8.9														
5	17.0	19.2														
6	9.0	10.1						2.2								
7	10.3	10.3														
8	7.7	11.5														
9	20.2	21.7														
10	16.3	18.7														
11	15.5	23.8										3.9				
12	14.8	16.7												3.7		
13	34.7	34.7														
14	5.0	5.0														

FIGURE 5-3. AN EXAMPLE OF A SIMULATED GANTT CHART

ID	2	3	4	5	6	7	8	9	10	11	12	13
2												
3	0.50											
4	0.25	0.50										
5	0.25	0.50	0.00									
6	0.10	0.20	0.00	0.40								
7	0.04	0.08	0.00	0.16	0.40							
8	0.05	0.10	0.00	0.20	0.50	0.00						
9	0.40	0.80	0.50	0.10	0.25	0.00	0.50					
10	0.09	0.18	0.10	0.25	0.63	0.20	0.10	0.00				
11	0.14	0.28	0.05	0.51	1.26	0.60	0.55	0.00	0.50			
12	0.16	0.32	0.07	0.57	1.42	0.84	0.27	0.00	0.70	0.40		
13	0.19	0.38	0.03	0.73	0.57	0.34	0.11	0.00	0.28	0.16	0.40	

<Note> Rework Concurrency is changed when feedback iteration happens.

FIGURE 5-4. AN EXAMPLE OF REWORK CONCURRENCY

5.3 Logistics Europe Project

The integrated method was used in planning a logistics project at a medical equipment company. The project was in the middle of planning processes when the case study was begun. The purpose of this case study is to demonstrate that the proposed methodology is applicable to real projects and improves project management practices.

5.3.1 Backgrounds

The medical company designs, develops, manufactures and markets a sample preparation system for medical diagnostic applications. The business sector grows at a rate of 8-10% per year, which requires the corporation to adjust its structure and procedures on a frequent basis. This includes the customer-driven supply chain management system that ensures the seamless flow of material and information from manufacturing forecasts to European end users. The mission of Logistics Europe is to support the European operations by providing an optimal service level through the entire supply/value chain and reducing cycle time for materials and information. To achieve this, a European distribution center (EU DC) was established and all local warehouses were consolidated into that platform except for the Italian warehouse (IW). This project is for consolidating the IW to the EU DC. The company can benefit from cost savings in inventory and logistics through the successful completion of this project.

5.3.2 Application of the Integrated Method to Project Planning

The data were collected in three steps: (1) a list of tasks, (2) information flows among tasks, and (3) characteristics of tasks and iterations. Between the steps, the author had the face-to-face discussion with the project manger who is actually responsible for maintaining the plan. The first two sets of data were analyzed in the structuring module using the DSM. The third set of data was analyzed in the modeling module. Based on the results of analyses, a network-based schedule was built with deterministic durations of tasks and coupled blocks.

The data for a list of tasks were gathered from the initial plan built using the PERT/CPM method. However, they need to be modified in order to employ the information-based approach of the proposed method in which a task is an information-processing unit. (Before modification, several tasks represented instantaneous actions similar to milestones rather than information-processing efforts.) After determining the list of tasks, information flows among the tasks were mapped in a square matrix following the two-type scheme of transfer patterns. Figure 5-5 illustrates the results of hierarchical decomposition in the DSM based on the as-early-as-possible rule. Figure 5-6 shows the collapsed DSM where various dependencies are classified.

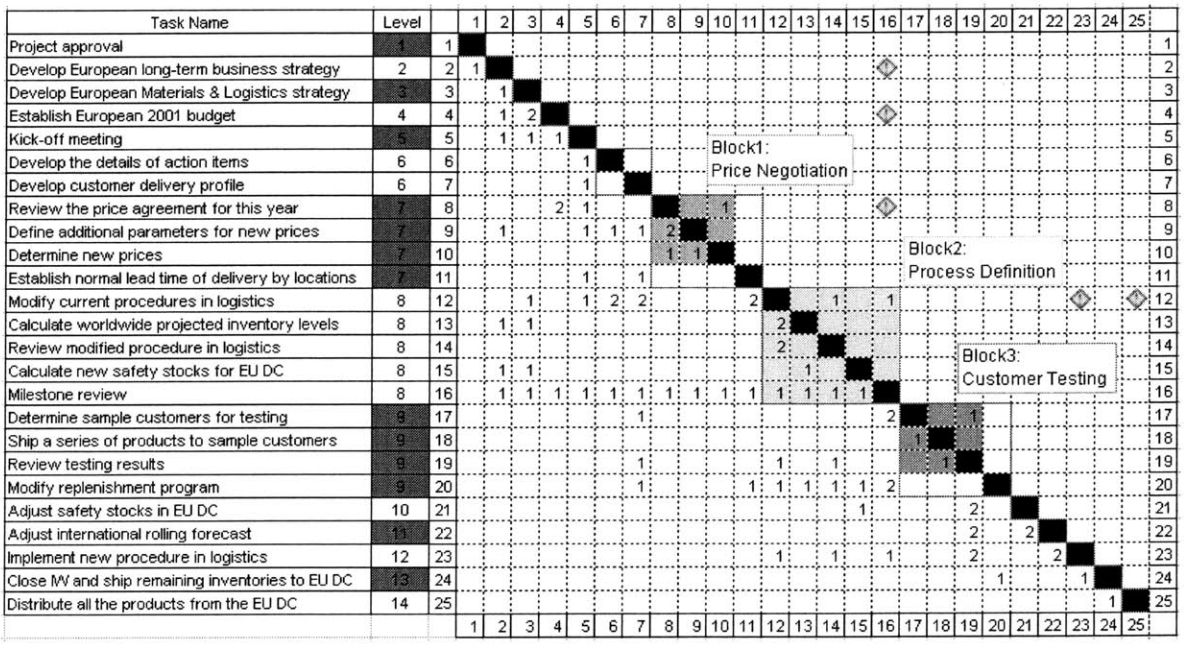


FIGURE 5-5. DESIGN STRUCTURE MATRIX (AS-EARLY-AS-POSSIBLE)

Task Name	Level		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
Project approval	1	1																		1
Develop European long-term business strategy	2	2	1																	2
Develop European Materials & Logistics strategy	3	3		1																3
Establish European 2001 budget	4	4		1	2															4
Kick-off meeting	5	5		1	1	1														5
Develop the details of action items	6	6					1													6
Develop customer delivery profile	6	7					1													7
Block1: Price Negotiation	7	8		1		1	1	1	1											8
Establish normal lead time of delivery by locations	7	9					1		1											9
Block2: Process Definition	8	10		1	1	1	1	1	1	2	2									10
Block3: Customer Testing	9	11							1			2								11
Modify replenishment program	9	12							1		1	2								12
Adjust safety stocks in EU DC	10	13										1	2							13
Adjust international rolling forecast	11	14											1							14
Implement new procedure in logistics	12	15										1	1							15
Close IW and ship remaining inventories to EU DC	13	16												1						16
Distribute all the products from the EU DC	14	17																	1	17
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	

<Note> **1** and **2** marks represent binding dependencies.

FIGURE 5-6. COLLAPSED DSM (AS-EARLY-AS-POSSIBLE)

The following observations from the comparison of the PERT/CPM approach and the information-based approach in this case study may explain some general improvements of the proposed method:

- In the initial plan, the tasks were arranged based on precedence *relationships* among tasks which are considered for scheduling the expected starting and ending dates of tasks (not for defining precedence *constraints*). Thus, precedence relationships of each task are accounted with only its neighboring tasks in sequence. In the information-based approach of the DSM method, information flows are thoroughly captured and precedence *constraints* are derived from those information flows. After analyzing the DSM, some additional constraints not documented in the initial plan were found as binding dependencies in the DSM.
- The initial plan did not explicitly show iteration loops while it implied the existence of planned iteration in customer testing by scheduling repetition of three tasks. From the structuring analyses, we found that there are three coupled blocks for *planned* iteration which represent price negotiation, process definition, and

customer testing. In addition, the project also has some chances to do *unplanned* iteration when it completely fails to meet the criteria. Being aware of such iterations was very helpful for the project manger to understand the potential sources of risk. This may lead to proactive risk management as the project progresses.

After structuring processes, the detailed data for durations, resources, overlapping and sequential iteration were collected. (See the section A5 in the Appendix for the data.) Using those data, simulations were performed to predict the lead time under the following assumptions: (1) 10th and 90th percentiles for optimistic and pessimistic duration estimates, (2) $P_{tolerance} = 0$, and (3) sequential iterations only among the tasks within the coupled blocks. Table 5-3 summarizes the results of simulations. Table 5-4 shows major critical paths and sequences with the percentages that they are critical. Figure 5-7 shows the probability distribution of the lead time with resource constraints. Note that the distribution would show more right-skewness if unplanned iterations were accounted in the simulation.

	avg.	s.d.	(10%,30%,50%,70%,90%)
Lead Time (w/o RCs)	157.4	19.9	(132.8, 146.7, 157.3, 168, 182.2)
Lead Time (w. RCs)	157.6	19.9	(132.9, 146.8, 157.6, 168.1, 182.2)
BLOCK1 : Price Negotiation	18.0	6.1	(10.6, 14.4, 17.3, 20.6, 26.7)
BLOCK2 : Process Definition	32.6	12.1	(18.0, 25.5, 31.6, 38.0, 48.8)
BLOCK3 : Customer Testing	27.9	7.9	(17.4, 23.1, 27.6, 32.0, 38.6)

<Note> "RCs" denotes resource constraints.

TABLE 5-3. SUMMARY OF SIMULATION RESULTS

Major Critical Paths (without Resource Constraints)
1-2-3-4-5-6-8-14-21-27-28-29-30-31 [63%]
1-2-3-4-5-7-8-14-21-27-28-29-30-31 [28%]
1-2-3-4-5-7-13-14-21-27-28-29-30-31 [6%]
Major Critical Sequences (with Leveled Resources)
1-2-3-4-5-6-8-14-21-27-28-29-30-31 [57%]
1-2-3-4-5-7-8-14-21-27-28-29-30-31 [28%]
1-2-3-4-5-6-13-14-21-27-28-29-30-31 [8%]

<Note> Task IDs are based on those in Figure 5-8.

TABLE 5-4. CRITICAL PATHS AND SEQUENCES

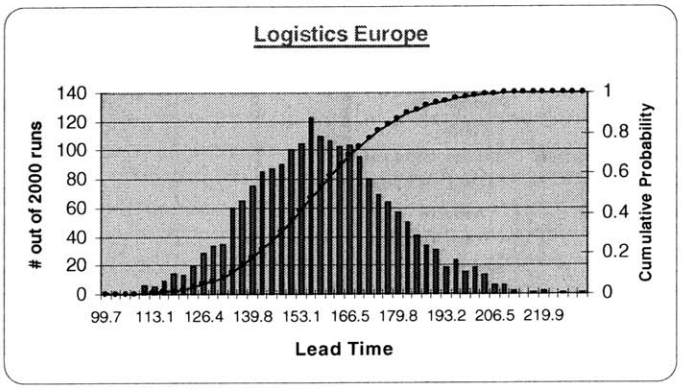


FIGURE 5-7. PROBABILITY DISTRIBUTION OF LOGISTICS EUROPE PROJECT DURATION

This project did not have a due date or a target while the greater cost-savings can be achieved as the project is completed successfully in less time. Thus, the network-based schedule was built with the most-likely durations of tasks and average durations of coupled blocks (without a buffer at the end), as shown in Figure 5-8. Three coupled blocks identified and simulated earlier are shown as rolled-up tasks within which precedence relationships including overlapping are represented.

Compared with the initial plan based on the PERT/CPM method, the revised plan has more structured and rigorous information about the project. Following the proposed approach, the project manger could better understand the structures of the project and

have better prediction about the outcomes of uncertain processes. The DSM as well as the Gantt chart can be effectively used to monitor and control the project as it progresses.

ID	Task Name	Duration	Predecessors
1	Project approval	1 day	
2	Develop European long-term business strategy	10 days	1
3	Develop European Materials & Logistics strategy	10 days	2
4	Establish European 2001 budget	10 days	3
5	Kick-off meeting	1 day	4
6	Develop the details of action items	15 days	5
7	Develop customer delivery profile	10 days	5
8	**Block1: Price Negotiation**	**17.3 days**	**6,7**
9	Review the price agreement for this year	2 days	
10	Define additional parameters for new prices	4.3 days	9FS-1.6 days
11	Determine new prices	10 days	9,10
12	(D) Block1 Duration	17.3 days	
13	Establish normal lead time of delivery by locations	10 days	7
14	**Block2: Process Definition**	**30.4 days**	**8,13**
15	Modify current procedures in logistics	15 days	
16	Calculate worldwide projected inventory levels	3.45 days	15FS-0.9 days
17	Review modified procedure in logistics	6.6 days	15FS-3 days
18	Calculate new safety stocks for EU DC	2 days	16
19	Milestone review	3 days	15,16,17,18
20	(D) Block2 Duration	30.4 days	
21	**Block3: Customer Testing**	**27.6 days**	**14**
22	Determine sample customers for testing	2 days	
23	Ship a series of products to sample customers	12 days	22
24	Review testing results	4 days	23
25	(D) Block3 Duration	27.6 days	
26	Modify replenishment program	4 days	14
27	Adjust safety stocks in EU DC	2 days	21
28	Adjust international rolling forecast	2 days	27
29	Implement new procedure in logistics	5 days	28
30	Close IW and ship remaining inventories to EU DC	16 days	26,29
31	Distribute all the products from the EU DC	3 days	30

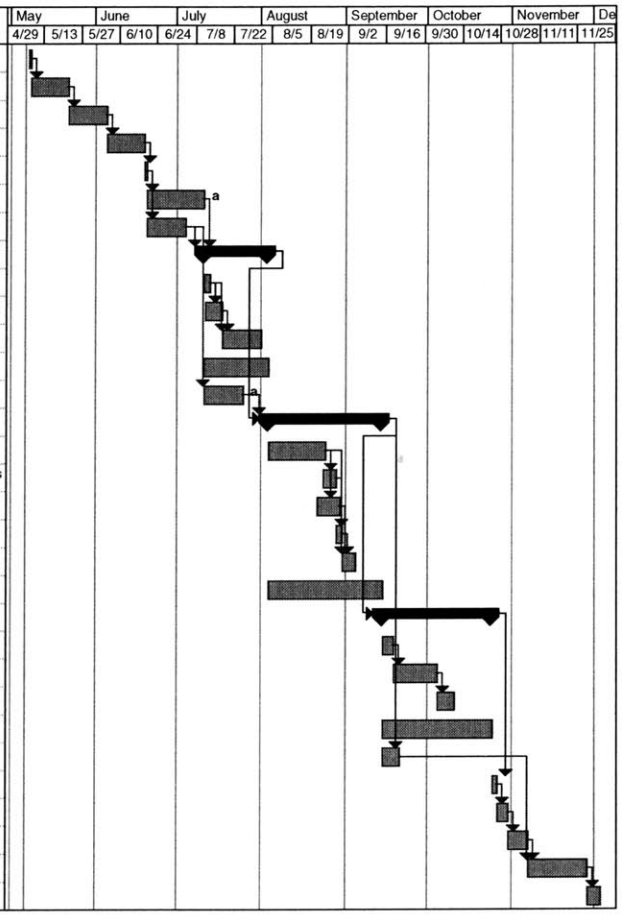


FIGURE 5-8. NETWORK-BASED SCHEDULE

5.4 Chapter Summary

In this chapter, two industrial cases are presented to demonstrate the integrated method. The method was proven to be applicable to real projects even though more extensive applications to projects in various industries need to be followed for improving the method.

CHAPTER 6

Conclusion and Future Research

6.1 Conclusion

This thesis proposes the integrated project management framework for complex engineering projects. It is based on the systems view of a complex project as well as on the information-processing view of a task. Under the integrated framework, analyses are performed to structure information flows, and to model the characteristics of information processing of each task and information exchanges among tasks. The network-based schedule is built upon these analyses.

The framework in part and as a whole provides helpful contributions to project management theory and application. It integrates several broad aspects of theories and applications in the areas such as system and project management, design theory and methodology, systems modeling and simulation, schedule network analyses, stochastic iteration modeling etc.

A key contribution of this work is the development of a generalized process model that allows for streamlining the DSM-based structuring analyses with the network-based scheduling analyses. The model accounts for important characteristics of product development processes, including information transfer patterns, uncertain task durations, resource conflicts, overlapping and sequential iterations, and rework concurrency. The model addresses several limitations imposed by previous analytical and simulation-based approaches. It can be applied to a wide range of processes where iteration takes place among sequential, parallel, and overlapped tasks in a resource-constrained project.

Increased understanding of complex processes can be achieved through structuring and modeling information flows. The integrated method is also useful for evaluating different

project plans and for identifying strategies for process improvements. Proactive risk management can be achieved by assessing the status of the project as it progresses.

6.2 Future Research

6.2.1 Extensive Applications

The integrated method has been used to analyze product development processes and to plan a logistics project. While the feasibility and applicability of the method are proven, more extensive applications in various industries need to be followed to further test and improve the method. In particular, it would be helpful to test the utility of the method in a multi-project environment where an efficient resource-allocation issue is more significant. In addition, the benefits of the integrated framework in project monitoring and control have not been proven yet.

The author observed that a basic information-based approach has been applied to build a template for standardized product development processes based on the PERT/CPM method in one of major U.S. automotive companies. However, lots of important aspects of real development projects could not be incorporated in the standardized template due to the limited capabilities of existing principles and tools. For instance, overlapping and sequential iterations were not documented and modeled in the plan even though they are widely practiced among people. The author believes that the integrated method can provide the methodologies to build a template for complex development projects.

6.2.2 Extensions of the Integrated Method

While the author believes that the integrated method has broad applicability to complex engineering projects, there are still several limitations and possible extensions, as discussed below.

The method does not include any process optimization technique such as finding optimal resource-allocation to minimize the lead time and finding optimal overlap amounts in multiple paths. This is a limitation of a richer process model incorporating iteration and overlapping in multiple paths. Also, the model is not suitable for predicting the progress of concurrent execution of tightly coupled tasks. For this type of parallel iteration, the method suggests that a coupled block is modeled as if it were a single task while representing information flows among tasks in the DSM. More accuracy and generality can be achieved if we could model frequent bi-directional information exchanges and consequent impacts between parallel tasks.

The method simply assumes that the processing time of each task is independent of those of other tasks. However, when uncertainties affect multiple tasks, independent duration distributions cannot be assumed (Williams 1992a). Various methods have been developed to model interdependencies, from estimation of correlation coefficients between tasks to use of joint distributions. However, difficulties remain in measuring correlation among task durations.

The method also assumes a fixed resource pool for the project and constant resource requirements for tasks. It can be extended by allowing variable resource capacity and requirements.

Lastly, the method can be further extended by incorporating development cost as in Browning and Eppinger (2000). However, it is very difficult to apply the same cost structure to different iterative process hierarchies in conjunction with development time.

Reference

[1] Alexander, C., *Note on the Synthesis of Form*, Harvard University Press, 1964.

[2] Adler, P., Mandelbaum, A., Nguyen V. and Schwerer E., "From Project to Process Management: An Empirically-based Framework for Analyzing Product Development Time", *Management Science*, Vol. 41, No. 3, pp. 458-484, 1995.

[3] Ahmadi, R., and Hongbo, W., "Rationalizing Product Design Development Process," Working Paper, Anderson Graduate School of Management, UCLA, 1994.

[4] Ahmadi, R. and Wang, R., "Managing Development Risk in Product Design Processes," *Operations Research*, Vol. 47, No. 2, pp.235-246, 1999.

[5] AitSahlia, F., Johnson, E. and Will, P., "Is Concurrent Engineering Always a Sensible Proposition?," *IEEE Transactions on Engineering Management*, Vol. 42, No. 2, pp. 166-170,1995.

[6] Andersson, J., Pohl, J. and Eppinger, S., "A Design Process Modeling Approach Incorporating Nonlinear Elements," *Proceedings of ASME Design Engineering Technical Conferences*, DETC98-5663, 1998.

[7] Austin, S., Baldwin, A., Li, B., and Waskett, P., "Analytical Design Planning Technique: A Model of the Detailed Building Design Process," *Design Studies*, Vol. 20, pp.279-296, 1999.

[8] Bowers, J., "Criticality in Resource Constrained Networks," *Journal of the Operational Research Society*, Vol. 46, pp 80-91, 1995.

[9] Browning, T., "Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development," Ph.D. Thesis (TMP), MIT, Cambridge, MA, 1998.

[10] Browning, T. and Eppinger, S., "Modeling the Impact of Process Architecture on Cost and Schedule Risk in Product Development," Working Paper, No. 4050, Sloan School of Management, MIT, 2000.

[11] Carrascosa, M., Eppinger S. and Whitney D., "Using the Design Structure Matrix to Estimate Product Development Time," *Proceedings of ASME Design Engineering Technical Conference*, DETC98/DAC-6013, 1998.

[12] Christian, A., "Simulation of Information Flow in Design," Ph.D. Thesis (ME), MIT, Cambridge, MA, 1995.

[13] Christofides, N., Alvarez-Valdes, R. and Tamarit, J., "Project Scheduling with Resource Constraints: A Branch and Bound Approach," *European Journal of Operational Research*, Vol. 29, pp. 262-273, 1987.

[14] Cooper, D., "Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation," *Management Science*, Vol. 22, No. 11, pp. 1186-1194, 1976.

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

[16] David, E. and Heidorn, G., "Optimal Project Scheduling under Multiple Resource Constraints," *Management Science*, Vol. 17, No. 12, pp 803-816, 1971.

[17] Eppinger, S., Whitney, D., Smith, R. and Gebala, D., "A Model-Based Method for Organizing Tasks in Product Development," *Research in Engineering Design*, Vol. 6, pp. 1-13, 1994.

[18] Eppinger, S., Nukala, M. and Whitney, D., "Generalized Models of Design Iteration Using Signal Flow Graphs," *Research in Engineering Design*, Vol. 9, No. 2, pp. 112-123, 1997.

[19] Goldratt, E., *Critical Chain*, The North River Press, MA, 1997.

[20] Grose, D., "Reengineering the Aircraft Design process," *Proceedings of the Fifth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, Sept. 7-9, 1994.

[21] Ha, A. and Porteus, E., "Optimal Timing of Reviews in the Concurrent Design for Manufacturability," *Management Science*, Vol. 41, No. 9, pp. 1431-1447, 1995.

[22] Hoedemaker, G., Blackburn, J. and Wassenhove, L., "Limits to Concurrency," Working Paper, Vanderbilt University Owen School of Management, 1995.

[23] Joglekar, N., Yassine, A., Eppinger, S., and Whitney, D., "Performance of Coupled Product Development Activities with a Deadline," Working Paper, No. 4122, Sloan School of Management, MIT, 2000.

[24] Keefer, D. and Verdini, W., "Better Estimation of PERT Activity Time Parameters," *Management Science*, Vol. 39, No. 9, pp. 1086-1091, 1993.

[25] Kelley, J. and Walker, M., "Critical-Path Planning and Scheduling," *Proceedings of Easter Joint Computer Conference*, pp. 160-173, 1959.

[26] Kerzner, H., *Project Management: a Systems Approach to Planning, Scheduling, and Controlling*, 5th Edition, VNR, New York, NY, 1995.

[27] Krishnan, A., Eppinger, S. and Whitney, D., "A Model-Based Framework to Overlap Product Development Activities," *Management Science*, Vol. 43, No. 4, pp.437-451, 1997.

[28] Kusiak, A., and Wang, J., "Efficient organizing of design activities," *International Journal of Production Research*, Vol. 31, pp 753-769, 1993.

[29] Ledet, W. and Himmelblau, D., "Decomposition Procedures for the Solving of Large Scale Systems," *Advances in Chemical Engineering*, Vol. 8, pp. 185-224, 1970.

[30] Levitt, R., Cohen, G., Kunz, J., Nass, C., Christiansen, T., and Jin, Y., *The virtual design team: Simulating how organization structures and information processing tools affect team performance*, in Carley, K.M. and M.J. Prietula, editors, Computational Organization Theory, Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

[31] Loch, C. and Terwiesch, C., "Communication and Uncertainty in Concurrent Engineering," *Management Science*, Vol. 44, No. 8, pp. 1032-1048, 1998.

[32] Malcolm, D., Roseboom, J., Clark, C. and Fazar, W., "Application of a Technique for Research and Development Program Evaluation," *Operations Research*, Vol. 7, No. 5, pp. 646-669, 1959.

[33] McKay, M., Beckman, R. and Canover, W., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, pp. 239- 245, 1979.

[34] Neumann, K., "GERT Network and the Time-Oriented Evaluation of Projects," Lecture Notes in Economics and Mathematical Systems, Vol. 172, 1979.

[35] Pascoe, T., "An Experimental Comparison of Heuristic Methods for Allocating Resources," Ph.D. Thesis, Cambridge University, England, 1965.

[36] Patterson, J., "A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem," *Management Science*, Vol. 30, No. 3, pp. 854-867, 1984.

[37] Patterson, J. and Roth, G., "Scheduling a Project under Multiple Resource Constrains: A Zero-One Programming Approach," *AIE Trans.*, Vol. 8, No. 3, pp 449-456, 1976.

[38] Patterson, J., Talbot B., Slowinski, R. and Weglarz, J., "Computational Experience with a Backtracking Algorithm for Solving a General Class of Precedence and Resource-constrained Scheduling Problems," *European Journal of Operational Research*, Vol. 49, pp. 68-79, 1990.

[39] Pritsker, A., Watters, W. and Wolfe, P., "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach," *Management Science*, Vol. 16, pp. 93-108, 1969.

[40] Pritsker, A., "Modeling and Analysis Using Q-GERT Networks, 2nd Edition, John Wiley/Halsted Press, 1979.

[41] Pritsker, A. and O'Reilly, J., *Simulation with Visual SLAM and AweSim*, 2nd Edition, John Wiley & Sons, 1999.

[42] Roemer, T., Ahmadi, R. and Wang, R., "Time-Cost Trade-Offs in Overlapped Product Development," *Operations Research*, Vol. 48, No. 6, pp. 858-865, 2000.

[43] Rogers, J., "A knowledge-based tool for multilevel decomposition of complex design problem," NASA TP 2903, 1989.

[44] Rogers, J., "Tools and Techniques for Decomposing and Managing Complex Design Projects," *AIAA Journal of Aircraft*, Vol. 36, No.1, pp. 266-274, 1999.

[45] Sabbaghian, N., Eppinger, S., and Murman, E., "Product Development Process Capture and Display Using Web-Based Technologies," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, CA, 1998.

[46] Sargent, W. and Westerberg, A., "Speed-Up in Chemical Engineering Design," Trans. Inst. Chem. Eng., Vol. 42, 1964.

[47] Simon, H., "Applying Information Technology to Organization Design," *Public Administration Review*, Vol. 33, pp. 268-278, 1973.

[48] Smith, R. and Eppinger, S., "A Predictive Model of Sequential Iteration in Engineering Design," *Management Science*, Vol. 43, No. 8, pp. 1104-1120, 1997.

[49] Smith, R. and Eppinger, S., " Identifying Controlling Features of Engineering Design Iteration," *Management Science*, Vol. 43, No. 3, pp. 276-293, 1997.

[50] Smith P., and Reinertsen, D., "Developing Products in Half the Time," 2nd Ed. Van Nostrand Reinhold, NY, 1995.

[51] Steward, D., "On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations," *SIAM Rev.*, Vol. 4, pp. 321-342, 1962.

[52] Steward, D., "Partitioning and Tearing Systems of Equations," *SIAM Numerical Anal.*, ser. B, Vol. 2, No. 2, pp. 345-365, 1965.

[53] Steward, D., "The Design Structure System: A Method for Managing the Design of Complex Systems," *IEEE Transactions on Engineering Management*, Vol. EM-28, No. 3, 1981.

[54] Stinson, J., Davis, E., and Khumawala, B, "Multiple Resource-Constrained Scheduling Using Branch and Bound," *AIIE Trans.*, Vol. 10, No. 3, pp 252-259, 1978.

[55] Takeuchi, H. and Nonaka, I., *The New Product Development Game*, Harvard Business Review, January-February, pp. 137-146, 1986.

[56] Talbot, F. and Patterson, J., "An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems," *Management Science*, Vol. 25, No. 11, pp 1163-1174, 1978.

[57] Taylor, B. and Moore, L., "R&D Project Planning with Q-GERT Network Modeling," *Management Science*, Vol. 26, No. 1, pp. 44-59, 1980.

[58] Ulrich, K. and Eppinger, S., *Product Design and Development*, 2nd Edition, McGraw-Hill, New York, 2000.

[59] von Hippel, E., "Task Partitioning: An innovation process variable," *Research Policy*, Vol. 19, pp. 407-418, 1990.

[60] Warfield, J., "Binary Matrices in System Modeling," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-3, No. 5, 1973.

[61] Weinblatt, H., "A new search algorithm for finding the simple cycles of finite directed graphs," *Journal of ACM*, Vol. 19, pp 43-56, 1972.

[62] Weist, J, "Some properties of schedules of schedules for large projects with limited resources," *Operations Research*, Vol. 12, pp 395-418, 1964.

[63] Wheelwright, S. and Clark, K., *Revolutionizing Product Development: Quantum leaps in Speed, Efficiency and Quality*, Free Press, 1992.

[64] Williams, T., "Practical Use of Distributions in Network Analysis," *Journal of the Operational Research Society*, Vol. 43, pp 265-270, 1992.

[65] Williams, T., "Criticality in Stochastic Networks," *Journal of the Operational Research Society*, Vol. 43, pp. 353-357, 1992..

[66] Woodworth, B. and Shanahan, S., "Identifying the critical sequence in a resource constrained project," *International Journal of Project Management*, Vol. 6, pp. 89-96, 1988.

[67] Yassine, A., Falkenburg, D., and Chelst, K., "Engineering design management: an information structure approach," *International Journal of Production Research*, Vol. 37, No. 13, pp 2957-2975, 1999.

[68] Zeigler, B., Praehofer, H., and Kim, T., *Theory of Modeling and Simulation – Integrating Discrete Event and Continuous Complex Dynamic Systems*, 2nd Edition, Academic Press, 2000.

Appendix

A1. Algorithm for the Analysis of a Coupled Block

A1-1. Model Inputs

For $i = 1, \ldots, n; j = 1, \ldots, n$ and $k = 1, \ldots, l$,

n ; total number of tasks

l ; total number of resources available for the project

$(R_k)_{total}$; total amount of resource k available for the project

$(R_k)_i$; amount of resource k that task i requires

D_{act} ; actual duration of the project that has been in progress

$(d_{act})_i$; actual duration of task i that has been in progress

$(d_{opt})_i$; estimate for optimistic remaining duration of task i

$(d_{likely})_i$; estimate for most likely remaining duration of task i

$(d_{pess})_i$; estimate for pessimistic remaining duration of task i

$$DSM(i, j) = \begin{cases} 0 & \text{if no dependency between task } i \text{ and } j, \text{ or } i = j \\ 1 & \text{if there is the finish-to-start type of information flow from task } j \text{ to task } i \\ 2 & \text{if there is the finish-to-start-plus-lead type of information flow from task } j \text{ to task } i \end{cases}$$

$OA(i, j)$; maximum overlapping amount of task i with task j

$OI(i, j)$; expected rework impact of task i from the overlapping with task j

$RP(i, j, r)$; rework probability that task j causes rework to task i in the rth iteration ($r = 1,2, \ldots$)

$RI(i, j)$; rework impact which represents the percentage of task i to be reworked when the rework is caused by task j

$(L_{ori})_i$; learning curve which represents the percentage of original duration when task i does the same work in the second time

$(L_{\max})_i$; maximum learning curve which represents the minimum percentage of original duration when task i does the same work repeatedly

$P_{tolerance}$; probability that determines rework concurrency of tasks

A1-2. Model Variables

For $i, j = 1, \ldots, n$; $k = 1, \ldots, l$; $m = 1, \ldots, N$; *and* $r = 1, 2, \ldots$

q ; state

$(R_k)'_q$; total amount of resource k available in state q

D ; total project duration until the current state

D_q ; duration of state q

$(d_{LHS})_{(i,m)}$; expected duration of task i in the mth simulation run sampled using the *LHS*

d_i ; remaining duration of task i

$(d_{ori})_i$; original duration of task i defined as the sum of $(d_{act})_i$ and d_i at the start of each simulation run

$(d_{complete})_i$; total project duration when task i is completed

$RP'(i, j, r)$; rework probability in the current state

L_i ; learning curve when repeating task i in the current state

$C(i, 1) = \begin{cases} 1 & \text{if task } i \text{ has been completed before} \\ 0 & \text{otherwise} \end{cases}$

$C(i, 2) = \begin{cases} 1 & \text{if task } i \text{ has been reworked before} \\ 0 & \text{otherwise} \end{cases}$

$PR_i = \begin{cases} 1 & \text{if task } i \text{ has all the necessary information to be worked on} \\ 0 & \text{otherwise} \end{cases}$

$CPR(i, 1) = \begin{cases} 1 & \text{if } PR_i = 1 \text{ and } d_i \neq 0 \text{ in the current state} \\ 0 & \text{otherwise} \end{cases}$

$CPR(i, 2) = \begin{cases} 1 & \text{if } CPR(i, 1) = 1 \text{ in the previous state} \\ 0 & \text{otherwise} \end{cases}$

$RR(i, 1) = \begin{cases} 1 & \text{if } CPR(i, 1) = 1 \text{ and task } i \text{ has all the necessary resources to be worked on in the current state} \\ 0 & \text{otherwise} \end{cases}$

$PR(i, 2) = \begin{cases} 1 & \text{if } RR(i, 1) = 1 \text{ in the previous state} \\ 0 & \text{otherwise} \end{cases}$

$RC(i, j)$; total direct and indirect rework probabilities that task j causes rework to task i

A1-3. Simulation Algorithm

For $p = 1, \ldots, N$, follow STEP1 through STEP7 below.

STEP1. Initialize model variables from the inputs at state *0* of the pth simulation run.

1. Set $d_i = (d_{LHS})_{(i,\, p)}$ and $(d_{ori})_i = (d_{act})_i + d_i$ for $i = 1, \ldots, n$.

2. Set $D = D_{act}$ and $(d_{complete})_i = 0$ for $i = 1, \ldots, n$.

3. Set $PR_1 = 1$ and $PR_i = 0$ for $i = 2, \ldots, n$.

4. Set $L_i = 1$ for $i = 1, \ldots, n$.

5. Set $RP(i, j, r) = RP'(i, j, r)$ for $i, j = 1, \ldots, n;\ r = 1,2,\ldots$

6. Set $C(i, 1) = 1$ if $(d_{likely})_i = 0$ for $i = 1, \ldots, n$. Otherwise $C(i, 1) = 0$ for $i = 1, \ldots, n$.

7. Set $C(i, 2) = 0$ for $i = 1, \ldots, n$. (Assumption: tasks have not been reworked yet – this is for computing overlap amounts.)

8. Set $q = 1$.

STEP 2. Initialize model variables in the current state q.

1. Set $RR(i, 2) = RR(i, 1)$ and $CPR(i, 2) = CPR(i, 1)$ for $i = 1, \ldots, n$.

2. Set $CPR(i, 1) = 0$ and $RR(i, 1) = 0$ for $i = 1, \ldots, n$.

3. Set $Q_1 = Q_2 = Q_3 = \phi$.

4. Set $(R_k)'_q = (R_k)_{total}$ for $k = 1, \ldots, l$.

STEP 3. Identify a set of concurrent active tasks in the current state q satisfying precedence and resource constraints based on the priority rules.

1. Find $a = \min i \in \{2, \ldots, n \mid d_i \neq 0\}$.

2. If $a = n$, then set $PR_n = CPR(n, 1) = RR(n, 1) = 1$ and go to 8.

3. If there exists $i \in \{1, \ldots, n \mid C(i, 1) = 1$ and $d_i \neq 0\}$ (*i.e.* there is any task that is reworked in the current state),

   (1) find $b = \max i \in \{2, \ldots, n \mid PR_i = 0$ and $d_i \neq 0\}$.

   (2) compute the *Rework Concurrency* between tasks a and b as follows:

   for $i = a + 1, \ldots, b$,

   (i) set $RC(ii, j) = RP(ii, j, 1)$ for $j = a, \ldots, b - 1$.

   (ii) for $i > a + 1$, execute the following loop:

   for $j = i - 2$ to 1 decreasing by 1

   for $k = j + 1$ To $i - 1$

   $RC(ii, j) = RC(ii, j) + RP(k, j, 1) \times RC(i, k)$

   next k

   next j

   (3) for $i \in \{a, \ldots, n \mid PR_i = 0$ and $d_i \neq 0\}$,

if $d_j = 0$ for every $j \in \{ a, \ldots, i-1 \mid (DSM(i, j) \neq 0$ and $C(j, 1) = 0)$ or $(RC(i,j) > P_{tolerance}$ and $C(j, 1) = 1) \}$, then set $PR_i = 1$.

Otherwise,

for $i \in \{ a, \ldots, n \mid PR_i = 0$ and $d_i \neq 0 \}$,

if $d_j = 0$ for every $j \in \{ 1, \ldots, i-1 \mid DSM(i, j) \neq 0 \}$, then set $PR_i = 1$.

4. For $i \in \{ 1, \ldots, n \mid PR_i = 1$ and $d_i \neq 0 \}$, set $CPR(i, 1) = 1$.

5. Set $Q_1 = \{ i = 1, \ldots, n \mid CPR(i, 1) = 1 \}$.

6. Calculate $(R_k)_{required} = \sum_i (R_k)_i$ for $k = 1, \ldots, l$ where $i \in Q_1$.

7. If $(R_k)_{required} \leq (R_k)'_q$ for every $k = 1, \ldots, l$ (*i.e.* there are no over-allocated resources), set $RR(i, 1) = 1$ and $(R_k)_q = (R_k)'_q - (R_k)_i$ for $k = 1, \ldots, l$; and $I \in Q_1$ (i.e. assign resources to the tasks and reduce resource capacity in the current state).

Otherwise, sort the elements of Q_1 in the descending order of resource priorities based on the priority rules. For $i \in Q_1$, beginning with the task with the highest priority, if $(R_k)_i \leq (R_k)'_q$ for every $k = 1, \ldots, l$, then set $RR(i, 1) = 1$ and $(R_k)_q = (R_k)'_q - (R_k)_i$ for $k = 1, \ldots, l$.

8. Set $Q_2 = \{ i = 1, \ldots, n \mid RR(i, 1) = 1 \}$.

STEP 4. Adjust overlapping iteration for the tasks in Q_2.

1. Set $Q_3 = \{ i \in Q_2 \mid RR(i, 2) = 0, CPR(i, 2) = 0$ and $\exists j = 1, \ldots, n$ *s.t.* $OA(i, j) \neq 0 \}$

2. If $Q_3 = \phi$, then go to STEP5.

Otherwise, for $i \in Q_3$,

(1) Set $K_1 = K_2 = K_3 = 0$

(2) If there exists $j \in \{ i + 1, \ldots, n \mid (d_{complete})_j = D$ and $OA(i, j) \neq 0 \}$, set the overlap amount K_1 of task i as $OA(i, j) \times d_i$ and go to (4). (This implies that task i starts to do feedback rework in the current state. It is assumed that feedback rework can be overlapped with only one task that causes the rework.)

(3) If there exists $j \in \{ 1, \ldots, i-1 \mid (d_{complete})_j = D$, $OA(i, j) \neq 0$ and $RR(j, 1) = 0 \}$ and for calculate the overlap amount K_1 as follows:

$$K_1 = \min (D - (d_{complete})_j + OA(i, j) \times d_i)$$

where $j \in \{1, \ldots, i-1 \mid OA(i, j) \neq 0$ and $RR(j, 1) = 0 \}$

Otherwise, go to (7).

(4) Calculate the actual overlap amount K_2 of task i based on resource availability in the prior states. If $(R_k)_i \leq (R_k)'_{q-1}$ for every $k = 1, \ldots, l$, set $K_2 = K_2 + \min(K_1, D_{q-1})$ and $K_1 = K_1 - D_{q-1}$. Otherwise go to (5). If $K_1 > 0$, repeat this procedure for prior states q-2, q-3, ... until $K_1 \leq 0$.

(5) If $C(i, 1) = 0$ and there exists $j \in \{1, \ldots, i-1 \mid OA(i, j) \neq 0, RR(j, 1) = 0$ and $C(j, 2) = 1 \}$, set K_2 equal to the minimum of K_2 and $\min (D - (d_{complete})_j +$ $(rework\ amount\ performed\ in\ previous\ iteration)_j) \}$

(6) If task i is overlapped with only one task, then calculate the actual overlap impact K_3 as follows:

$$K_3 = K_2 \times OI(i, j)$$

Otherwise,

$$K_3 = \sum_j \left((K_2 - D + (d_{complete})_j) \times OI(i, j) \right)$$

where $j \in \{1, \ldots, i-1 \mid OA(i, j) \neq 0$ and $RR(j, 1) = 0\}$

(7) Set $d_i = d_i - K_2 + K_3$.

STEP 5. Adjust the total project duration and the remaining durations of the tasks in Q_2.

1. Set $D_q = \min d_i$ for $i \in Q_2$.
2. Set $D = D + D_q$.
3. Set $d_i = d_i - D_q$ for $i \in Q_2$.
4. If $d_i = 0$ for $i \in Q_2$, then
   (1) Set $(d_{complete})_i = D$.
   (2) If $C(i, 1) = 0$, set $C(i, 1) = 1$. Otherwise, set $C(i, 2) = 1$.
   (3) Set $L_i = \max (L_i \times (L_{orig})_i , (L_{\max})_i)$.

STEP 6. Generate sequential iteration rework.

For each task $i \in \{ i \in Q_2 \mid (d_{complete})_i = D \}$,

<feedback rework>

1. If $i > 1$, for $j \in \{ 1, \ldots, i-1 \mid RP'(j, i, 1) \neq 0 \}$,

if a random number between 0 and 1 is less than $RP'(j, i, 1)$, then

(1) Set $d_j = \min(d_j + (d_{ori})_j \times RI(j, i) \times L_j, \ (d_{ori})_j \times (L_{ori})_j)$

(2) Set $RP'(j, i, k) = RP'(j, i, k+1)$ for $k = 1, 2, \ldots$

(3) Set $PR_j = 0$ and $PR = 0$ for all the successors of j.

<feedforward rework>

2. If $C(i, 1) = 1$ and $i < n$, for $j \in \{ i+1, \ldots, n \mid RP'(j, i, 1) \neq 0$ and $d_j \neq (d_{orig})_j \}$,

if a random number between 0 and 1 is less than $RP'(j, i, 1)$, and it is not true that rework of task i has been generated by task j and has never been executed, then

(1) If C(j, 1) = 1,

set $d_j = \min(d_j + (d_{ori})_j \times RI(j, i) \times L_j, \ (d_{ori})_j \times (L_{ori})_j)$.

Otherwise,

If $d_j < (d_{ori})_j \times 0.9$,

set $d_j = \min(d_j + (d_{ori})_j \times RI(j, i) \times L_j, \ (d_{ori})_j \times 0.9)$.

Otherwise,

set $d_j = \min(d_j + (d_{ori})_j \times RI(j, i) \times L_j, \ (d_{ori})_j)$

(2) Set $RP'(j, i, k) = RP'(j, i, k+1)$ for $k = 1, 2, \ldots$

(3) Set $PR_j = 0$ and $PR = 0$ for all the successors of j.

STEP 7. Test a termination of the simulation run.

If $d_i = 0$ for every $i = 1, \ldots, n$, the pth simulation run is complete. Otherwise, set $q = q + 1$ and go to STEP2 (*i.e.* make a transition to the next state).

<Note>

For conciseness, some minor details of algorithms are explained verbally and some model variables are not included.

A2. Comparison with Other Rework Process Models

This section explains different aspects of the proposed model with respective to task concurrency and (sequential) rework generation. Note that all the other rework models mentioned below do not account for resource conflicts and overlapping iteration.

A2-1. Analytical Models

By definition, a network is in a certain state at a certain instance, and the sum of state-transition probabilities is equal to one in a Markov chain or a reward Markov chain. Since a state represents a task in the process models based upon a Markov chain (Ahmadi and Wang 1999) or a reward Markov chain (Smith and Eppinger 1997a), they have inherent limitation to model a project having parallel tasks (or paths) as shown in Figure A-1 (a). For the same reason, tasks *a* and *b* in Figure A-9 (b) cannot be reworked in parallel after task *c* is completed.

(a) Parallel Path *(b) Sequential Path*

FIGURE A-1. EXAMPLES FOR COMPARISON WITH ANALYTICAL MODELS

Eppinger et al. (1997) made an extension to the analytical model using a signal flow graph in order to model parallel task execution. However, as they discussed in their work, the result quantity is the total effort (sum of durations of all tasks) instead of the lead time. In addition, the model is not able to account for the 'AND' rules, where a number of tasks have to be completed before a task starts.

A2-2. DSM-Based Simulation Model (Browning and Eppinger 2000)

The DSM-based simulation model was the first attempt to overcome inherent limitations of the analytical methods for modeling iteration in parallel tasks (or paths). They used the discrete *time* simulation to achieve this purpose (note that the thesis employs the discrete *event* simulation). In this section, the author discusses its assumptions in task concurrency and rework, and consequent impacts to lead time computation. The summary of improvements in the proposed model is given at the end of the section.

<1> Task concurrency using the "bands"

"Banding" is the graphic technique proposed by Grose (1994) to show a set of tasks in the DSM that can work in parallel potentially. Browning (1999) made the following comments about this technique:

> "An innovation of adding alternating light and dark bands to a DSM to show potentially concurrent activities comes from the work from Grose ..... Since one activity in each band sits on the critical path, fewer bands are preferred, implying greater concurrency and shorter process duration for a given set of activities."

Browning and Eppinger (2000) presented the following algorithm using the bands to identify a set of concurrent tasks (or activities) working in each time step:

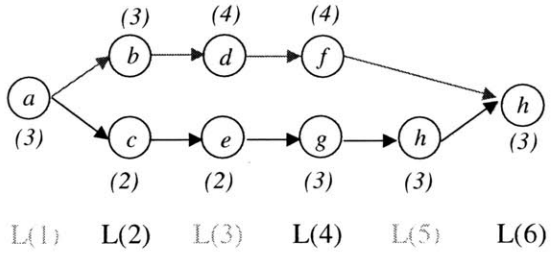
> "Find first activity, *i*, that has unfinished work .....
> Loop through subsequent activities to identify concurrent work for the current time step. If next activity has unfinished work and is not dependent on an unfinished, upstream activity, then set its WN entry to TRUE *(i.e. this activity works in the current time step).* Otherwise, the complete band has been found (stop checking activities)."
>
> *<Note> The author added italicized words above for clarity.*

A simple example is given in Figure A-2 to illustrate possible marginal errors that might be caused by the above algorithm. The example has nine tasks with six bands in two

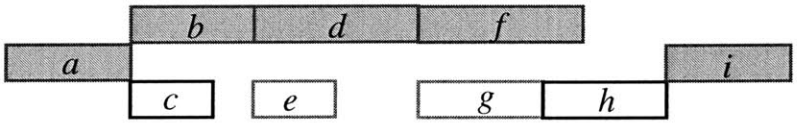
parallel paths as illustrated in (a) and (b). The numbers in the parenthesis at (b) represent the durations of tasks. Note that the bands are equivalent to the process levels based on the as-early-as-possible rule in the network without feedback information flow. The path along the tasks *a-b-d-f-h* is a critical path with 19 units while the tasks *c*, *e*, *g*, and *h* have one unit of slack.

Task Name		1	2	3	4	5	6	7	8	9
a	1									
b	2	1								
c	3	1								
d	4		1							
e	5			1						
f	6				1					
g	7					1				
h	8							1		
I	9						1		1	

(a) DSM with Bands



(b) Project Network



(c) Simulated Gantt Chart Following the Above Algorithm

FIGURE A-2. TASK CONCURRENCY OF THE FIRST DSM SIMULATION MODEL

At the time step right after task *c* has been completed, following the algorithm in the above,

(i) Task *b* is the first task with unfinished work.
(ii) Task *c* is the next task but does not have unfinished work. Thus, a complete band including only task *b* is found.

However, task *e* should start to work when task *c* is over, according to their assumption that tasks are supposed to start as early as possible once upstream dependent tasks are completed. The same explanation can be made for the delayed start of task *g*. Thus, the lead time is overestimated by 2 units as illustrated in the figure. There are more chances to give a misleading result when a project network has multiple paths with more than two tasks in each path. Besides, task *h* is not on the critical path even though it takes one band alone. Generally speaking, it is not true that only one task in each band sits on the critical path.

<2> Rework generation

In this section, rework generation patterns of the algorithm proposed by Browning and Eppinger (2000) are discussed using the case example of Browning (1999) in Figure A-3. The simulated Gantt chart in (b) is one of many possible outcomes from the rework probability matrix in (a). Following their algorithm,

(1) At the time step *t0* when task 9 is completed,
  (i) First-order (feedback) rework is generated to task 2
  (ii) Second-order (feedforward) rework is generated to task 3 due to rework of task 2
(2) At the time step *t1* when task 2 starts to be reworked,
  (i) Task 2 is the first task with unfinished work.
  (ii) The next task 3 has unfinished work but is dependent upon task 2. Thus, a complete band including only task 2 is found.
(3) At the time step *t2* when task 3 starts to be reworked,
  (i) Task 3 is the first task with unfinished work.

(ii) Task 11 is the next task with unfinished work and is not dependent upon task 2. The next task 12 has unfinished work but is dependent upon task 11. Thus, a complete band including tasks 3 and 11 is found.

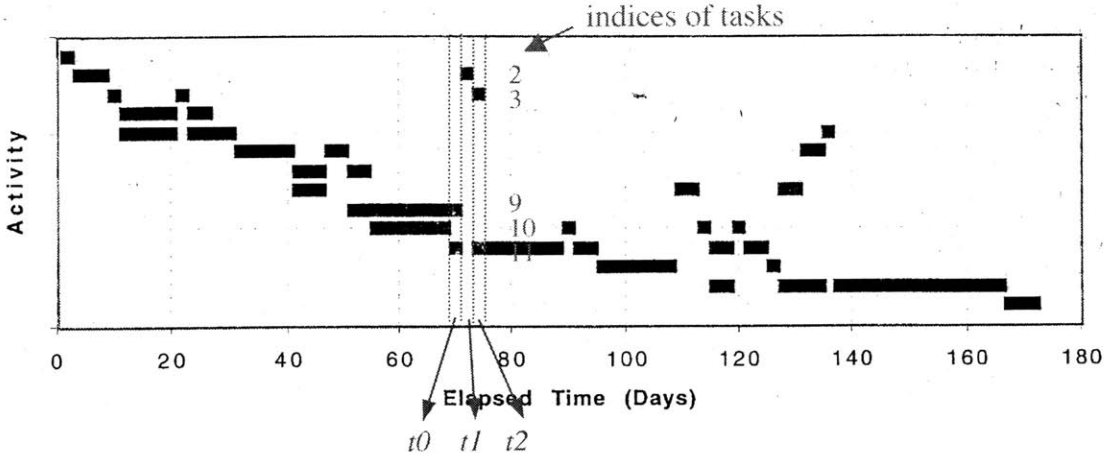
<Note>

There may be other time steps between time steps *t0* and *t1*, and between *t1* and *t2*, if the constant size of time step is smaller than the time for the rework of task 2 or 3 shown in the figure.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
A4	Prepare UAV Preliminary DR&O	1													
A511	Create UAV Preliminary Design Configuration	2	.4							.2					
A512	Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings	3		.5		.4									
A531	Perform Aerodynamics Analyses & Evaluation	4	.3		.5										
A521	Create Initial Structural Geometry	5	.4		.5			.1		.1			.3	.1	
A522	Prepare Structural Geometry & Notes for FEM	6	.1				.4								
A5341	Develop Structural Design Conditions	7	.4					.4							
A532	Perform Weights & Inertias Analyses	8						.5					.5		
A533	Perform S&C Analyses & Evaluation	9	.4		.5	.5				.5					
A5342	Develop Balanced Freebody Diagrams & Ext. Applied Loads	10				.1		.5	.2	.1		.4			
A5343	Establish Internal Load Distributions	11						.5	.5	.5		.5			
A5344	Evaluate Structural Strength, Stiffness, & Life	12	.4					.4	.5			.5	.4		
A54	Preliminary Manufacturing Planning & Analyses	13	.5				.5						.4		
A6	Prepare UAV Proposal	14	.3	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	

Source: Browning (1999) Chapter 6. pp.174, Table 5

(a) DSM Dimension One – Rework Probability



Source: Browning (1999) Chapter 6. pp 168, Figure 2

(b) Simulated Gantt Chart of One Scenario

FIGURE A-3. REWORK GENERATION PATTERNS OF THE FIRST DSM SIMULATION MODEL

The below is the summary of the issues identified in the example:

1. *Task concurrency* – Following the banding algorithm, task 11 did not work in parallel with task 2 at *t1* while it did with task 3 at *t2*. But there is no rationale for this interruption of task 11. This shows another limitation of the banding-algorithm's stopping rule in iteration – the algorithm stops checking other downstream tasks even when it misjudges that a complete band has been found.

2. *Timing of second-order rework generation* – The second-order rework of task 3 was determined at the same time when the rework of task 2 was determined. Because of *pre-generated* rework of task 3, task 11 was interrupted at *t1*, which shouldn't have happened otherwise. Besides, in practice, the decision point of rework of task 3 is more likely when the rework of task 2 is completed.

3. *Chain feedforward rework after second-order rework* – In the above example, the rework of task 3 may cause additional feedforward rework to tasks 4, 5 and 9, which may cause subsequent reworks to further downstream tasks, as indicated by the rectangles in (a). If theses successive feedforward reworks are not accounted as in their algorithm, the model may significantly underestimate the lead time. In addition, it does not take into account the size of a feedback loop which can be measured by the distance of a feedback mark from the diagonal in the DSM.

<3> Summary of improvements from the first DSM-Based simulation model

Browning (1999) provided insightful suggested extensions for his model. Below is a list of topics and achievements of the proposed model:

- dynamic durations and iterations – *solved*
  (The task durations are updateable and the iteration characteristics such as rework probability and the learning curve are changed in each iteration.)
- increased task concurrency (overlapping iteration) – *solved*
- resource requirements and constraints – *solved*
- distinctive rework probability in each output – *not incorporated*
- independent blocks of highly coupled activities – *solved*

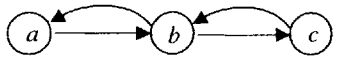
- cost modeling (NPV) – *not incorporated*

Additional improvements not included in the above list are:

- identifying a correct set of concurrent tasks
- modeling successive chain feedforward rework
- employing the discrete event simulation instead of the discrete time simulation, which saves computation time and increases accuracy
- employing the Latin Hypercube Sampling technique instead of Monte Carlo sampling technique, which reduces the number of simulation runs for convergence.

A2-3. Notes on the Proposed Model in Rework Generation

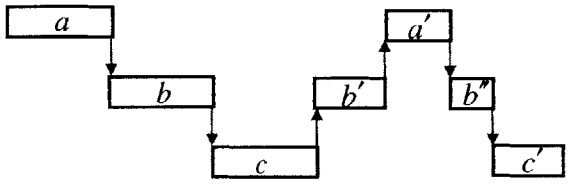
Consider a simple example in Figure A-4. The scenario in (c) is under the assumption that the total rework probability that either task a or b causes task c to rework is larger than $P_{tolerance}$. After the first iteration of task b (*i.e.* b'), there is a chance that both upstream task a and downstream task c need to incorporate new information from the rework b'. When that case happens, task c waits until rework of task a (*i.e.* a') is completed. The model simulates that the rework of task c accumulated after the completion of b' is performed as if it updated latest information from the second iteration b''.



(a) Information Flow Diagram

Task Name		1	2	3
a	1		1	
b	2	1		1
c	3		1	

(b) DSM



(c) One Possible Outcome from Simulation

FIGURE A-4. REWORK GENERATION PATTERN OF THE PROPOSED MODEL

A3. Definitions of Heuristic Priority Measures

A3-1. Rank Positional Weight (RPW)

Cooper (1976) defined a rank positional weight of task i as follows:

$$rpw_i = d_i + \sum_j d_j$$

where, d_i: expected duration of task i

$\sum_j d_j$: sum of all expected durations over all successors of task i

(Note: a set of successors includes all downstream tasks that receive outputs from the task.)

By adding the summation part of the rank positional weight, it measures global importance of a task while it measures only local importance among neighbor tasks without it (Cooper 1976).

A3-2. Cumulative Resource Equivalent Duration (CUMRED)

Cooper (1976) defined a cumulative resource equivalent duration of task i using a resource usage duration (Pascoe 1965) of each resource as follows:

$$RUD_k = L_k + E_k / R_k$$

where, RUD_k : resource usage duration of resource k

L_k : last time period of requirement for resource k

E_k : total excess requirement for resource k

R_k : total available amount of resource k

$RUDMAX = Maximum \{ RUD_k \}$ where $k = 1, 2, \ldots, K$ (number of resources)

$$RED_i = \sum_{k=1}^{K} (r_{ik} \; / \; R_k) \cdot (RUD_k \; / \; RUDMAX) \cdot d_i$$

where, RED_i : resource equivalent duration of task i

r_{ik} : amount of resource k required by task i

$$CUMRED_i = RED_i + \sum_j RED_j$$

where, $CUMRED_i$: cumulative resource equivalent duration of task i

$\sum_j RED_j$: sum of RED over all successors of task i

A4. Explicit Resource Links

During the forward pass computation, the model creates explicit resource links between the tasks that cause a resource conflict. If the conflict is between two tasks, it simply creates the link from the task with a higher resource priority to the one with a lower priority. If more than two tasks are involved, it requires a careful examination. The example in Figure A-5 is used for illustration. Figure A-6 shows two possible alternative methods for creating explicit resource links in the example, under the assumption that (i) all the durations of tasks are equal, (ii) tasks *b*, *c*, and *d* compete for the same resource *A* while two units of *A* are available, and (iii) the order of resource priorities are task *c*, *b*, and *d* from the highest. Note that resource links are considered as if they were precedence constraints during the backward pass computation.

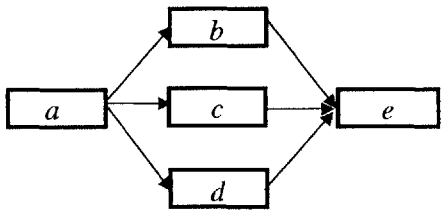


FIGURE A-5. RESOURCE CONFLICTION AMONG PARALLEL TASKS

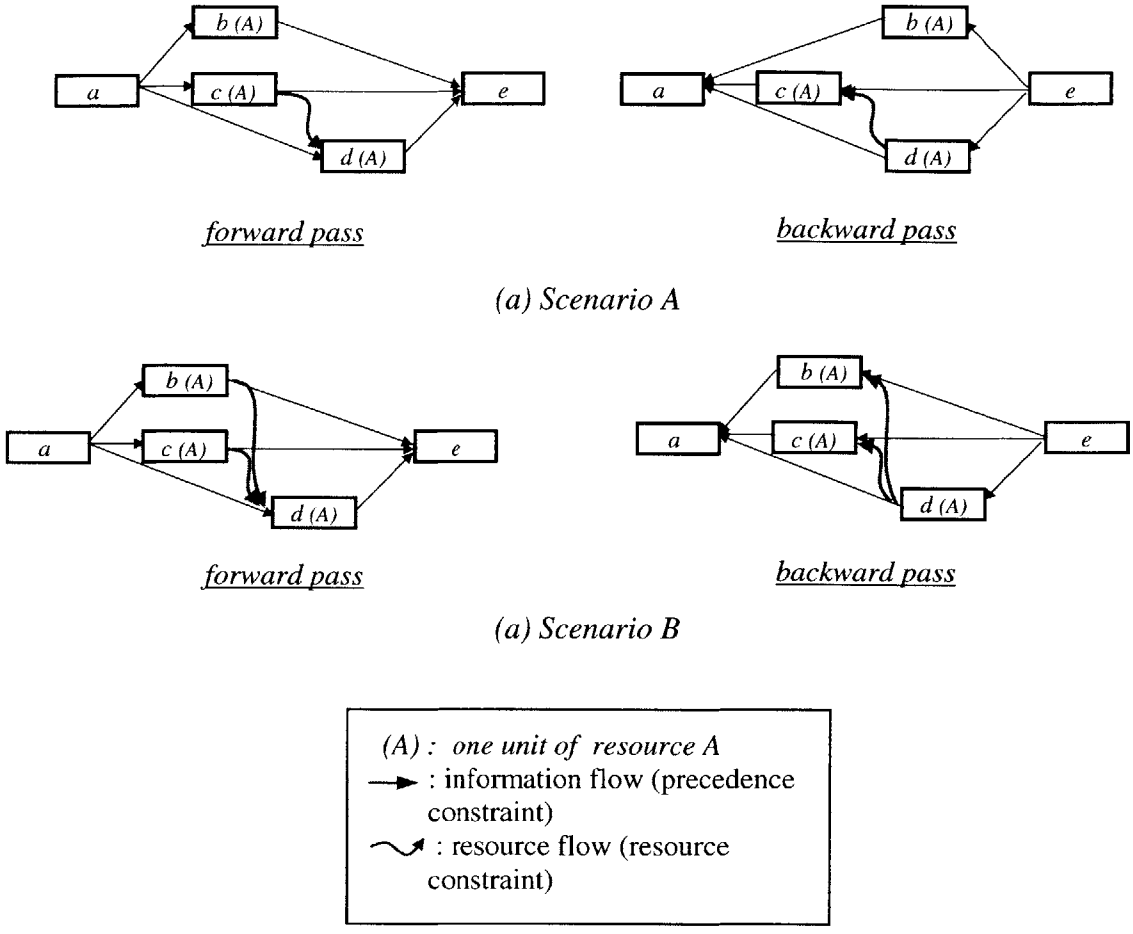


FIGURE A-6. ALTERNATIVES METHODS FOR CREATING EXPLICIT RESOURCE LINKS

In the first scenario, task *d* is delayed because of its lowest priority and one explicit resource link is created between tasks *c* and *d* assuming that the same resource *A* that is assigned to task *c* is assigned to task *d*. Under this scenario, only the RC slack of task *b* is nonzero although task *d* has a lower priority than task *b*. In the same way, if the resource link is created between tasks *b* and *d*, task *c* with a highest priority has some RC slack. Thus, this method reveals inconsistency that task *d* with a lowest priority is critical (because it has zero RC slack) while task *b* or *c* with a higher priority has nonzero RC

slack. This is the method of creating resource links when more than two tasks compete for limited resources in Bowers (1995).

In the second scenario, two explicit resource links are created even though task *d* uses only one unit of resource *A*. Under this scenario, all three tasks are critical because they have no RC slack. Thus, the tasks have RC slack in accordance with their resource priorities which represent the relative importance of tasks. This is the method developed in this thesis that creates explicit resource links from *all* active tasks, that finish in a certain state, to the tasks that have all inputs but are delayed due to lower priorities for the same resource used by those active tasks.

In the method proposed by Bowers, a resource link is created for any resource assignment, as illustrated in Figure A-7. And there is no need to account for resources in the backward pass computation using this method. In contrast, the proposed method does not create a resource link unless there exists a resource conflict between two tasks during the forward pass computation. Thus, it does not create the resource links in the figure while it requires that resource conflicts be handled in the backward pass computation as well. For instance, during the backward pass computation, if one unit of resource *A* is available for the project, the model simulates that task *b* is delayed after task *d* by interpreting that task *d* has a lower priority during the forward pass computation because it started later than task *b*.

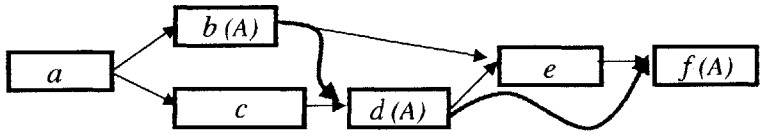


FIGURE A-7. RESOURCE LINKS CREATED TO ALL RESOURCE FLOWS

A5. Detailed Data & Computation Results of Logistics Europe Project

Table A-1 summarizes the data and computation results of the Logistics Europe project in a task level. Resource requirements are not included in the table because there was a resource conflict only between tasks 6 and 13 within this project. In reality, the resources used in this project did multi-tasking with other projects. More extensive data collection would give more precise results constrained by limited resource capacity. Figure A-8 shows the data related to iterations. The learning curve was determined as 30% for the tasks that calculates numeric values using computer models, 50% for review tasks, and 70% for other tasks, while the maximum learning curve is determined as 35%, 20%, and 50%, respectively. No additional overlapping was planned other than between the tasks within coupled blocks as indicated in the figure.

		Expected Duration			Prede-	Slack	Criticality	RC_Slack	RC_Criticality
Task ID	Task Name	Opt	Likely	Pess	cessors	(avg,sd)	%	(avg,sd)	%
1	Project approval	1	1	5		0, 0	100	0, 0	100
2	Develop European long-term business strategy	5	10	15	1	0, 0	100	0, 0	100
3	Develop European Materials & Logistics strategy	5	10	15	2	0, 0	100	0, 0	100
4	Establish European 2001 budget	5	10	15	3	0, 0	100	0, 0	100
5	Kick-off meeting	1	1	5	4	0, 0	100	0, 0	100
6	Develop the details of action items	5	15	20	5	1.6, 3.1	65	1.4, 2.9	67
7	Develop customer delivery profile	5	10	15	5	4.1, 4.7	35	4.3, 4.7	33
8	Block1: Price Negotiation				6, 7	.1, .8	93	.3, 1.2	87
9	Review the price agreement for this year	1	2	3					
10	Define additional parameters for new prices	2	4	6	9FS				
11	Determine new prices	5	10	15	9, 10				
12	(D) Block1 Duration	10.6	17.3	26.7					
13	Establish normal lead time of delivery by locations	5	10	15	7	12.4, 8.3	7	8.2, 6.7	13
14	Block2: Process Definition				8, 13	0, 0	100	0, 0	100
15	Modify current procedures in logistics	5	15	25					
16	Calculate worldwide projected inventory levels	1	3	6	15FS				
17	Review modified procedure in logistics	2	6	12	15FS				
18	Calculate new safety stocks for EU DC	1	2	3	16				
19	Milestone review	2	3	10	15,16,17,18				
20	(D) Block2 Duration	18	31.6	48.8					
21	Block3: Customer Testing				14	0, 0	100	0, 0	100
22	Determine sample customers for testing	1	2	4					
23	Ship a series of products to sample customers	10	12	14	22				
24	Review testing results	2	4	6	23				
25	(D) Block3 Duration	17.4	27.6	38.6					
26	Modify replenishment program	2	4	6	14	33.3, 9.2	0	33.3, 9.2	0
27	Adjust safety stocks in EU DC	1	2	3	21	0, 0	100	0, 0	100
28	Adjust international rolling forecast	1	2	3	27	0, 0	100	0, 0	100
29	Implement new procedure in logistics	1	5	10	28	0, 0	100	0, 0	100
30	Close IW and ship remaining inventories to EU DC	8	16	24	26, 29	0, 0	100	0, 0	100
31	Distribute all the products from the EU DC	2	3	4	30	0, 0	100	0, 0	100

<Note> "Task ID" is different from "ID" in the DSM.

TABLE A-1. DATA AND COMPUTATION RESULTS OF LOGISTICS EUROPE PROJECT

ID	Task Name	Ori. Learning	Max. Learning
8	Review the price agreement for this year	50%	35%
9	Define additional parameters for new prices	50%	35%
10	Determine new prices	50%	35%

<Rework Probability>

1st Iteration

	8	9	10
8			0.5
9	0.8		
10	1.0	1.0	

2nd Iteration

	8	9	10
8			0.2
9	0.2		
10	1.0	1.0	

3rd Iteration

	8	9	10
8			0.1
9	0.1		
10	1.0	1.0	

<Rework Impact>

	8	9	10
8			0.8
9	0.6		
10	0.6	0.4	

<Overlap Amount>

	8	9	10
8			
9	0.4		
10			

<Overlap Impact>

	8	9	10
8			
9	0.2		
10			

(a) Block1: Price Negotiation

ID	Task Name	Ori. Learning	Max. Learning
12	Modify current procedures in logistics	70%	50%
13	Calculate worldwide projected inventory levels	30%	20%
14	Review modified procedure in logistics	50%	35%
15	Calculate new safety stocks for EU DC	30%	20%
16	Milestone review	50%	35%

<Rework Probability>

1st Iteration

	12	13	14	15	16
12			0.5		0.3
13	1.0				
14	1.0				
15		1.0			
16	1.0	1.0	1.0	1.0	

2nd Iteration

	12	13	14	15	16
12			0.1		0.1
13	1.0				
14	1.0				
15		1.0			
16	1.0	1.0	1.0	1.0	

<Rework Impact>

	12	13	14	15	16
12			0.5		0.5
13	0.2				
14	0.7				
15		0.2			
16	0.5	0.5	0.5	0.5	

<Overlap Amount>

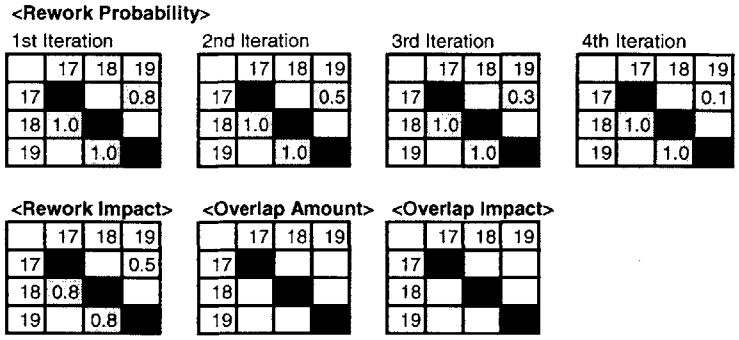
	12	13	14	15	16
12					
13	0.3				
14	0.5				
15					
16					

<Overlap Impact>

	12	13	14	15	16
12					
13	0.5				
14	0.2				
15					
16					

(b) Block2: Process Definition

ID	Task Name	Ori. Learning	Max. Learning
17	Determine sample customers for testing	70%	50%
18	Ship a series of products to sample customers	70%	50%
19	Review testing results	50%	35%



(c) Block3: Customer Testing

<Note> "ID" is based on that in the *AEAP DSM*.

FIGURE A-8. DATA FOR ITERATIONS IN LOGISTICS EUROPE PROJECT

5701-19