SYSTEMS THEORY BASED METHOD FOR CONSTRUCTION PROJECT PLANNING

By

WEN LIU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2008

To my husband, Zhen Zhao;
my son, Daniel Zhao;
and my parents, Ying Fan and Xiqing Liu.

ACKNOWLEDGMENTS

I would like to gratefully and sincerely thank Dr. Ian Flood and Dr. Raymond Issa for their guidance, understanding, patience and encouragement during my graduate studies at Rinker School. Dr. Flood's insightful comments and constructive suggestions at different stages of my research were essential to the completion of this dissertation. He has also taught me innumerable lessons on the workings of academic research in general. Dr. Issa has spent many hours reading this work and given me a lot of editorial advice. His mentorship was paramount in providing a well-rounded experience consistent with my long-term career goals.

My thanks also go to Dr. Robert Cox at the Purdue University, who came all the way from Indiana to attend my final defense.

I am also indebted to the School of Technology at Michigan Tech, for the opportunity to work in the Department of Construction Management during the period of my doctoral studies. Special thanks to Dr. Scott Amos, Dean of the School of Technology, who understood the nature of the doctoral endeavor and allowed me a lot of flexibility in the work schedule.

Finally, and most importantly, I would like to thank Zhen Zhao, my husband and best friend, for his unwavering support and unconditioned love; and my parents, who had faith in me and provided continuous motivation.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

SYSTEMS THEORY BASED METHOD FOR CONSTRUCTION PROJECT PLANNING

By

Wen Liu

May 2008

Chair: R. Raymond Issa
Cochair: Flood Ian
Major: Design, Construction, and Planning

A variety of methods have already been developed in the past for planning and scheduling construction projects, including the CPM method, the LSM method and the simulation methods. One of the biggest problems with the existing methods is that each of them has a limited scope of application, yet most construction projects have mixed features that extend beyond the boundaries of the individual tools. Second, all of the existing methods have some fundamental limitations that restricted their abilities in the modeling of many realistic situations. Thirdly, the existing methods are becoming more and more complicated to learn and to use.

This study aims to develop a new planning and scheduling method that is simple in form, powerful in function, and universally applicable to all types of construction projects. The proposed method is based on the system theory and formalized with the DEVS (Discrete Event System Specification) specification. A set of graphical modeling elements have been designed so the user can easily utilize the full capacity of this method without any specialized knowledge in the underlying theory or any computer languages.

The proposed method has been applied to four real-world projects including a typical regular project, a typical linear project, a typical repetitive project and a project of mixed features. By comparing these models with those built with the existing methods, it was

demonstrated that the proposed method has advantages in the scope of application, accuracy of modeling, form of representation and multi-level modeling.

CHAPTER 1
INTRODUCTION

1.1 Background

Construction planning and scheduling is, in a broader sense, an endeavor of modeling and optimization: real world construction projects are represented as abstract models so that the duration, cost or some other performance measures of a certain plan can be predicted — this is *modeling*; the plan can be improved to minimize or maximize the performance measures so that the project goal can be achieved — this is *optimization*. Valid modeling is always the first step to successful optimization; and optimization is the final goal of modeling. This study focuses on the modeling of construction operations.

For different types of construction projects, the requirements on the modeling tool are different. Construction projects can be classified into three typical categories: regular projects, repetitive projects and linear projects. *Regular projects* are used to define projects in which most activities are performed only once and at one place. Construction of petroleum refineries and petrochemical plants are typical regular projects. *Repetitive projects* refer to projects that consist of many identical or similar discrete units, such as multi-family dwellings and high-rise buildings. *Linear projects* refer to projects that have a linear geometric layout, for example, highways, pipelines, and tunnels.

The requirements also vary at different management levels. At the *project level*, planning and scheduling focuses on the logic of how the project will be constructed with different trades and crews and usually the objective is to minimize the overall project cost or duration; while at the *operational level*, the focus is the collaboration among the individual resources within the crews and normally the objective is to maximize the utilization of the major resource.

A variety of planning and scheduling tools have been developed in the past, addressing the various needs of different types of construction projects and at different management levels. The most influential ones include the Critical Path Method for regular projects; the Line-of-Balance method, for repetitive projects; the Linear Scheduling Method, for linear projects; and simulation methods, primarily for operational level planning and scheduling.

One of the biggest problems with the existing modeling techniques is that each of them has a limited scope of application, yet most construction projects have mixed features that extend beyond the boundaries of the individual tools (Flood et al. 2006). So the user must either choose one technique and sacrifice the accuracy and efficiency of some parts of the schedule or use different techniques for different parts of the project.

Moreover, as been pointed out in numerous research studies and field reports, each of the existing techniques has limitations that restrict their abilities in representing some realistic situations. Though these techniques have been improved over the years, some of their limitations cannot be overcome because the problems are rooted in their most fundamental assumptions.

Thirdly, an overall trend in the development of the existing techniques is that they are becoming more and more complicated to learn and to use — which has created a large barrier to their application in the construction industry. The majority of the industry is still using the simplest form of the traditional CPM method developed over thirty years ago, despite many of its shortcomings and limitations. A lot of the more capable but more complicated methods have only been used in the academic field for research purposes.

As today's construction projects grow in scale, variety and complexity, and under the pressure of intensified market competition, there is a need for a new type of planning and scheduling technique for construction projects.

To address this need, Flood et al. (2006) has proposed a conceptual framework for a new technique that aimed to synthesize and enhance the best features of the existing ones. An example is shown in Figure 1-1.With this technique, activities can be organized into hierarchical structures, dependency relationships can be defined between activities at any levels on the hierarchy and on any attributes including time, progress, and distance. It is believed that this proposed technique is able to offer versatility in modeling all types of construction work, maintain simplicity in use, and aid visual understanding of construction projects.

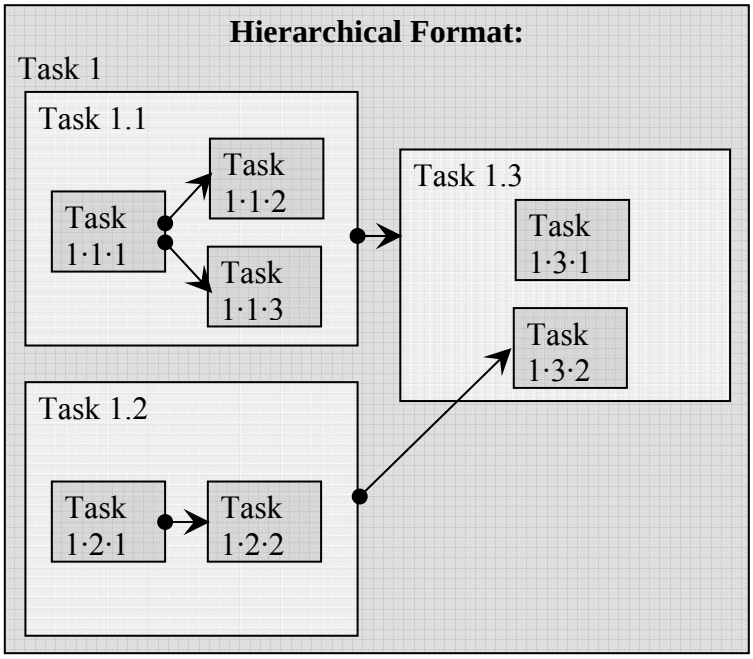


Figure 1-1. Conceptual framework of proposed universally applicable methodology. [Adapted from Flood, I., Issa, R., and Liu, W. (2006)."Rethinking the critical path method for construction project planning." CIB W107 Construction in Developing Economies International Symposium, Santiago, Chile. (Page 353, Figure 1)]

A research project has been launched by the Rinker School of Building Construction at the University of Florida to fully develop this new planning and scheduling methodology. The ultimate goal is to develop a powerful, simple-to-use, applicable-to-all planning and scheduling tool that provides full modeling, optimization and visualization functionalities.

1.2 Aim and Objectives

As part of the research project at the University of Florida, this study aims to establish a new modeling method that is simple in form, powerful in function, and universally applicable to all types of construction projects. It includes the following objectives:

- Identifying the specific needs in modeling different types of construction projects;
- Identifying the desired characteristics of an "ideal" modeling tool for construction project planning and scheduling;
- Proposing a new modeling theory that can be applied to construction project planning and scheduling.
- Developing a new modeling tool, based on the proposed theory, to satisfy the identified needs and provide the identified characteristics.

The scope of this study is limited to the modeling of construction projects — it proposes a new way of describing the construction plan and predicting the resulting performance; optimization of the plan and visualization of the process is not involved. The performance of the project is going to be measured by time, though the method can be easily extended to measure cost, space or any other resources. Also, this study focuses on modeling at the project level; operational level modeling may require further considerations for the handling of resources.

1.3 Methodology

To achieve the above aim and objectives, this study was performed in the following steps:

- Step one: Identify the strengths, weaknesses and fundamental limitations of the existing modeling methods through literature review;
- Step two: Conduct case studies with a set of representative, real-world examples to (1) identify the specific needs for modeling each type of construction projects, and (2) examine and identify more strengths, weaknesses and limitations of the existing methods in the context of realistic, complex projects;
- Step three: Summarize the desired characteristic for an ideal modeling method;
- Step four: Propose a new philosophy for the modeling of construction projects;

- Step five: Develop a new modeling method, including a complete set of modeling elements and modeling rules, based on the proposed philosophy;

- Step six: Apply the new method to the cases to verify whether it is able to deliver the desired characteristics and satisfy the identified needs for various types of construction projects.

CHAPTER 2
LITERATURE REVIEW

This chapter reviews the existing modeling techniques for construction project planning and scheduling. A general classification system for models is introduced at the beginning of the chapter. Then the Gantt chart, the CPM method, the Line-or-Balance method, the Linear Scheduling Method, and the simulation method are examined in depth. The review not only includes the basic principles of each method, but also the major research problems and different tools that are developed around the same generic concept — this is to ensure that the identified strengths reflect the latest development in the technique, and the identified limitations are fundamental for the generic method rather than specific for any particular tools.

2.1 Classification of Models

As was mentioned at the beginning of this dissertation, a plan/schedule is an abstract model of a real-world project. Before we review the modeling principles of each planning and scheduling technique, it would be beneficial to first introduce some basic concepts used in the general classification of models. Classification can often help to reveal the most fundamental assumptions and limitations of a modeling method. A model usually can be classified along three dimensions:

Deterministic vs. **stochastic**: Models that have no random input are deterministic; deterministic models always give exactly the same result for a given set of initial conditions. Stochastic models, on the other hand, operate with at least some inputs being random. In stochastic models, inputs are described as probability distributions rather than as unique values, and results are produced as ranges of values with specified confidence levels rather than as exact numbers. Construction projects are subjected to numerous uncertainties, such as weather occurrences, design changes, labor shortages, equipment and material delivery delays,

subcontractor quality, regulatory changes. Contractors often need to know how much contingency time they should include in their bids to avoid late completion penalties, and owners often need to know how "confident" they can be with the predicted project delivery date and cost. Stochastic models can help answer these questions.

Discrete vs. **continuous**: In a continuous model, the state of the system can change continuously over time; in a discrete model, change can occur only at separated points in time. The choice between discrete and continuous modeling is often a matter of accuracy. For the same activity, for example, if we need to track its accurate progress over the entire process, continuous modeling would be necessary; but if we only need to know its start time and the finish time (the intermediate progress points could be approximately interpolated when needed), then discrete modeling would be sufficient and much more efficient.

Static vs. **dynamic**: In a static model, the output at time *t* depends only upon the values of the inputs at time *t* (mathematically represented, $Y = f(x_1, x_2, ...x_n)$). By contrast, the output of a dynamic model at time *t* is dependent upon the "history" of the values of the inputs (i.e., $Y(t) = \int_0^t f(x_1(t), x_2(t), ...x_n(t))dt$) — the inputs could change over time and possibly interact with each other and with the output. Consider a model where the estimated durations of the activities are taken as the inputs. If the model can reflect the fact that the estimated durations of the activities might change as the project evolves, then it is a dynamic model.

Often times, "stochastic" is confused with "dynamic"; but they are two distinctive concepts: a stochastic model is not necessarily a dynamic model. For example, a Monte Carlo simulation model is a stochastic model, but not a dynamic model. It is stochastic because the durations of the activities can be defined as probability distributions and the output project duration are produced as probability distributions. It is NOT dynamic, however, because once the

values of the activity durations are determined by sampling, they will remain constant for that simulation run (although the values will be re-sampled for the next simulation run). Actually, time does not play a role in the Monte Carlo Simulation.

Table 2-1 shows the general classification of the major planning and scheduling techniques that are reviewed in this chapter.

Table 2-1. General Classifications of the existing planning and scheduling models

	Discrete	Continuous	Determi-nistic	Stochastic	Static	Dynamic
Bar chart	√		√		√	
CPM	√		√		√	
CPM-based stochastic methods	√			√	√	
LOB	√		√		√	
LSM	√	√	√		√	
Discrete simulation	√			√		√
Continuous simulation		√		√		√

2.2 Bar Chart

The bar chart is also referred to as the Gantt chart, in deference to Henry Gantt, who developed the initial format in 1910 (Gantt 1910). A bar chart is essentially a list of activities that are required to complete the project. These activities are arranged from top to bottom usually by their starting times. The start and end point of each activity are shown in a time grid and connected to form a bar. The length of the bar therefore, represents the duration of the activity.

The bar chart is a strong communication tool for scheduling information. Because its simple graphical representations can be quickly and easily understood without much training, it is the most commonly used means by which the project manager reports the project status to upper management and assigns works to field forces.

However, the bar chart only has very limited modeling capabilities. One of the primary shortcomings is that it does not represent the interdependency relationships among activities (Gould 2005; Hinze 2008). When the duration of one activity changes or the starting time delays, it is difficult to identify the affected activities and evaluate the impact on the project's overall completion time.

This shortcoming is overcome by the linked bar chart, where bars are connected with logic links to show activity interdependencies. The linked bar chart essentially is a time-scaled activity-on-node (AON) diagram for the CPM method.

A lot of software packages now allow the user to enter and edit data directly on a bar chart interface while the system interacts with the embedded CPM module to perform the scheduling calculation; and the generated result would be automatically converted into bar charts for display.

Because bar charts are primarily used as a format for presenting scheduling information and seldom as an independent modeling tool, it will not be further examined in this study.

2.3 Critical Path Method

2.3.1 Deterministic Critical Path Method

The ***Critical Path Method (CPM)*** was developed in the late 1950s by DuPont in an effort to create a formalized project management tool. Ever since then, it has been the most popular construction planning method among project owners, architects, engineers and contractors. A lot of owners require CPM schedules as contract submittals. Hence most commercial planning software applications are CPM-based.

CPM uses network diagrams to display both activities and their logic links. A network diagram can be constructed in two different graphical formats: the ***activity-on-node (AON)*** diagram or the ***activity-on-arrow (AOA)*** diagram. AON diagrams use nodes to symbolize the

activities and arrows to show their logical links, while AOA diagrams use arrows to represent the activities and nodes to represent events in time. Between the two, the AON diagram is more preferred by the construction industry, and has been implemented in most scheduling software applications.

The original AON diagram is limited to the Finish-to-Start type of relationship only. Its extension, the Precedence Diagramming Method (PDM) allows for four types of links, including Finish-to-Start, Finish-to-Finish, Start-to-Finish and Finish-to-Start.

The core of the CPM method is the forward and backward pass computational algorithm. For a network diagram in the AOA, AON or PDM formats, through a forward pass and a backward pass, the CPM algorithm can determine the early start time, the early finish time, the late start time, the late finish time and the floats of the activities, as well as the total duration and the critical path of the entire project.

There have been some proposed modifications to the traditional CPM method. For example, Yi and Lee (2002) have proposed using a two dimensional coordinate system, i.e., a resource coordinate and a space coordinate, to arrange CPM activities. However, the forward and backward computational algorithm has remained unchanged as the core of the CPM method.

2.3.2 Related Stochastic CPM Methods

The CPM method has several stochastic extensions. The ***Program Evaluation and Review Technique (PERT)*** was the first step to allow consideration of uncertainties with the CPM. It assumes a beta probability distribution for all activity durations, and uses the expected values ((*Most Optimistic Value* + *4*×*Most Likely value* + *Most Pessimistic Value)/6*) to find the critical path. The overall project duration is determined by summing up the expected values along the identified critical path. PERT provides a simplified analytical solution to calculate the probabilistic project duration. Its results are subjected to *merge-event bias* ─ a systematic bias to

underestimate the expected completion(David 1978). One way to eliminate the merge event bias is by using ***PNET*** (Ang et al. 1975), a method that remove paths that are highly correlated with (and thus represented by) other longer (and therefore more critical) paths in the network. The other way is by using ***Monte-Carlo simulation*** (Van Slyke 1963), a static simulation method that iteratively evaluates the CPM model using sets of random numbers as inputs.

It has also been realized that uncertainties not only exist in the durations of activities, but also in the dependency relationships. ***The Graphical Evaluation and Review Technique (GERT)*** (Prisker 1977) can model probabilistic branches. It is also able to handle the "and" and "or" logic. A GERT diagram consists of logical nodes and directed branches with two parameters: the probability that a given arc is taken and the distribution functions describing the time required by the activity. GERT uses very sophisticated mathematical probability theory to calculate the total duration of the network. However, when "EXCLUSIVE-OR" or multi-parameter branches are involved, conceptual and computational problems still exist.

The major strengths of the CPM method (including its related stochastic methods) have been well recognized:

- It is easy to learn, easy to understand, and simple to use.
- With the CPM method, it is very easy to determine the critical path of the project and the floats of the activities, which are useful in making management decisions.

On the other hand, the CPM method has been criticized for the following limitations:

- The activity is represented by two discrete points (the start and the finish point). The progress of the activity between the two points and variations in productivity cannot be indicated (Stradel and Cacha 1982; Rahbar and Rowings 1992; Harris and Ioannou, 1998).
- It assumes that if an activity starts at time *t*, it must finish at time (*t +the duration of the activity)*, which means that once the activity starts, it cannot be stopped in the middle and its duration will not change no matter what happens.
- It does not include resources in the model. However, almost any activity can be performed with a range of different types and quantities of resources; which type of the resources is

available and what quantity can be acquired ***when the activity starts*** determine the actual duration of the activity. The CPM method uses activity durations as basic input data, assuming that the preferred resources can always be acquired with adequate quantities (Birrell, 1980; Peers, 1974).

- It is not able to represent the "OR" logic in the constraints. In the CPM algorithm, the start/finish time of the successor activity is determined by the maximum value of the start/finish time of all its predecessors, assuming that all the constraints must be satisfied.

- It is cumbersome for representing repetitive activities (Chrzanowski and Johnston, 1986; Reda 1990; Harris and Ioannou, 1998).

- When applied to linear projects, the continuous activities have to be arbitrarily divided into many discrete segments (Mattila and Abraham, 1998).

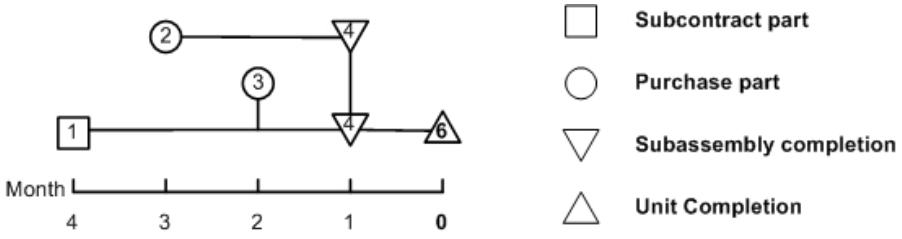
- It is not able to maintain work continuity, which is one of the major concerns in the planning and scheduling of repetitive and linear projects (Selinger 1980; Stradel and Cacha 1982; Rahbar and Rowings 1992; Harris and Ioannou, 1998).

2.3 Line-of-Balance Method

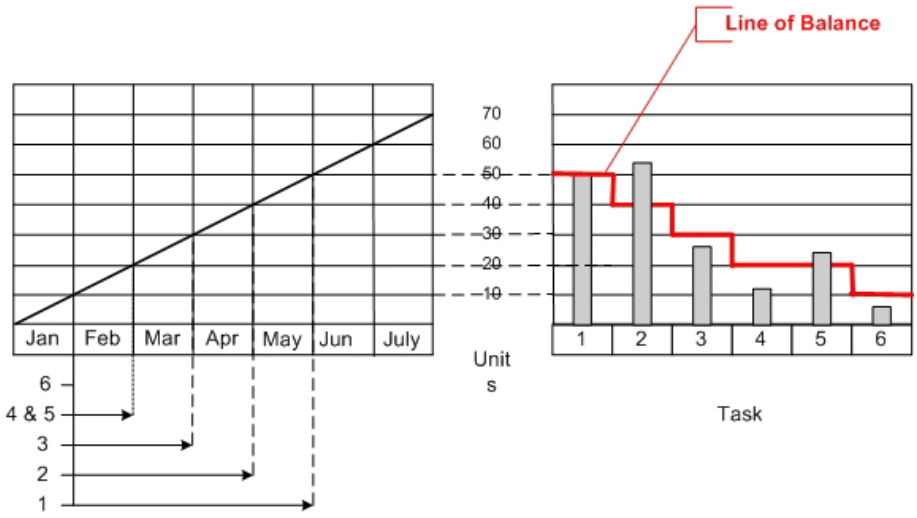
The Line-of-Balance (LOB) method was developed by the US NAVY in 1942 (Suhail and Neale 1994). Its methodology is based on the principle of the assembly line balance. It up-streams the production of each component so as to ensure that the final products can be assembled and delivered to the users on time.

There are three main elements in a LOB diagram: (1) a unit network, (2) an objective chart, and (3) a progress chart, as shown in Figure 2-1. When scheduling with the LOB method, the first step is to draw the objective chart (Figure 2-1 (b)) that reflect the planned delivery date of each unit. The next step is to schedule the unit network (Figure 2-1 (a)) with CPM to determine the time required for completing one unit, and obtain the time between each activity's finish and the unit's completion. Then the progress chart (Figure 2-1 (c)) of the project for a given date can be determined. On this chart, the horizontal scale corresponds to the tasks in the unit network. The vertical scale corresponds to the units to be completed. The line of balance shows the target progress of each task on that date. As the project progresses, histograms can be drawn on the

progress chart to show the actual progress of each task. By comparing the line of balance and the actual progress represented by the histograms, late activities can be identified and resources can be shifted from fast ones to the delayed ones.



(a) Unit Network



(b) Objective Chart (c) Progress Chart

Figure 2-1. Illustration of the LOB method. [Adapted from Johnston, D. W. (1981). "Linear scheduling method for highway construction." Journal of the Construction Division, ASCE, Vol. 107(No. CO2), 247-259. (Page 250, Figure 4)].

The following example illustrates the LOB method. According to the objective diagram in Figure 2-1 (b), Unit 1 to 10 are supposed to be delivered by Jan. 31st, which means that by Feb. 1st, all tasks in these units should have already been finished. Unit 11 to 20 are expected to be delivered by the end of February, which is one month away. The unit network in Figure 2-1 (a)

shows that in order for a unit to be delivered in next month, Task 1 to Task 5 for that unit should have been completed by the current time. Therefore, as shown in Figure 2-1 (c) , on Feb. 1st, in Unit 11 to 20, Task 1 to Task 5 are covered beneath the line of balance. This procedure continues till the desired progress for all units have been determined and shown as the line of balance.

On Feb. 1st, the actual progress have been collected from the field and drawn on the progress chart. As indicated, Task 2 and Task 5 are ahead of the schedule, Task 1 is right on schedule, while Task 3, 4 and 6 are behind the schedule.

Lumsden (1968) first modified the LOB method and applied it to building construction projects. Later, Car and Meyer (1974) used it on the scheduling of high-rise building construction. They concluded that the LOB method should only be considered as a complementary method to CPM. Suhail and Neale (1994) combined CPM and the LOB method into a new technique, which utilizes the LOB method to calculate the desired progress and CPM to level resources and calculate float times. Wang and Huang (1998) analyzed the inability of the original LOB method in controlling the interval times between adjacent activities in a repetitive unit, and presented the multistage linear scheduling (MLS) method as an improvement.

Despite its advancement through the years, the fundamental principles of the LOB method have remained unchanged: it uses an identical unit-network to represent the works required for each and every unit, and the identical unit-networks are stacked together in a fixed sequence to form the project network. By doing so, the LOB method assumes that:

- The project is composed of identical repetitive units;
- All units are processed by an identical network which consist of identical activities with identical durations;
- Only one crew is used to perform each task in all units, and all crews follow a fixed sequence to work though the units.

These assumptions and simplifications might be reasonable for manufacturing assembly lines, where the LOB method originated from, but they hardly hold true for construction projects. As a result, the LOB method only has very limited application in construction planning and scheduling, and it will not be examined further in this study.

2.4 Linear Scheduling Method

For the planning and scheduling of repetitive projects and linear projects, the Linear Scheduling Method has been long regarded much more effective than the CPM-based methods.

The fundamental concept of the Linear Scheduling method was established by Peer and Selinger (Peer 1974; Peer and Selinger 1972), in the process of analyzing parameters affecting construction time in repetitive housing project. In the Construction Planning Technique (CPT) they proposed, repetitive activities performed by one crew in different units were portrayed as one continuous line on a time-versus-units diagram, the slope of the line indicating the production rates (units per day) of the crew. Crews must keep certain distances from their preceding crews — this constraint was satisfied by adjusting the starting points and the slopes of the lines to maintain the buffers between the adjacent lines.

Peer and Selinger (1972) also argued that planning and scheduling for repetitive projects should aim to maintain work continuity and achieve maximum crew utilization, rather than simply calculate the project duration from an "arbitrarily determined" set of activity durations as the CPM method did. So they proposed that when scheduling with the CPT technique, the first step was to define "what should be made critical" in terms of cost consideration; and then to adjust the production rates of the non-critical activities to that of the chosen one. An optimum schedule, they pointed out, should have as many lines parallel (and thus critical) and continuous as possible. A mathematical algorithm was later developed (Selinger 1980) to support this method. This method actually could not guarantee the maximum overall cost efficiency (cost is

always the ultimate measure of project performance), because it might only optimize the chosen "critical activity" but not the overall project, and it completely ignored the constraint on the completion date. Nonetheless, the ideas of "balancing" and "maintaining continuity" presented in the CPT technology have remained at the heart of the LSM method.

A graphical format similar to that of the CPT was presented by O'Brien (2000) in his Vertical Production Method for high-rise building projects. He concluded that preparation works such as site work, foundations, and sub-structure, were better scheduled by the CPM, while repetitive works in typical floors could be more effectively scheduled using a floor-level versus time graph.

Earlier works presenting similar graphics include the "Time versus Distance" diagram proposed by Gorman (1972), the "Bar Chart for Repetitive Operations" by Clough and Sears (1979), and the "Trade Progress Chart" by Goldhaber, Jha, and Macedo (1977).

Johnston(1981) examined previous works and put them under one unified name — the Linear Schedule Method (LSM). He summarized the core concepts of this method and developed a detailed description of the graphical elements that can be utilized in a LSM diagram. He applied this method to several highway construction and maintenance projects, included the variation of production rates, buffers, activity interruptions, calendar considerations, and resources constraints in the linear schedule. Based on the case studies and a survey conducted among several highway contractors, he concluded that for the repetitive portions of these projects, the LSM technology offered some advantages over the CPM method: it conveyed the nature of the problem easily, provided quick solutions and intuitive presentations of the result. At the same time, he also pointed out that for discrete activities the CPM method was still necessary.

Stradal and Cacha (1982) labeled this technology the Time Space Scheduling Method (TSSM). They evaluated this method through a variety of projects that consisted of repetitive sections. Their conclusion was that this method's major limitation was in the area of complex projects and projects where activities had different alignments or measuring scales.

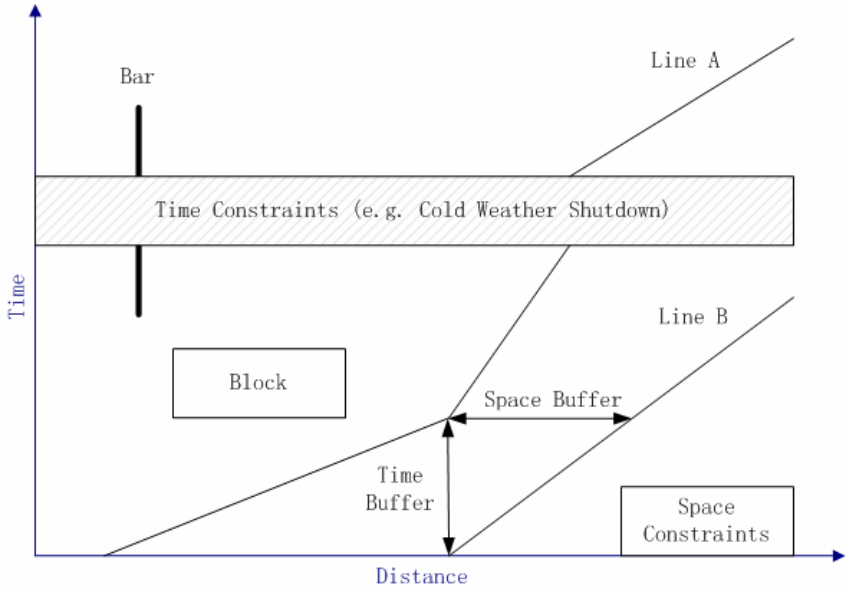


Figure 2-2. Standard format of linear scheduling diagram. [Adapted from Vorster, M. C., Beliveau, Y. J., and Bafna, T. (1992). "Linear scheduling and visualization." Transportation Research Record, 1351, 32-39. (Page 37, Figure 10)].

Another milestone in the development of the LSM method was made by Vorster et al. (1992), who standardized and improved the graphical format of the LSM. They suggested that an activity could always be represented by one of three geometrical shapes: lines, bars and blocks (See Figure 2-2). *Lines* are drawn to show the continuous movement of a crew on a particular activity throughout the job. The slope of the line represents the rate of production. *Blocks* are used to represent activities that occupy a substantial portion of space for a given period of time, for example, grading. *Bars* are used to represent activities that require work to be performed at a given location for a long period of time, such as the construction of a bridge. They also showed that time buffers, space buffers, time constraints and space constraints could be easily visualized

in a LSM diagram (See Figure 2-2). Harmelink and Rowings (1998) later divided the activities into seven smaller categories: continuous full-span lines, intermittent full-span lines, continuous partial-span lines, intermittent partial-span lines, full-span blocks, partial-span blocks, and bars.

Most of the early research on the LSM method concentrated on developing its graphical formats. The analytical ability of this technique, however, had been considered inadequate compared to the CPM method: it did not have a well-defined critical path and float times and was not as adaptable to computerization as the CPM methods. Several researchers recommended using the LSM only as a visualization tool to complement the CPM method (Chrzanowski and Johnston 1986).

During the 1990's, the analytical capabilities of the LSM method had been greatly improved. First, Harmelink and Rowings (1998) defined the term the *controlling activity path*: it is analogous to the critical path in CPM scheduling, but it is different because it allows portions of an activity to be critical, whereas the CPM only allows the entire activity to be critical. As suggested by Harmelink and Rowings, three key features, the least time interval (LT), the coincident duration, and the least distance interval (LD), were needed to identify the controlling activity path. Harris and Ioannou (1998) found that the controlling activity path in the LSM diagram switch from one activity to another where these activities are in the greatest proximity. They also pointed out that to maintain the resource continuity in repetitive and linear projects, some non-critical activities (by CPM definition) would actually become "critical" — although a delay in any of these activities will not delay the project, it will cause an interruption of resource usage. They termed such activities the *resource critical activities*. Further, Ammar and Elbeltagi (2001) developed a computerized algorithm capable of identifying the controlling activity path on the LSM diagram.

To summarize, the strengths of the LSM method mainly show in the modeling of repetitive projects and linear projects, including:

- Its ability to maintain work continuity and achieve maximum crew utilization;
- Its ability to model activities with either the discrete or continuous representation: the continuous representation uses a continuous line to show the changing production rates and the interruptions; the discrete representation uses blocks or bars which indicate only the start and finish points at each location but hide interim progresses.
- Its ability to visually convey the nature of the problem, which may provide quick solutions and intuitive presentations of the result.

On the other hand, the LSM method has the following fundamental limitations:

- It does not allow scheduling discrete and continuous activities at the same time (El-sayegh 1998).
- It can be difficult to use on projects that require a large number of trades or operations. (Sikangwan, and Tokdemir 2002).
- It cannot represent projects where activities have different alignments or measuring scales.
- It has difficulties in representing uncertainties and dynamic factors.

2.5 Simulation Methods

A simulation is an imitation of the operation of a real-world process or system over time (Banks et al. 1999). Compared to other methods, simulation has the advantages of being able to address uncertainties and dynamics in real-world systems (Zhang et al. 2002). Simulation modeling assumes that "a system can be characterized by a set of variables, with each combination of variable values representing a unique state or condition of the system" and thereby "manipulation of the variable values simulates the movement of the system from state to state (Prisker and O'Reiley 1999)". In a simulation experiment, the system's status, i.e., the variable values, evolve dynamically according to operation rules that have been pre-designed into the model. The performance of the system can be evaluated by analyzing the statistics of the

variables, or by directly observing the animated system components, as supported by some simulation software.

For a general introduction to the simulation technology, refer to Appendix A.

2.5.1 Discrete Simulation

Most simulation methods developed for construction planning and scheduling are discrete simulation methods, in which the variable values change only at discrete points in simulated time referred to as event times.

As introduced in Appendix A, the two main discrete simulation strategies are activity scanning (AS) and process interaction (PI). The third strategy, event scheduling (ES), is the underlying building block of all discrete simulation methods, and it can be combined with either the AS or the PI strategy to increase modeling flexibility.

With the AS strategy, a system is decomposed into activities whose starts are subjected to activation conditions. For simulation advancement, AS scans the entire set of activities for eligibility, choose the one whose starting conditions are satisfied and then take appropriate actions, typically including acquiring the requested resources, determining how long the activity will last, holding the acquired resources for the determined duration (when the activity starts), and releasing the resources (when the activity ends).

Three-phase AS is a modified approach that improves AS's computing efficiency. With this strategy, the activities are separated into Bs (activities that are bound to start at a predictable time), and Cs (activities that are not dependent on the simulation clock but must wait until predefined conditions can be satisfied or until requested resources are available). As the simulated time advances, only the Cs will be scanned and tested, while Bs will be immediately executed once the simulation clock reaches the scheduled time.

The other type of discrete-event simulation strategy is PI. With this strategy, the modeler needs to identify the entities (or transactions in some cases) that flow through the system and describes the life cycles for each class of them. Typically, the life cycles of the entities involves entering the system, undergoing some processing where they capture and release scarce resources, and then exit. During runtime, entities will be created and pushed though the life cycle, triggering the events on the path in sequence.

The AS strategy, particularly the three-phase AS strategy, has been the dominating modeling paradigm for construction simulation, with only rare exceptions. The following discussion will thus focus on these two paradigms.

2.5.1.1 Three-phase approach

The most influential simulation tool in the construction area is the CYCLONE and CYCLONE-based simulation tools. In 1973, Halpin developed CYCLONE (Cyclic Operation Network) and it has since become the corner stone of later simulation tools in the construction industry. CYCLONE employs the three-phase strategy, and to better suit this strategy, it extends the ACD (Activity Cycle Diagram) by differentiating two types of active states. The "normal" node in CYCLONE corresponds to the Bs in the three-phase concept, and the "conditional" node corresponds to the Cs. Special function nodes have also been included to duplicate and consolidate entities and to control the simulation run length. Thereby a CYCLONE model can be readily translated into a three-phase simulation program.

CYCLONE is purely graphical based — the diagram is the only place where everything being specified. Consequently, it is simple to use but has several important limitations: the inability to differentiate individual resources, the inability to access the dynamic state of the simulated process, and the inability to make use of the resources properties and the dynamic state to define the model behavior (Martinez 1996). There are at least four CYCLONE

implementations ─ mainframe CYCLONE, Insight, UM-CYCLONE, and Micro-CYCLONE. They have similar functions and are subjected to similar limitations.

RESCUE (Chang 1986) is also built on CYCLONE but it makes a significant improvement on it. A Process Description Language (PDL) is provided in RESCUE to supplement the CYCLONE diagram. With the PDL, each individual resource can be assigned a unique identifier, so the resources on the same path can have different activity duration probability distributions and different routing rules. Another improvement of RESCUE is that it can assemble and disassemble resources so that multiple resources can temporarily work as one group.

Another system, COOPS (Liu 1991) greatly enhances CYCLONE's user friendliness. Developed with the object-oriented programming technology, it provides an interactive interface where users can drag and drop graphical objects to quickly build a simulation model. It can track individual resources (but cannot characterize them) and report the statistics, generate and consolidate resources during the operations, and use calendars to preempt activities during work breaks.

CIPROS (Tommelein et al. 1994) also uses the object-oriented technology to enhance CYCLONE. By representing resources as a hierarchical of objects, CIPROS allows multiple real properties for resources and enables complex resource selection schemes.

STROBOSCOPE (Martinez et al. 1994) is the acronym for State and ResOurce Based Simulation of COnstruction ProcEsses. STROBOSCOPE is also based upon the three-phase strategy and the extended CYCLONE ACDs, but it has eliminated many of the simplifying assumptions in CYCLONE, and its modeling ability and flexibility far exceeds other CYCLONE-based techniques. At the conceptual level, it represents the simulation models with

an enhanced set of CYCLONE elements, which include four new nodes and four special purpose links. At a more detailed level, each network element and resource in STROBOSCOPE has attributes and methods that can be defined, dynamically accessed, and used in expressions to define the methods and attributes of other network elements and resources. It also gives the user the access to event-level modeling, so events can be used to trigger all types of dynamic changes.

STROBOSCOPE is able "to consider uncertainty in any aspect (not just duration), such as quantities of resources produced or consumed (e.g., the volume of rock resulting from a dynamite blast.)", to allow dynamic branching, to use complex resource selection schemes, to include "complex startup conditions not directly related to resource availability (e.g., do not blast rock until all crews of all trades have left the vicinity, the wiring has been inspected, and there are lee than 10 minutes left in the current shift)" etc (Martinez and Ioannou 1995).

Though very powerful and flexible, the application of STROBOSCOPE is very limited because of it complexity. The user has to have a deep understanding of the fundamental simulation principles and concepts, and invest time to learn STROBOSCOPE statements and comments to truly utilize its flexibility. Major features of the above CYCLONE-based discrete simulation techniques are shown in Table 2-2.

Despite all the advancements in the CYCLONE-based discrete simulation techniques methods, all of them rely on the ACD as the skeleton of modeling. As the name implies, a typical ACD (Activity Cycle Diagram) consists of circles of activities that are linked together at common points. This structure maps directly to the operational level systems — each circle can represent a type of equipment/crew that cycles along a fixed sequence of activities; some activities require two or more equipment/crews to cooperate, thus the circles are joined at these points.

When used to model project level systems, however, the ACD is less intuitive. At the project level, activities are not "threaded" by resources that cycle around — one resource is usually responsible for only one activity; rather, activities are linked by logical relationships. To represent the logical relationships with the ACD diagrams, a virtual entity has to be created, send through the network from the first activity to the last one, and then terminated. Since the entity has to get out of one activity before it can enters the next one, it has difficulties in representing logical relationships that are not of the "Finish-to-Start" type, and it has to be duplicated and converged whenever there are parallel activities in the network. To simplify this modeling process, Martinez has developed an add-on for STROBOSCOPE for project level simulation(Martinez 1996). Several new nodes have been provided in this add-on and the resulted diagram looks very similar to a CPM network. This add-on, however, is only able to handle the most basic finish-to-start relationships without lead and lag times. One way to represent the complicated logical relationships in STROBOSCOPE is by coding at the detailed level; but in that way these relationships will not show in the diagrams and is difficult to track.

To use the Cyclone-based method for modeling repetitive projects, each unit in the project can be represented as one entity that travels through a typical unit net-work. An example of such application can be found in a study by Lutz and Halpin (1992). In their example they assumed that: all units in the project are identical; only one crew is working on each activity; and crews of different trades follow exactly the same sequence when working though the units.

2.5.1.2 Activity-scanning approach

Shi (1999) proposed the activity-based construction (ABC) modeling and simulation method with the objective to simplify the CYCLONE simulation process. In the ABC diagrams, activities (active states) are the only modeling elements, queues (idle states) have been integrated into relevant activities as attributes, consolidating and multiplying functions are also carried out

in the activities. The ABC modeling and simulation method does not require the scheduler to differentiate between normal activities and conditional activities or to arrange entity flows among activities or entity nodes, thereby making modeling with the ABC method very similar to constructing an AON network diagram. The simulation algorithm developed for the ABC method is based on the pure Activity Scanning simulation strategy rather than the three-phase strategy. It is named the three-stage simulation algorithm and involves the following three stages: (1) select an activity for execution; (2) advance simulation; and (3) release simulation entities.

Shi later used the object-oriented technology to implement the ABC method (Shi 2000). With the object-oriented technology, the characteristics of an activity can be described using six classes of attributes including the activity duration, logical consequence, resource requirements, etc.

The major difference between the ABC method and the cyclone-based methods is that the ABC method has a simplified modeling interface which does not require the user to define the queues and put them into proper places — based on the activity scanning strategy instead of the three-phase strategy, it does not require the user to differentiate between the Bs (normal activities) and the Cs (conditional activities). However, its computing efficiency has been compromised, since it has to scan all activities every time the simulation clock advances, rather than just the “Cs”.

As for modeling abilities, the ABC method lacks the same level of flexibility and accuracy compared to the SCOBOSCOPE method (refer to Table 2-2).

Table 2-2. Comparison of the major discrete simulation techniques

		CYCLONE	RESCUE	COOPS	CIPROS	STROBO-SCOPE	ABC
Activity	Duration	Cannot be determined by resources. Cannot be dynamically determined.	Can be determined by resource type and subtype. Cannot be dynamically determined.	Can be determined by resource type. Cannot be dynamically determined.	Can be determined by resource type and subtype. Cannot be dynamically determined.	Can be determined by any variables. Can be determined dynamically.	Cannot be determined by resources. Cannot be dynamically determined.
	Resource selection	No	No	No	Yes	Yes	Yes
	Interruption	No	No	Can only be triggered by resource break-time.	No	Can be programmed flexibly	No
Dependency relationships	Start-up constraints	Resource availability only	Resource availability only	Resource availability only	Resource availability only	Both resource availability and logical dependencies	Logical dependencies only
	End branching	Probabilistic branching	Probabilistic branching	Probabilistic branching	Probabilistic branching and decision-based branching	Probabilistic branching, dynamic branching and decision-based branching	Probabilistic branching

Table 2-2. Continued

		CYCLONE	RESCUE	COOPS	CIPROS	STROBO-SCOPE	ABC
Resources	Static attributes	Type	Type and Subtype	Type	Multiple attributes	Multiple attributes	Type
	Dynamic attributes	No	No	No	No	Yes	No
	Unbalanced resource involvement	No	Yes	Yes	Yes	Yes	No
	Activity Priority	No	On defined attributes of resource	NO	NO	On any expressions	No
	Probabilistic routing	No	No	Yes	No	Yes	No
	Decision-based routing	No	No	No	No	Yes	No
	Resource calendar	No	No	No	No	Yes	No
	Assembly and disassembly	No	No	No	Yes	Yes	No
	Uncertainties in resource production and consumption	No	No	No	No	Yes	No

2.5.2 Continuous Simulation

Compared to discrete simulation, the use of continuous simulation in construction planning and scheduling has been rare.

Continuous simulation has been used in MUD (Carr 1979) and DYNASTRAT (Morua-Padilla 1986) in order to take account of the continuously changing factors that impact productivity and thereby activity durations in regular projects. MUD(Carr 1979) suggested that these factors could be calendar-dependent (DECAD) such as temperature, precipitation, and wind; or calendar-independent (INCAD) such as supervision. The sensitivity of each activity to the INCAD and DECAD variables is specified by the user and used in a formula to determine a daily correction factor in MUD. The progress of an activity each day is then calculated by applying this correction factor to its estimated standard duration. For example, if the correction factor for the first day of a 10 day activity is 0.7, then at the end of the day, there will be (10 – 1×0.7 =) 9.3 days of work remaining to be done. An activity is considered complete when there is no work remaining to be performed. MUD actually used the fixed-step continuous simulation in its computation. (See Appendix A for a brief introduction of the continuous simulation method.)

DYNASTRAT (Morua-Padilla 1986) enhanced MUD by incorporating resource availability considerations. A variety of resource allocation strategies can be selected to specify how resources are going to be allocated among competing activities. An ongoing activity may be interrupted if its resource is requested by another activity with a higher-priority. The resources assigned to an activity, along with the INCAD and DECAD factors, will determine the daily progress.

Continuous simulation has also been used to address the continuity of linear projects. Shi and Abourizk (1998) simulated a gas pipe project with SLAM II, a general-purpose simulation

tool widely used in the manufacturing industry. The continuous model they developed consisted of six separate networks each representing the birth-to-finish process of one activity, and six subroutines written in FORTRAN each responsible for updating the production rate and the progress status of a corresponding activity. The model also included a number of resource describing nodes and status detecting nodes. The produced results were presented in a LSM velocity format. They also built a discrete model for the same pipeline project for comparison. In this discrete model, the activities were progressed at the completion of each operation cycle. In conclusion, they commented that even though the discrete model was easier to construct with the SLAM elements, it would require a lot more data collecting and analysis at a lower operational level.

2.6 Other Methods

Models for construction planning and scheduling can also be formulated with other techniques including linear programming (Reda 1990), dynamic programming (El-Rayes and Moselhi 2001; Senouci and Eldin 1996), integer programming (Liu et al. 1995), and optimal control theory (1986). The purposes of most of these models are to quickly find a satisfactory solution to improve a certain project performance measure, rather than to accurately represent the project. A construction planning problem usually has to be simplified a lot to fit the assumptions of these methods, and sophisticated mathematical knowledge and skills are often required. So these models will not be discussed in detail in this study.

CHAPTER 3
CASE STUDIES WITH EXISTING METHODS

In Chapter 2, we have reviewed the major existing planning and scheduling techniques. In Chapter 3, we will select proper tools from these techniques and apply them to four case studies, each representing a distinct type of construction projects, including a typical regular construction project, a typical repetitive construction project, a typical linear construction project, and a project of mixed features. By doing so, two objectives can be accomplished: (1) the specific requirements for modeling each typical type of construction projects can be identified, and (2) the strengths and limitations of each planning and scheduling method in addressing these needs can be identified.

Real-world construction projects are used in these case studies. The first project is the construction of a medium-rise building. This project has been divided into three separate case studies: the construction of the foundational part of the building is studied as an example of the typical regular project; the construction of the upper-structure, an example of the typical repetitive project; and the whole process, an example of projects with mixed features. The second project is the construction of a gas pipeline, which is examined as a typical linear project. The discussion on the modeling of linear projects is further broadened with a short study on the third project, which involves the construction of multiple utility lines.

These case studies together cover the full spectrum of typical construction project categories. The problems presented in these case studies are very representative of the realistic, complex situations that could be encountered on the construction jobsite.

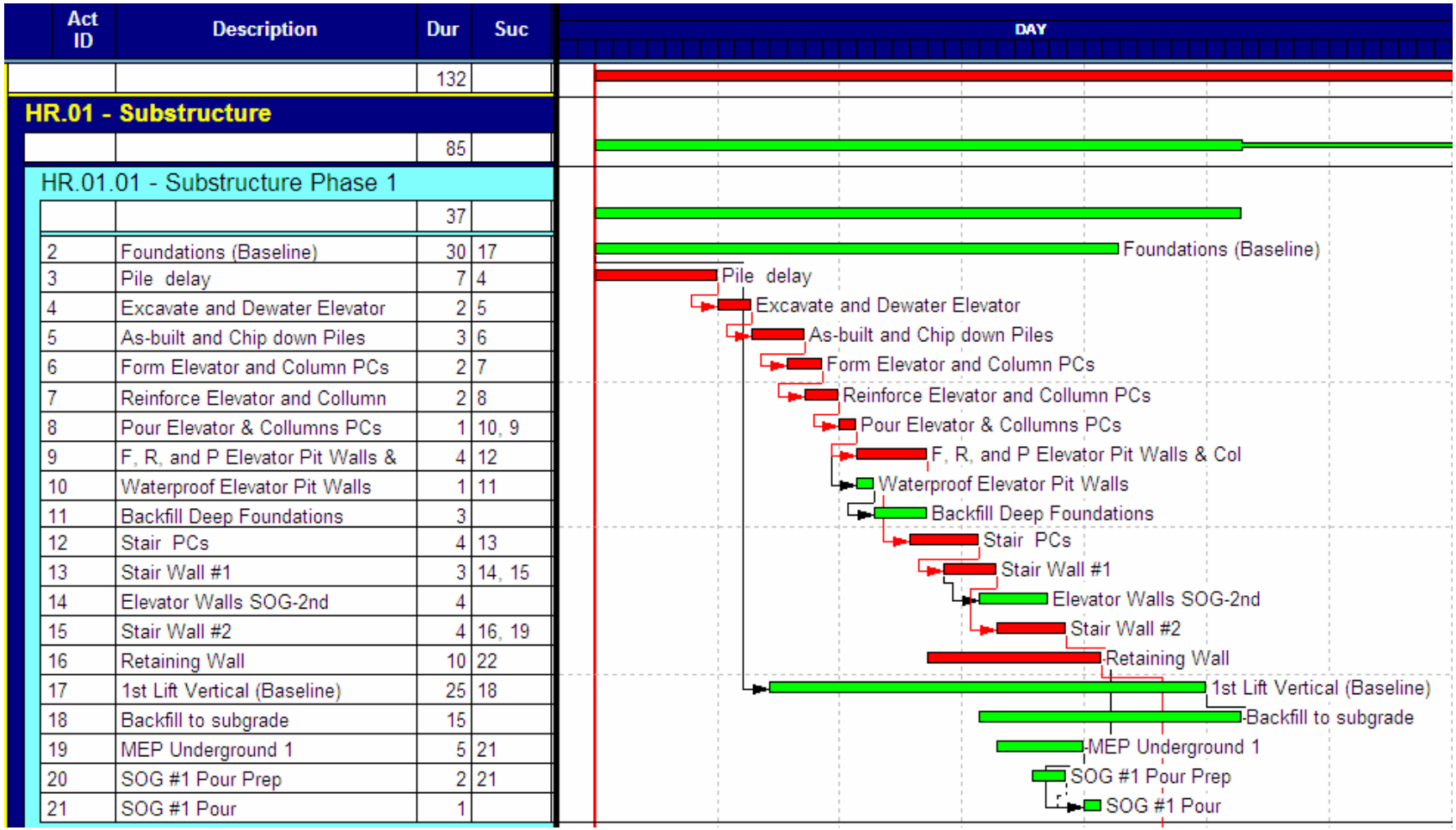


Figure 3-1. The CPM model for Case One

3.1 Case One: A Typical Regular Project

The data used in Case One, Three and Four came from a work report on the planning and scheduling of the structural work in a 14-level condominium at southern Florida (Hughes 2005). The subcontractor, hereto referred to as Company A, was responsible for the construction of the framing system of the foundation and the upper structure of the building. In the work report, Hughes recorded the construction process in detail, presented the CPM schedule prepared by Company A, and reported the requirements and concerns of the management members on various planning and scheduling issues from their individual perspectives.

Case One examines the first phase of Company A's work — construction of the foundation for the building. This is a typical, regular construction project, which can be modeled with the CPM method and the discrete simulation method. The LSM and the LOB method are not applicable to regular construction projects and thus will not be discussed in this case study.

3.1.1 The CPM Model

The original schedule prepared by Company A used the CPM method. The schedule shown in Figure 3-1 is a revised version. The obvious errors in the original schedule have been removed, and the format has been changed for better clarity. Note that it is possible to further improve this schedule, but the changes would involve a lot of tricky "workarounds". As the purpose of this study is to examine what the CPM method is able to do with realistic and moderate input, further changes are not going to be made.

It also needs to be pointed out is that although this example was developed with the deterministic CPM method, the discussions is not limited thereby; rather, it is targeted at the most fundamental limitations and assumptions of the generic method, including both

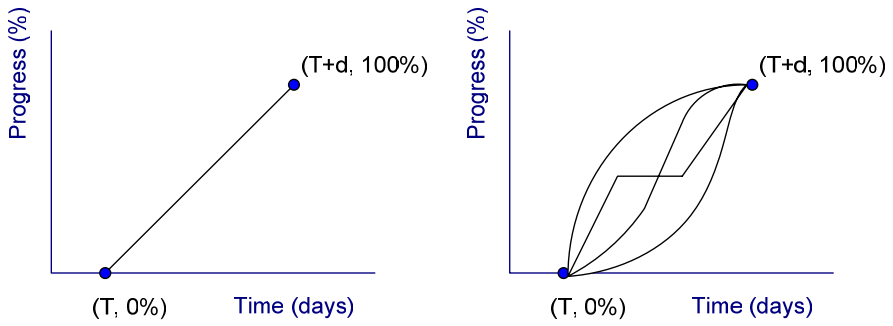
the deterministic and the stochastic CPM. The discussion includes four parts: modeling of activities, modeling of dependency relationships, modeling of resources, and modeling of other factors.

3.1.1.1 Modeling of activities

In the CPM method, activities are modeled as two discrete points with a fixed duration *d* in between. It assumes that an activity starting at time *T* will definitely end at time (*T+d)*. Problems with this approach have been identified as following:

Durations of the activities cannot be dynamically determined. Durations of the activities have to be entered as real numbers or probabilistic distributions. This means that the values of the durations cannot be dynamically determined according to real-time situations, such as what resources are available when the activity starts, the weather and the site conditions at that time, and other activities going on in adjacent areas. This often requires the scheduler to determine a value in advance with many arbitrary assumptions.

The intermediate progress of the activities cannot be accurately represented. Every activity in the real world is a continuous process, progressing gradually minute by minute, day by day. With the CPM method, this continuous process has to be reduced to two discrete points — a start point and a finish point. When the production rate (or the expected value of the production rate) of the activity is constant, this simplification is adequate and computationally efficient. But, if the production rate varies throughout the process, such representation will lead to loss of information, which is illustrated by Figure 3-2.



(a) Constant Production Rate (b) Variable Production Rate

Figure 3-2. Discrete representation of an activity

In Figure 3-2, the horizontal axes represent time *t* and the vertical axes represent progress *p*. Suppose that the estimated duration for the activity is *d*, and the activity starts at time *T*. In Figure 3-2(a), the production rate is constant. Given a starting point (T, 0%) and an end point (T+ d, 100%), any intermediate progress at time *(T+t)* (0<t<d) can be determined — two discrete points can always determine a unique continuous straight line. But if the production rate varies, two points are not sufficient to represent the whole process — there are infinite numbers of possibilities about how the progress line could pass through the start point and the end point, as shown in Figure 3-2(b).

From the project manager's point of view, the intermediate progress of most activities in this project was not of much concern; the project manager only needed to plan and control the dates on which these activities start and finish. The details could be left to the subcontractors and the foremen. But there were a few activities that lasted too long, sat on the critical path and required tighter control. In the schedule shown in Figure 3-1, Activity Foundation had an estimated duration of 30 days. The project manager needed to check the progress of this activity every two or three days and take immediate actions if it deviated from the schedule. From his experience, the project manager knew

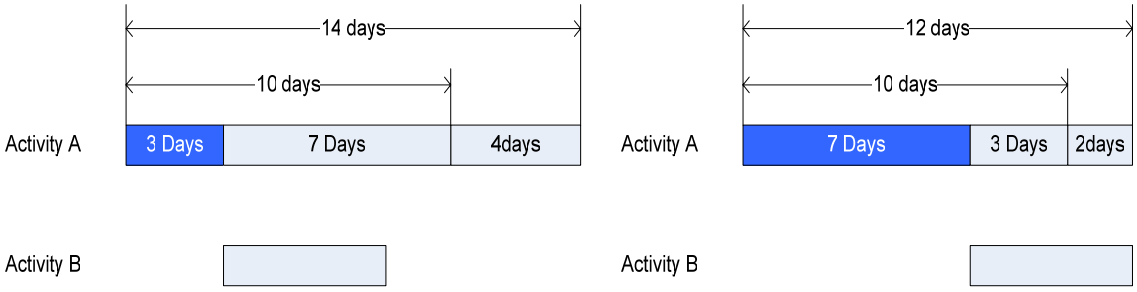
that production would gradually speed up and it was quite normal for the activity to progress slower at the beginning. Still, the project manager would like to have some quantitative criteria to determine whether the progress was just "reasonably" behind or there were problems that needed immediate correction. The CPM schedule did not help setting the intermediate goals.

Interruptions during the middle of the progress cannot be represented. The CPM method assumes that once the activity start, it will not stop until 100% finished. In reality, the activity could be interrupted in many situations. For example, whenever the temperature drops to below 40°F, any activity involving concrete placement has to stop. When such events would occur and how long they would last are unpredictable.

Certainly we could roughly estimate the likelihood of such events and use a probabilistic distribution to account for the impacts of possible interruptions. But this solution is not able to capture the cause-and-effect relationships between the events and the change of the durations; moreover, it might over-estimate the total amount of resources used by the activity, for it does not separate the actual time that the resources are working on the activity and the interruption time during which the resources can be temporarily released.

Adjustments of activity durations during the middle of the progress cannot be represented. Various types of events could also cause the duration of the activity (here it refers to the actual duration excluding the interruptions) to lengthen or shorten during the middle of its process. For instance, as shown in Figure 3-3(a), the estimated duration of Activity A was 10 days. On the third day, Activity B started, interfered with Activity A and caused its remaining duration of 7 days to extend to 11 days. With the CPM method,

there is no way to model this adjustment. We cannot simply add 4 days to Activity A's estimated duration to account for this possibility. If Activity B had started on the seventh day, for instance, Activity A would experience less interference and suffer less extension (as illustrated in Figure 3-3(b)). The effect has to be determined "dynamically" at the point where the event occurs, during the middle of the activity rather than at the beginning of the project or the activity.



(a) Activity B starts on the 3rd day (b) Activity B starts on the 7th day

Figure 3-3. Adjustments of Activity A's duration

3.1.1.2 Modeling of dependency relationships

There are four types of dependency relationships that can be defined with the CPM method: the Finish-to-Start relationship, the Start-to-Start relationship, the Finish-to-Finish relationships and the Start-to-Finish relationship, with possible lead and lag times. These dependency relationships have to be defined either on the start point or the finish point of the activities, and the leads and lags have to be measured in time units, i.e., hours, days, weeks, etc. The following problems have been identified:

Non-time-based dependency relationships cannot be accurately represented. Non-time-based dependency relationships are everywhere in construction projects. In this project, it was required that the vertical lift on the first floor could not start until the foundation was at least 1/3 finished — a constraint based on the progress of Activity

Foundation. Since the CPM method can only handle time-based dependency relationships, this progress-based constraint had to be converted to a Start-to-Start relationship with a 10 days' lag time (shown in Figure 3-4). The lag time (30 days × 1/3 = 10 days) was calculated based on the assumption that Activity Foundation would take 30 days to complete and its production rate was constant.

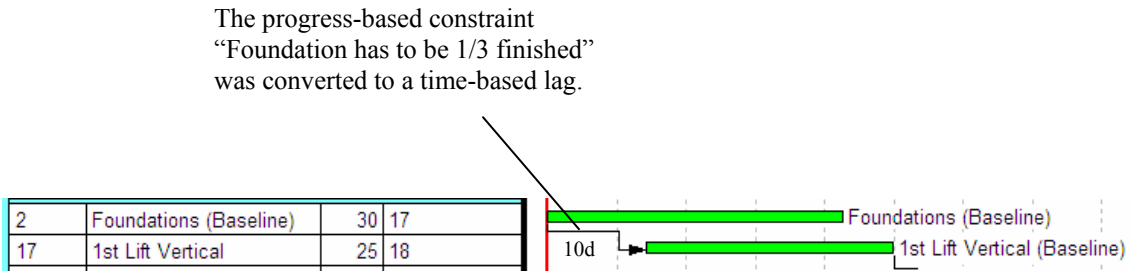


Figure 3-4. The CPM representation of a non-time-based dependency relationship

This duration-based dependency relationship is not equivalent to the original progress-based dependency relationship. Consider this scenario: 10 days after Activity Foundation started, only 1/5 of the foundation had been finished. As a matter of fact, Activity 1st Lift Vertical could not start since the progress of Activity Foundation had not reached 1/3. However, according to the CPM schedule, Activity 1st Lift Vertical would start anyway for the 10 day's lag time had passed. A similar problem would occur had Activity Foundation progressed faster than the assumed speed.

For the incoming dependency relationships, only "AND" logic can be represented. In an AON network diagram, every arrow coming into an activity node represents a constraint on the start or finish of that activity. The CPM method evaluates the incoming constraints by combining all of them with "AND"— only when all of the constraints have been satisfied, the activity can start or finish. "OR" type logic cannot be

handled. The CPM-based GERT technique is able to represent the "EXCLUSIVE OR" but not the "INCLUSIVE OR" logic.

For the outgoing dependency relationships, all branches will be fulfilled. The links coming out from an activity node represent the constraints that the activity imposes on its succeeding activities. When the activity has been finished (or started as in the SS or SF type of relationships), these constraints are satisfied and the succeeding activities may proceed. Oftentimes, however, not all of the succeeding activities are meant to be executed. If they are exclusive alternatives, only one of them shall. In such cases, decisions must be made on which succeeding activity to choose. The CPM method is not able to model this situation; all outgoing branches will be fulfilled. The GERT technique can choose one branch randomly based on the probabilities assigned to the branches (a feature called *Probabilistic Branching*), but is not able to make the decision according to the values of user-defined variables or expressions (which is referred to as *Decision-based Branching* in this study).

3.1.1.3 Modeling of resources

For most construction projects, resources are always the critical constraints on the performance of the project. The durations of the activities are dependent upon the type and amount of the resources available for the activity. The dependency relationships between the activities are also influenced by resource assignment: when two activities share the same resource, they cannot run in parallel, but have to be sequenced linearly.

The CPM method does not include resources as the basic input. Durations of activities and dependency relationships are entered directly. The underlying resource constraints that determine activity durations and dependency relationships are not represented. Consequently, it is difficult to see and understand the assumptions and

limitations used in the schedule, and therefore difficult to detect scheduling mistakes and generate alternative schedules.

3.1.1.4 Modeling of other factors

The CPM method is not able to represent factors such as temperature, precipitation, wind and supervision, which may have great impacts on project schedules.

3.1.2 The STROBOSCOPE Simulation Model

The second model for this case study, which is shown in Figure 3-5, is developed with STROBOSCOPE. STROBOSCOPE was selected because it is widely recognized as the most powerful and flexible discrete simulation tool for construction operations and projects.

The STROBOSCOPE network shown in Figure 3-5 may look very similar to the CPM Activity-on-Node network shown in Figure 3-1, but there is one fundamental difference: in the CPM network, the links represent the start/finish constraints one node imposed upon another; in the STROBOSCOPE ACD diagram, the links represent the directions that the resources travel through the activities.

In Figure 3-5, a virtual resource — token, is generated by the Queue node "TokenGen" at the left end of the network, and then sent through the nodes as directed by the links. The names of the links "T_i (i = 1 to 29)" indicate that only resources of type "T" (Token) are allowed to flow through these links (i in the names does not suggest any sense of sequence).

The STROBOSCOPE model is also examined in four aspects, including: modeling of activities, modeling of dependency relationships, modeling of resources and modeling of other factors.

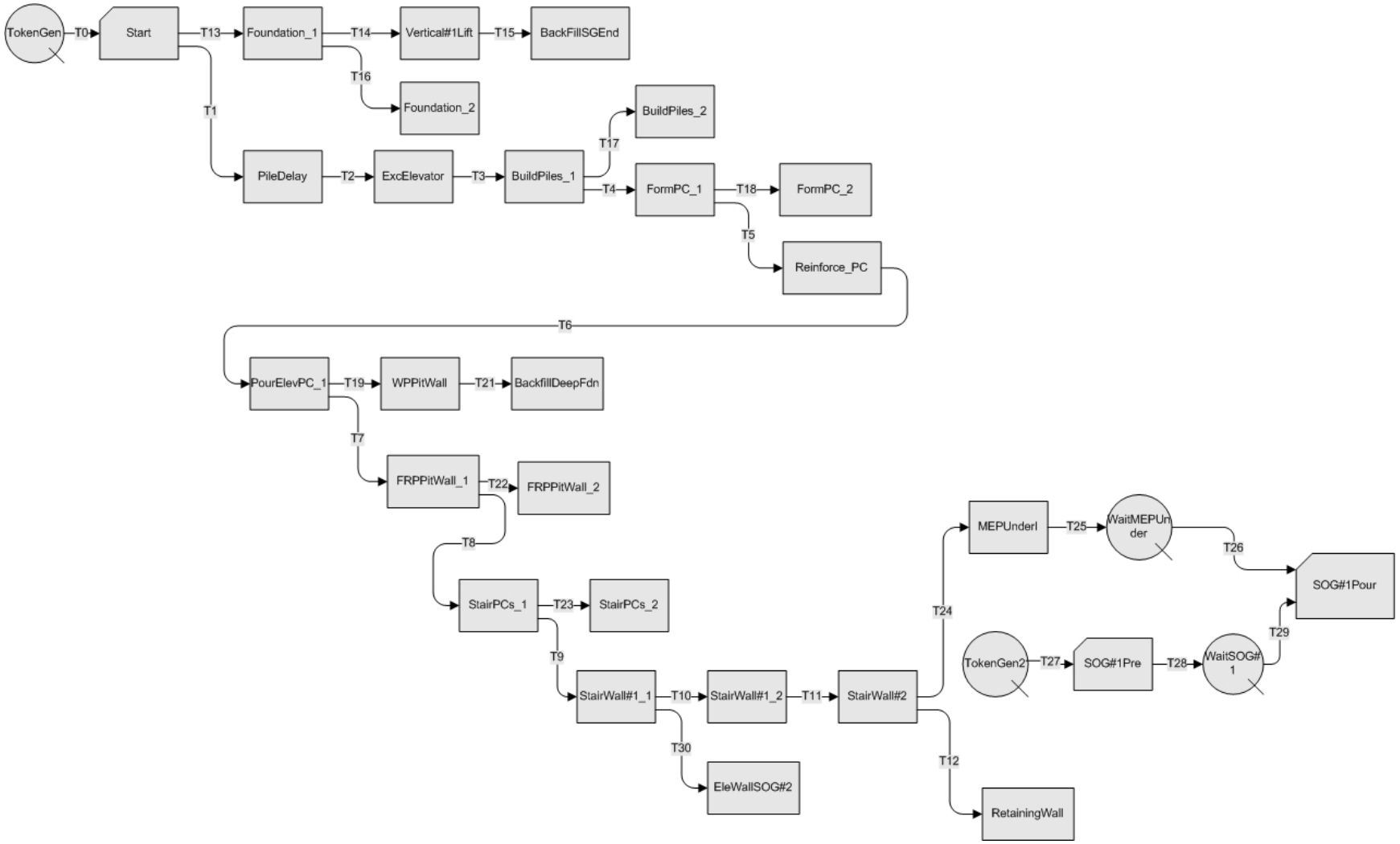


Figure 3-5. The STROBOSCOPE model for Case One

3.2.1.1 Modeling of activities

It is able to dynamically determine the durations of the activities. With STROBOSCOPE the duration of the activity can be defined as an expression of system variables and user-defined variables. The value of the expression is determined at the point when the activity starts, so it can reflect which type of resource has actually been acquired, the "current" status of weather, site conditions and impacts of any other dynamic factors.

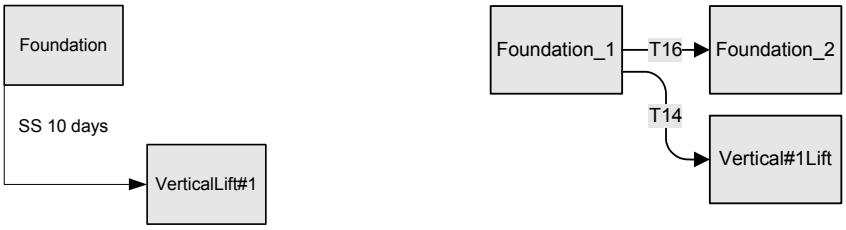
It is unable to represent the intermediate progress of the activities. In discrete simulation, the clock advances from one event to the next. With the CYCLONE-based simulation methods, each activity is essentially modeled as a pair of events: a start event at time *T* and a finish event at time *(T + d)* (*d* is the duration of the activity). The state of the activity does not change between time *T* and time *(T + d)*, and the intermediate progress of the activity cannot be accurately represented.

It is very difficult to represent interruptions and adjustment of activity durations during the middle of activity. With STROBOSCOPE, as with any other CYCLONE-based simulation methods, the start event of an activity at time *T* will automatically schedule a finish event of that activity at time (*T*+*d*); there is no way to remove or re-schedule a finish event except with very complicated code writign at the event level.

3.2.2.2 Modeling of dependency relationships

SS, SF and FF type of dependency relationships cannot be directly represented. In STROBOSCOPE, as in any other ACD-based simulation methods, the activities are activated by the arrival of required resources. The resources have to finish one activity before they can enter the next one. So naturally, STROBOSCOPE is most

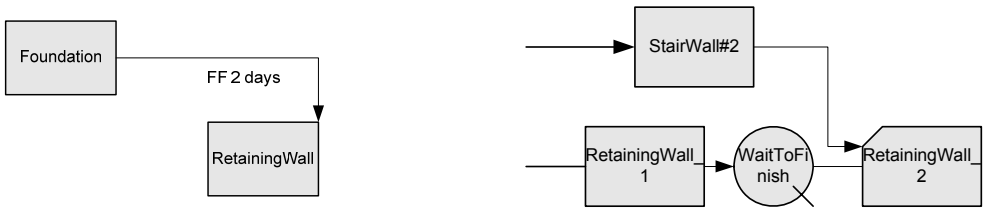
efficient with the Finish-to-Start type of dependency relationships. Dependency relationships of the other types cannot be directly represented. For example, Activity Vertical#1Lift cannot start until 10 days after Activity Foundation has started. The CPM method can simply represent this constraint as a Start-to-Start relationship with a 10 days' lead time, as shown in Figure 3-6. With STROBOSCOPE, the first activity has to be divided into two nodes — Activity Foundation_1 with a duration of 10 days (the length of the lead time) and Activity Foundation_2 of 20 days (the original duration 30 days minus the lead time 10 days) — in order to allow the token to flow out during the middle of the predecessor activity to activate the successor activity.



(a) CPM Model (b) STROBOSCOPE Model

Figure 3-6. Representation of an SS dependency relationship

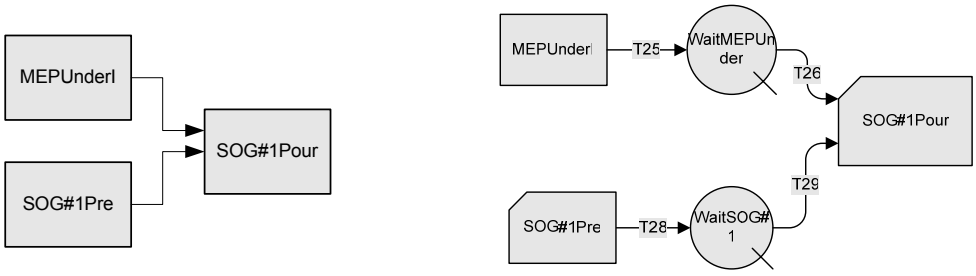
It is even more difficult to represent SF and FF types of dependency relationships. An example of the FF dependency relationships is shown in Figure 3-7. With the SCOBOSTROPE method, the successor activity needs to be represented with three different types of nodes.



(a) CPM Model (b) Time Elapsed in the Current State

Figure 3-7. Representation of an FF dependency relationship

It is able to model compound startup constraints, but not very easily and quickly. With the CYCLONE ACD diagrams, the startup of the activity is controlled by the arrival of the required resources/tokens. An activity has to obtain resources or tokens from all of its preceding queues before it can attempt to start. This means that in the ACD diagram all incoming links to one activity are combined with "AND" to form the startup constraint, just as in the CPM method. However, one would find that the CYCLONE ACD diagram requires some extra nodes when used for this purpose.



(a) CPM Model (b) STROBOSCOPE Model

Figure 3-8. Representation of the "AND" logic

Figure 3-8 shows how the CPM method and the STROBOSCOPE method are used to describe: Activity SOG#1Pour cannot start until both Activity MEPUnderI and Activity SOG#1Pre are finished. The former only uses one type of node, whereas the latter uses three different types of nodes and needs to define and initiate the tokens that flow through the nodes.

One advantage of STROBOSCOPE over the other CYCLONE-based simulation tools is that it provides a second way to represent the startup constraints. The "OR" logic and more complicated compound logic can be defined in the Semaphore of the activity. The Semaphore is a logical expression that must return TRUE before the activity can

attempt to start and acquire resources. A Semaphore may contain any system or user-defined variables, and may include multiple logical statements linked with "AND" and "OR". However, one must be very familiar with STROBOSCOPE variables and statements in order to define the Semaphore properly. Moreover, the compound startup constraints written in the Semaphore cannot be directly read from the ACD diagram.

It is not able to represent non-time-based dependency relationships. As STROBOSCOPE does not model the intermediate progress of the activity, progress-based dependency relationships have to be converted to time-based dependency relationships, which are not equivalent conversions most of the times, as discussed in 3.1.1.

It is able to represent probabilistic branching and decision-based branching. When an activity is finished, the released resources/tokens can be routed via the Fork or the Dynafork node, which is able to decide which successor activity the resource will go to next using either probabilistic branching or decision-based branching.

3.2.2.3 Modeling of resources

It is able to realistically represent various types of resources. In STROBOSCOPE, resources can be modeled either as "Generic Resources", which are simply measured by the quantity, or "Characterized Resources" which have a set of properties. The properties of the characterized resources can be used to carry a lot of information and thus increase the flexibility of modeling. For example, if different dozers need different length of time to accomplish an activity "Backfill", the resource "Dozer" should be defined as a characterized resource which has a property "Backfill Duration". Each dozer then can be assigned different values for this property and spend a different length of time in the same activity "Backfill". A set of resources can also be bundled

together to form a “Compound Resource”, and the compound resource can be dissembled as required.

It is able to represent activity priorities for shared resources, but the activity cannot have different priorities for different resources. When one resource is shared among multiple activities, it is necessary to specify the priorities of these activities if they are requesting the resource simultaneously. STROBOSCOPE allows specifying the priority as one attribute of the activity; however, it assumes that the activity has the same priority for all the resources that it shares with others.

It is able to model various resource selection schemes, but the resources have to be of the same type. When one activity could be done with several alternative resources, it is necessary to specify the priority of these resources. In STROBOSCOPE, the activity can select the resources from its preceding queue, in which the resources are sorted according to a specified rule (e.g., first-in-first-out, last-in-first-out) or on the value of an evaluation expression. But the selection can only be performed among resources of the same type because the sorting has to be done in a queue and one queue can only hold resources of the same type.

3.2.2.4 Modeling of other factors

Temperature, precipitation, wind and other dynamically changing factors can be modeled with separate sub-networks in STROBOSCOPE. The values of these factors may be accessed by the activities and thereby impact the attributes and behaviors of the activities.

As can be seen from the analysis above, the simulation method has advantages over the CPM method in terms of modeling abilities. However, there are some major obstacles in the application of the simulation method, including:

It is difficult to learn and to use. As can be seen from this example, using the simulation method to model a construction project requires significant amount of time, effort and skill. There is a lack of direct correspondence between the elements and rules in the simulation tools and the real construction project: often times, the user has to assemble various types of nodes and links to represent a single activity or constraint, and do fairly amount of code writing. This demands a deep understanding of the fundamental simulation theories, familiarity with the particular variables, functions, syntax of the specific programming language, and also creativity in figuring out "workarounds" for many situations.

It does not facilitate understanding and communication. The complexity of the simulation method makes it very difficult to use it as a communication tool on the construction jobsite. A STROBOSCOPE model as shown in Figure 3-5 does not resemble the natural way that people would describe this project. Moreover, a lot of information (such as complex startup constraints written in the Semaphores) cannot be represented graphically in the ACD diagram, so one needs to read the code to find out the whole and accurate meaning of the model.

3.1.3 Summary of Modeling Regular Projects

Based on Case Study One, we can summarize the most important requirements for modeling typical regular projects. Table 3-1 shows the abilities of the CPM method and the STROBOSCOPE method in addressing these requirements.

Table 3-1. Modeling requirements for regular projects

		CPM	STROBOSCOPE
Activities	Duration	Only real numbers or probabilistic distributions.	Both deterministic and probabilistic. The values are dynamically determined. Can be resource driven.
	Progress curve	No	No
	Interruption	No	No
	Duration adjustment	No	No
Dependency Relationships	FS,SS,FF, SF relationships	Yes	Only FS relationships can be directly represented.
	Non-time-based dependencies	No	No
	Compound constraints	Only "AND" logic.	Yes, but need extra nodes or coding.
	Branching	No	Support both probabilistic and decision-based branching.
Resources	Direct representation	No	Yes, include generic, characterized and compound resources.
	Activity selecting resources	No	Yes, but activities can only select from resources of the same type.
	Resource selecting activities	No	Yes, but an activity can only be assigned one priority value for all the resources it shared with other activities.
Environmental factors		No	Yes, but these factors need to be modeled with separate sub-networks.
General		Easy to learn, to use and to understand.	Difficult to learn, to use and understand.

3.2 Case Two: A Typical Linear Project

The second case study is a typical linear project, which contains only continuous activities and continuous dependency relationships — buffers. The project data came from a real-world example provided by Shi and Abourizk (1998) in their study on the modeling of pipeline construction projects. The project was to install a 10 km long gas line in a natural bush area. As shown in Table 3-2, it consisted of six tasks: right-of-way, stringing, welding, trenching, lowering-in and backfilling. The resources available to this project are shown in Table 3-3.

The Right-of-way was divided into two sections: Section 1 of 6000m length which was in a thickly wooded area and Section 2 of the remaining 4000m length that was easier to clear. The two sections were both constructed by Crew A.

Stringing used Crew B, and the productivity did not vary too much throughout the whole length.

Welding was assigned to Crew C. As there were many uncertainties that could influence welding, the production rate of this activity followed a uniform distribution.

Trenching required different equipment under different geotechnical conditions. The whole site was therefore divided into 4 sections for trenching. The first section, 3000m long, was excavated with two backhoes; the second section, a 500m valley, only used one backhoe; the third, 4000m, used a trencher. The fourth section, 2500m long, had similar soil conditions as the third, but once in a while (approximately 10% of the time) rocks were encountered and a backhoe was needed to assist the trencher.

Lowering, shared Crew B with Stringing.

Backfill, only used one dozer.

Table 3-2. Description of the pipeline construction project (Shi and Abourizk 1998)

Activity	Number. of sections	Length of section (m)	Advance rate (m/h)	Crew	Distance buffer (m)
Right-of-way	2	Section 1: 6000	80	Crew A	100
		Section 2: 4000	100		
Stringing	1	10,000	400	Crew B	100
Welding	1	10,000	U(120,150)*	Crew C	100
Trenching	4	Section1: 3000	T(50,70,100)*	2 backhoes	50
		Section 2: 500	50	1 backhoe	
		Section 3: 4000	100	1 trencher	
		Section 4: 2500	U(80,100)*	1 trencher + 1backhoe	
Lowering-in	1	10,000	400	Crew B	20
Backfilling	1	10,000	200	1 dozer	

* Uniform distribution.
* Triangle distribution.

Table 3-3. Resources available to the pipeline construction project

	Quantity
Crew A	1
Crew B	1
Crew C	1
Backhoe	2
Trencher	1
Dozer	1

A distance buffer had to be maintained between any two activities. When the buffer was violated, the succeeding activity stopped or slowed down. The last column of Table 3-2 shows the required distance between an activity and its succeeding activity.

Though the construction of this project is not complex technically, it is not easy to find a proper tool to accurately model it. The CPM method and the CYCLONE-based simulation methods have been excluded because they are not able to represent continuous activities and continuous dependency relationships (buffers). Certainly a continuous activity could be divided into many discrete segments and a continuous buffer could be reduced to point-to-point dependency relationships at the start and the end of the segments, this approach is complicated, and more importantly, not accurate.

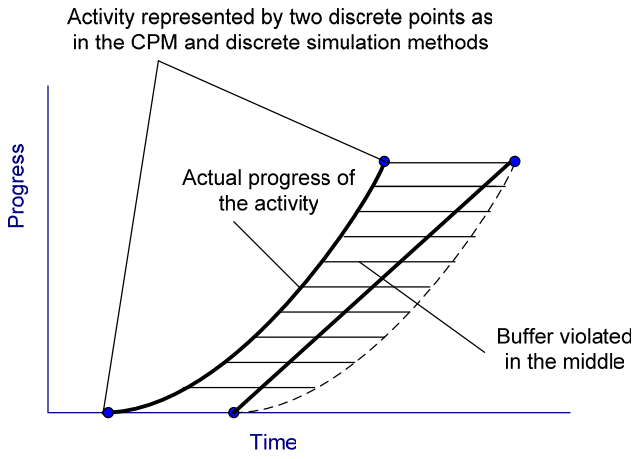


Figure 3-9. Problems of discretizing continuous activities and buffers

As shown in Figure 3-9, the succeeding activity might run into the preceding activity in the middle even when there is enough distance between their start points and end points.

In fact, the LSM method is not adequate for this project either because it is not able to represent probabilistic production rates. Nevertheless, as it is widely considered the

best scheduling tool for linear projects, we will still apply it here, ignoring all of the probabilistic factors. We will also examine a SLAM II model built for this project by Shi and Abourizk (1998), which is the only continuous simulation model developed for construction projects that have been published so far.

3.2.1 The LSM Model

With the LSM method, the schedule can be quickly laid-out on the location-time diagram in a two-step manner: first, draw the individual progress of each crew as if it were working independently without any constraints; second, arrange the progress lines from left to right, making sure that the buffers will not be violated. Figure 3-10 shows the resulting LSM schedule. The total project duration is 211 hours.

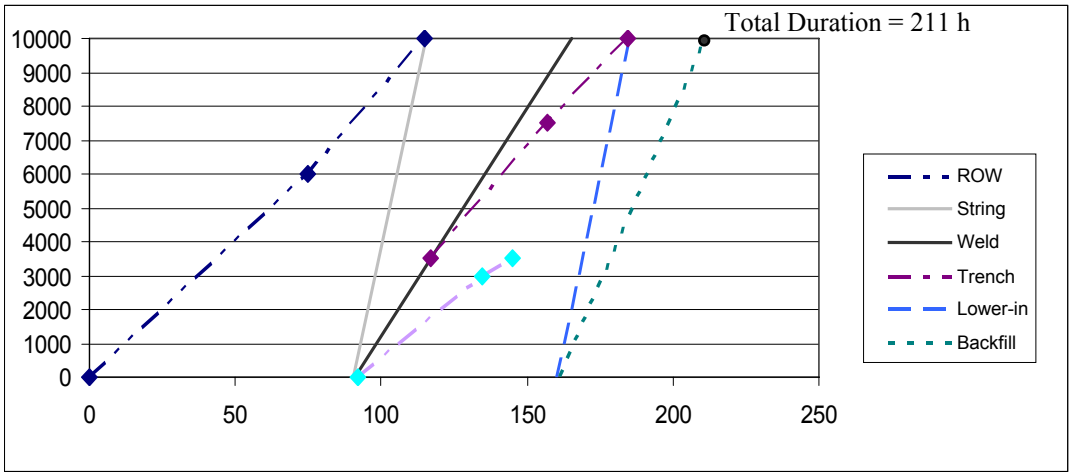


Figure 3-10. The LSM model for Case Two

If the project has a tight deadline, the above schedule may need to be crashed. The project manager may choose to start Stringing earlier (at the 48th hour, for example) so that all its succeeding activities can start earlier. When Stringing runs into the Right-of-Way, the stringing crew may leave the activity for a certain period of time and relocate to work on another activity. The break should be long enough to allow the crew to finish a

fair amount of work on the other activity — otherwise the time and cost involved in relocating cannot be justified. As the result of this change, the total project duration has been reduced to 190 hours, as shown in Figure 3-11.

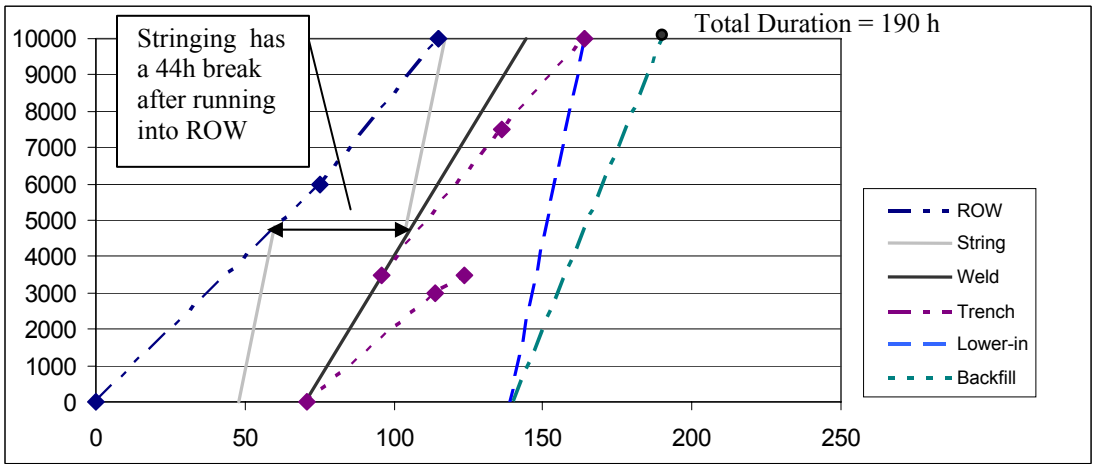


Figure 3-11. The LSM model for Case Two with a break

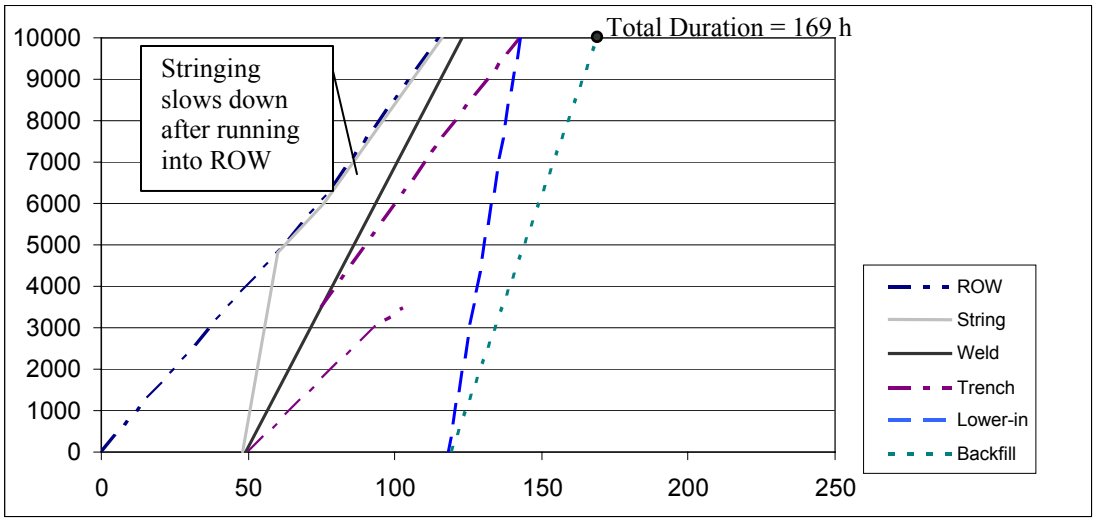


Figure 3-12. The LSM model for Case Two with a slow-down

The project manager may also choose to have the stringing crew continue its work after it has been blocked, but at a reduced pace following the Right-of-Way, as shown in Figure 3-12. Because the stringing crew is not working at its full capacity, this solution is

not very efficient cost-wise, but the project duration can be further reduced to 169 hours. The same strategies can be applied to other activities if it is necessary to further crash the schedule.

The advantages and limitations of the LSM method in the planning and scheduling of typical linear projects can be examined from the following aspects: modeling of activities, modeling of dependency relationships, modeling of resources and modeling of other factors.

3.2.1.1 Modeling of activities

It is able to show the continuous progress of the activities at any point in time. One major advantage of the LSM method when applied to the linear construction projects is that it is able to show the progress of the activities at any point in time, not only at the start and the finish point as in the CPM or discrete simulation methods.

It shows the positions of the activities as they progress. A big difference between the regular projects and the linear projects is: the activities in the regular projects usually are done at a relatively small space, such as a room or an area; while the activities in the linear projects are spread along long lines throughout the whole project. Crews working on the regular projects change places only when they begin new activities, while crews working on linear projects are always moving. So for linear projects, it is necessary to show at every point in time, which portions of the activities have being done, and which position is currently being working at. The LSM method represents this information graphically.

It is only able to represent deterministic production rates, and only efficient for constant production rates. In this project, several activities had probabilistic production rates. But the LSM method can only deal with deterministic values, so the

means of the distribution were used in the models. This resulted in over-optimistic estimates of the project duration, due to the bullwhip effect. Though it might be possible to develop a stochastic LSM method, it is doubtful whether such a method would lose the biggest advantage of the LSM method — the ability to facilitate visualization. The diagram might contain too much information and become too complicated for the user to make sense of it.

Even when there are no uncertainties in the production rates, the LSM method could become difficult to use. When all the activities in the project have straight progress line — all production rates are constant from the beginning to the end, scheduling with the LSM method is fast and easy. The *controlling point* (the point where one continuous activity touches the buffer of its predecessor activity) is either the start or the end of the progress line, which means that one can quickly finish the schedule by connecting the lines head-to-head or end-to-end.

However, if the activities are segmented and have different production rates in different sections (e.g., the right-of-way and the trenching activity in the example), or the activity's progress line has a learning curve, it could be difficult to determine the controlling point. Often, knowledge in algebraic geometry is needed to calculate the accurate positions of the controlling points.

It can easily represent the slow-downs and breaks of the continuous activity. As shown in Figure 3-11 and Figure 3-12 , the LSM method shows clearly when the activity will be blocked by its preceding activity, whether it will slow-down to follow the pace set by the successor, or stop the work completely for a predefined period of time.

3.2.1.2 Modeling of dependency relationships

LSM can model continuous dependency relationships — buffers. By maintaining the buffers between the activities, the LSM method can ensure the constraints between the activities at any point.

LSM can model both time-based and distance-based dependency relationships. In the LSM method, the buffers could be measured both horizontally and vertically, which means that the imposed constraints could be either time-based or distance-based.

3.2.1.3 Modeling of resources

LSM shows the work-path of the resources, but does not model resources directly. On the LSM diagram, each line represents the work path of a resource (or a crew). It could be easily read from the diagram how many crews are working on each activity, when and where each of them starts their work, where they are and how much work they have completed at a certain point of time, which direction they have been progressing in, etc.

However, the LSM method does not model the resources directly: the quantity limit of the resource, the priorities of the activities in requesting the shared resource, alternative resource assignment plans cannot be represented. From the LSM diagram alone, it is difficult to realize that Stringing and Lowering-in both used Crew B, and there was only one such crew available for the project. This might result in the mistake of scheduling Lowering-in to start before Stringing was completely finished.

3.2.1.4 Modeling of other factors.

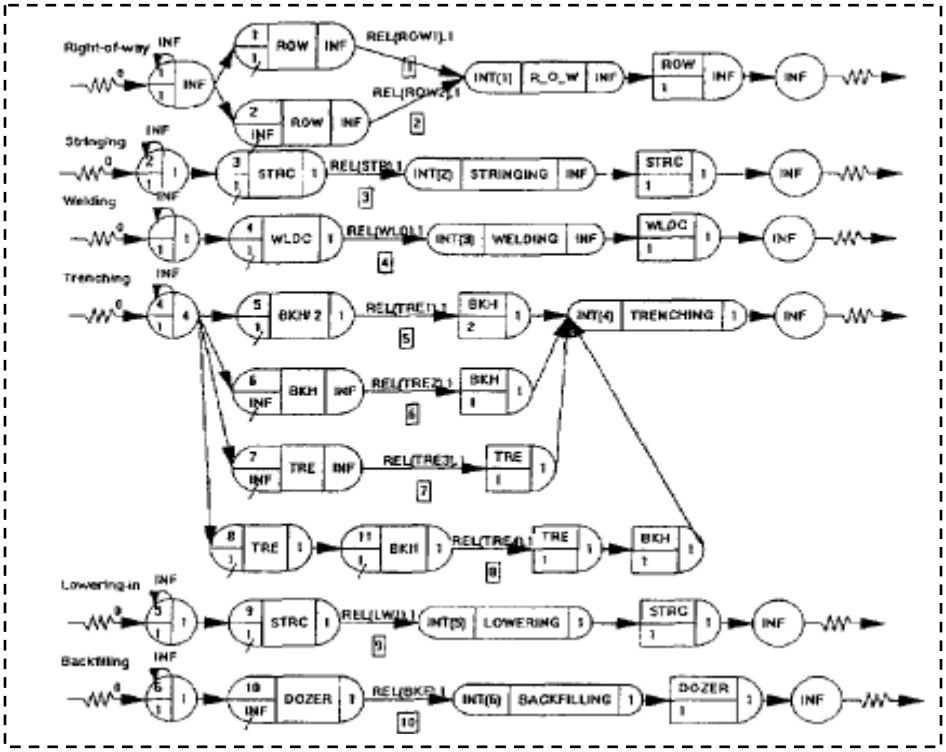
LSM is not able to represent dynamic factors and their impacts. Temperature, precipitation, wind and other dynamically changing factors cannot be represented with the LSM method. Consequently, their impacts on the activities cannot be modeled.

3.2.2 The Continuous Simulation Model

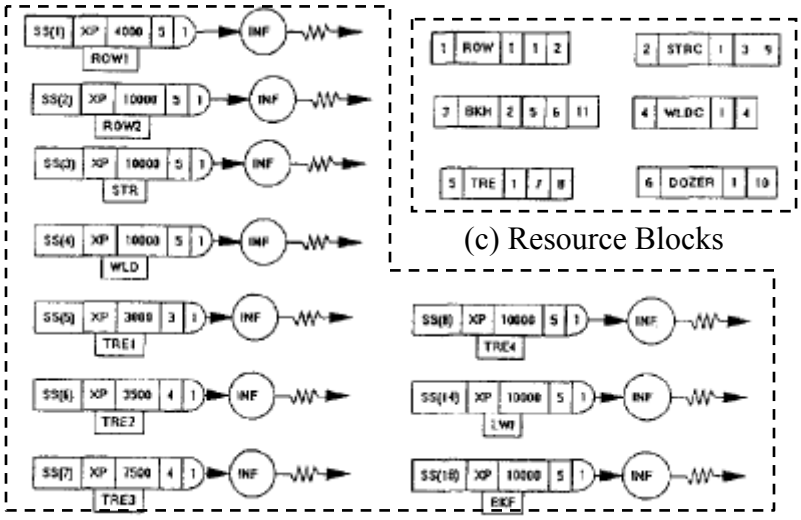
Figure 3-13 shows the continuous simulation model for this project developed by Shi and Abourizk (1998) with the general purpose simulation tool Slam II. The model includes six sub-networks (Figure 3-13 (a)), ten DETECT nodes (Figure 3-13 (b)), and six RESOURCE blocks (Figure 3-13 (c)) and a user-written sub-routine.

Each sub-network represents one continuous activity. In the first sub-network, the one represents the Right-of-way, two entities are generated at the beginning of the network by the CREATE node. The two entities are then routed to the two following AWAIT nodes , both waiting for the resource labeled as "ROW". ROW is defined in the resource block. It has a quantity limitation of 1, and gives Activity #1 a higher priority over Activity #2. Therefore, the entity in the upper Await node occupies the resource ROW and starts Activity #1, i.e., the first 6000m of the right-of-way. The duration of Activity #1 is defined as REL(ROW1), which means that this activity is going to be treated as a continuous process, and it will be associated with a DETECT node named "ROW1".

The detect node ROW1 monitors the value of the state variable SS (1). Once the value approaches 6000m, Activity #1 will be ended, a COLLECT node will collect the statistics, and a free node will release the occupied resource ROW back to the resource pool. Then Activity #2 can start. The sub-networks of the other activities are constructed with a similar structure.



(a) Sub-networks



(c) Resource Blocks

(b) Detect Nodes

Figure 3-13. SLAM II model for Case Two. [Adapted from Shi, J., and Abourizk, S. (1998). "Continuous and combined event-process models for simulation pipeline construction." *Construction Management and Economics*, 16, 489-498. (Page 492, Figure 2)]

In the user-written SUBROUTINE, 18 variables are defined to represent the status of the activities. For example, *SS (3)* represents the progress of Activity #3 Stringing. The value of *SS (3)* is updated at fixed time intervals according to:

SS(3) = SSL(3) + RATE(3) DTNOW*

where *SSL(3)* is the value of S*S(3)* at current time *TNOW*, *DTNOW* is the length of the fixed time-step, and *RATE (3)* is the production rate of Activity #3 for the next time interval, which is determined by:

RATE (3) = 400 m (the actual production rate) when

SSL (3) – SSL (1) ≥ 100m and SSL (3) – SSL (2) ≥ 100m;

RATE (3) = 0 otherwise.

where SSL (1) and SSL (2) represent the current progress of the right-of-way Section 1 and Section 2.

The advantages and limitations of the continuous simulation method can be discussed in parallel to those of the LSM method as presented in 3.2.1.

3.2.2.1 Modeling of activities

SLAM II is able to represent probabilistic and stochastic production rates. With Slam II, the production rate of the activity could be defined as probabilistic distributions or expressions that contain any accessible variables, which would lead to a much more realistic estimation of the total project duration compared to the deterministic LSM method.

SLAM II is able to show the continuous progress of the activities at any point in time. At each fixed time step, the productivity of the activity is re-sampled (if probabilistic) and re-calculated, and the progress of the activity is updated. By setting the

size of the steps small enough, the resulted progress curve could show the progress of the activity at any point with adequate accuracy.

SLAM II is able to represent slow-downs when the continuous activity is blocked, but have difficulties in representing breaks. As demonstrated in Figure 3-11 and 3-12, whether the activity is stopped or slowed-down when it is blocked, has a big influence on the total project duration. The response as programmed in the SLAM II model actually imitates the slow-down option: the activity would pause for a few short intervals and proceed for a few short intervals; the resulted progress line appears like a shadow of its predecessor activity.

It is very difficult to simulate the other option. To suspend the activity for a *specified* period of time would require writing codes in the subroutine to set the RATE to zero before the simulation time approaches (TNOW + specified interruption length). A more difficult task is to release the occupied resources from the interrupted activity so that they can be used elsewhere. With SLAM II, the resource is grabbed by the entity before it enters the activity and is not released until the entity leaves the activity. During the interruption, though the RATE is zero and the activity makes no progress, the entity stays in the activity and the resource cannot be released.

3.2.2.2 Modeling of dependency relationships

SLAM II is able to represent the continuous dependency relationships - buffers. The buffer constraints are checked at every fixed time step. So when the size of the steps is small enough, it could be ensured with adequate confidence that the constraints are maintained throughout the whole process.

Theoretically, SLAM II is able to represent buffers based on any measures. The constraints on the production rate of the activity could be a compound expression

containing any accessible variables; therefore, it is able to represent buffers measured on distance, percentage completion or any other measures.

3.2.2.3 Modeling of resources.

Resources can be realistically represented. With SLAM II, as with most of other simulation tools, resources can be represented realistically with their type, quantity and other attributes. The shared resources can select the next activity to execute according to the predefined priorities.

3.2.2.4 Modeling of other factors.

Dynamic factors and their impacts can be represented. With SLAM II, the changes of the dynamic factors can be simulated as continuous processes in separate sub-networks. The values of the dynamic factors can be accessed and used in the representation of the production rates, thereby impact the progress of the activities.

The continuous simulation model is even more difficult to develop and understand than the discrete simulation models. Continuous simulation is much less popular than discrete simulation in the area of management science. Only a few simulation packages provide continuous simulation tools as minor features, and usually those tools are not convenient to use. The user needs to build complicated networks and write codes to model even one simple continuous activity.

3.2.3 More on Linear Projects

Cheng(2005) presented an interesting scheduling problem: in a site development project, there are often multiple utility lines (such as water, gas, sewage, storm water, electrical,) that need to be constructed. These utility lines usually have different layouts. The construction of each line needs its own work space; when conflicts occur, the one with the lower priority has to pause to give way to the one with the higher priority. The

contractor would like to predict the total project duration given this requirement, so that they would be able to compare different schedules and minimize the impacts of interruptions. Figure 3-14 shows part of the site plan Cheng presented in his study.

If one tries to apply the LSM method or the continuous simulation method to this project, as in the previous example, one would quickly fail — the problem poses some new challenges. If all activities in the project are performed along one single path (even if it is not a straight path), it is adequate to represent their positions with a one-dimensional co-ordinate, i.e., the total distance that has to be traveled from the starting point to their current position, which is often referred to as the *chainage*. The distance between the activities therefore is the difference of their chainages.

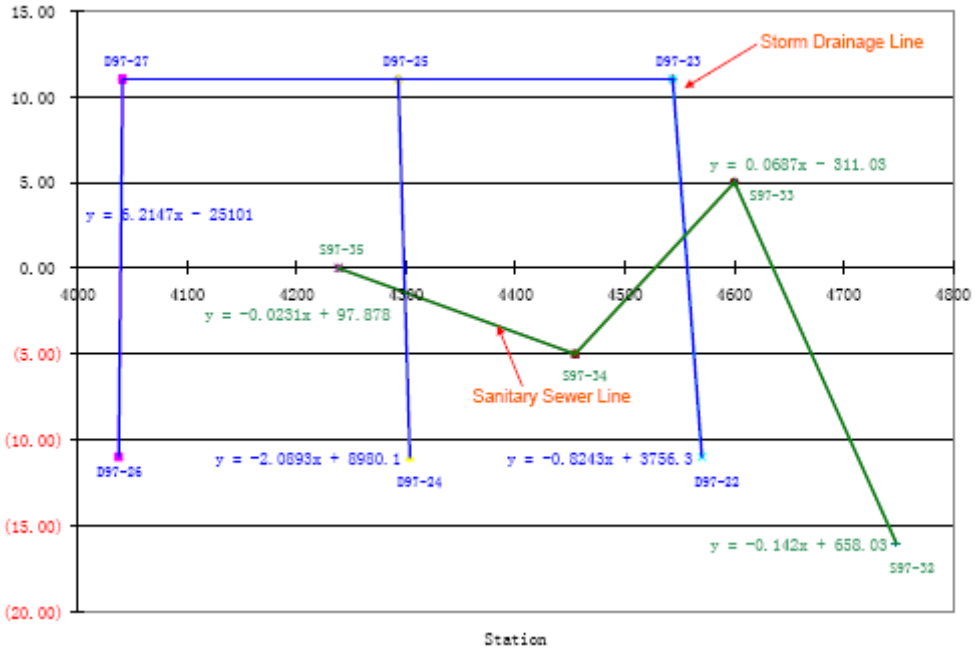


Figure 3-14. Site layout of two intersecting utility lines. [Adapted from Cheng, B. (2005). "Limitations of Existing Scheduling Tools in Planning Utility Line Construction Projects." University of Florida, Gainesville. (Page 58, Figure 6-5)]

Now the activities have different layouts. A two-dimensional coordinate system has to be used. The LSM method has only one axis to represent the spatial dimension, so it cannot satisfy the scheduling requirements of this type of projects. With SLAM II, it is possible to use two variables to record the position of the activities (e.g., use *XX (i)* and *YY (i)* to represent the horizontal and vertical coordinates of Activity *i*), and use the decomposed vector of the production rates (e. g., the horizontal component and vertical component of *RATE (i)*) to update the positions; but this would make the code-writing work in the subroutines of SLAM II even more demanding.

The second problem is related with multiple levels of scheduling. The construction of one utility line includes multiple activities: just as in the previous example, the construction of the gas line includes 6 activities. The constraint "when the construction of Line 1 runs too close to the construction of Line 2, Line 1 must yield" actually means that "when any activity on Line 1 runs into the buffer of any activity on Line 2, that activity on Line 1 must yield". Suppose that both lines consist of 6 activities, to represent this constraint, 6*6/2 = 18 dependency relationships have to be defined. None of the existing scheduling methods provides an efficient way to represent these types of constraints.

3.2.4 Summary of Modeling Linear Projects

The requirements for modeling typical linear construction projects have been summarized in Table 3-4.

Table 3-4. Modeling requirements for linear projects

		LSM	SLAM II
Activities	Productivity	Deterministic	Probabilistic and stochastic
	Slow-down	Yes	Yes
	Break	Yes	No
	Positions	Yes	Difficult. Could be represented with extensive coding.
	Layout	One dimensional	Difficult. Could be multi-dimensional with extensive coding.
Dependency relationships	Buffers	Yes	Yes
	Buffer types	Distance-based and time-based	All types, theoretically
Resources	Direct representation	Not directly represented, but the work paths of the resources can be visualized.	Yes. Resources can be defined with type, quantity, and other attributes.
Environmental factors		No	Yes, but have to be represented with separate networks.
General		Easy to learn, to use and understand, and provide great visualization when the project is not complex.	Difficult to learn, to use and understand.

3.3 Case Three: A Typical Repetitive Project

A repetitive project consists of many similar discrete units or continuous segments. The construction of each discrete unit can be seen as a regular project. The construction of each continuous segment can be seen as a linear project. The major problems in the modeling of repetitive projects are how to represent the similarities and variances of the units/segments and how to connect the units/segments.

In this case study, we are going to examine the construction of the upper-structure of the 14-level condominium, including two garage levels (Level 1 and Level 2) and twelve condo floors (Level 3 to Level 13).

After the foundational part of the building (as presented in Case One) was done, the work developed into a rhythm repeated on each level. Each floor in this condominium was 13,000 SF and was built in two separate pours: a smaller pour of 5,500 SF, and a larger pour 7,500 SF. The larger pour carried over the center line of the building to include the area around the elevator shaft. The roof was also divided into two pours, 1,400 SF each. So in total there were 24 concrete pours.

Each pour consists of five repeating activities: “Form”, “RMEP”, “Pour”, “Cure & Stress”, and “Vertical”. The first activity, “Form”, was to run aluminum beams spanning the vertical columns off the previous floor, and place slab formworks. This activity experienced an apparent learning curve: the productivity increased gradually from 1584 SF/Day, to 1703 SF/Day, to 1875 SF/day and finally stabilized at 2187 SF/day. The second activity, “RMEP” included the reinforcement of the slab, and the mechanical, electrical & plumbing preparation before the concrete placement. “RMEP” also had a learning curve: at the beginning, it took 3 days to finish a pour; afterwards the time was reduced to 2 days. “RMEP” had to keep at least 1500 SF behind “Form” to give it enough workspace. “RMEP” was immediately followed by “Pour”, which was finished within 1 day. After the poured concrete deck hardened to 66% of its designed strength, the post-tensioning cables were stressed. “Cure and Stress” took 2 days.

For this case study, we are going to examine the CPM model, the LSM model and the STROBOSCOPE model, and focus on their weaknesses in the modeling.

3.3.1 The CPM Model

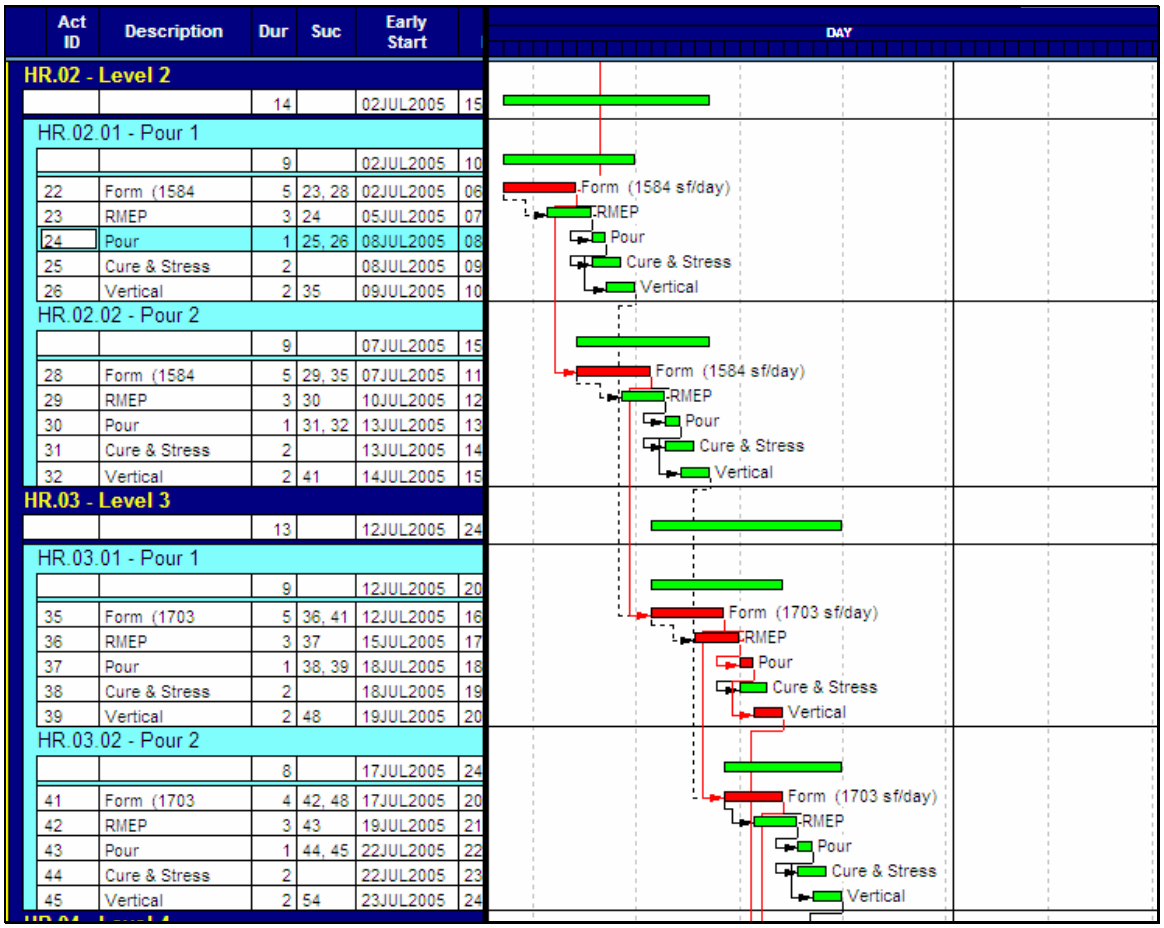


Figure 3-15. Part of the CPM model for Case Three

Figure 3-15 shows part of the CPM schedule developed for this case study. The following problems have been identified in this schedule:

3.3.1.1 Efficiency

Development of the CPM schedule for the repetitive project is time-consuming and error-prone. First of all, it is very time-consuming to develop this schedule. There were 28 pours in the upper part of the structure, each built with the same construction method, though the duration of some activities varied at a few locations. The CPM schedule does not utilize this repetitive characteristic. The sub-network of each pour has to be inputted separately. One might save some time and effort by copying the sub

network of one pour and then paste it 27 times; however, the dependency relationships between these sub-networks still need to be entered one by one.

The process is also error-prone. When repeatedly inputting a group of unit-networks that have 90% similarity, one would tend to ignore the 10% difference. This happened with the 3rd floor, where the reinforcement plan was different from the other floors. The scheduler forgot to change the length of duration for the activity Reinforcement at this floor when copying and pasting the sub-networks. This mistake caused a hectic rush when the work progressed to this level.

3.3.1.2 Resource-imposed dependency relationships

CPM is inflexible in representing resource-imposed dependency relationships. There are two types of dependency relationships in construction projects. The first type is imposed by technical requirements and normally cannot be altered. Most dependency relationships in regular construction projects belong to this type. The dependency relationships between different activities within one unit/segment in repetitive projects also belong to this type. For example, within each pour, "Form" must begin before "RMEP", and "RMEP" must be finished before "Pour" can start.

The second type of dependency relationships are imposed by resources. In regular projects, if a resource is shared among more than one activity, the activities need to be sequenced linearly. In repetitive projects, crews working on one activity need to go through the units/segments one by one, imposing predecessor/success relationships among these units/segments. Most connections between different units/segments in the schedules of repetitive projects belong to this type. The resource-imposed dependency relationships are much more flexible, and may be changed as a result of different managerial preference of crew utilization strategies.

For example, according to the schedule shown in Figure 3-15, the Form crew was going to do “Pour 1” first and “Pour 2” second, but it was totally feasible for the crew to do it the other way, i.e., “Pour 2” first and “Pour 1” second.

The number of crews assigned to one activity may also change the dependency relationships among the units/segments completely. Now there was only one crew working on the activity “Form”, so that “Form” in all units/segments had to be sequenced into one line with the Finish-to-Start relationships. If the project manager decided to reduce the total duration of the project and add a second crew to work on “Form”, the dependency relationships would need to be changed completely. There are many different ways that the two crews could be assigned: the project manager could assign a smaller crew to work on the smaller pours, and a big crew to work on the larger pours; or use the same type of crews and have each of them work on the earliest available units. Each strategy would result in a different way of linking the units/segments.

Since the CPM method cannot represent resources, it cannot represent resource-imposed dependency relationships directly. Every time the managerial preferences, the number of crews, or the crew utilization strategies change, their impacts on the dependency relationships need to be figured out manually, and a lot of adjustments need to be made to the schedule.

3.3.2 The LSM Model

Figure 3-16 shows the LSM schedule developed for Case Three. As can be seen in this example:

- **LSM is difficult to represent parallel activities on the diagram.**

In the example project, several activities were taking place the same time in the same unit. “Pour” partly overlapped with “Cure & Stress”, which also partly overlapped

with “Vertical”. Many color, shading and denotation combinations had to be tried before a proper scheme could be designed to differentiate the parallel activities.

- **Scheduling with the LSM method for repetitive projects is not efficient when certain assumptions cannot be fulfilled.**

When used for repetitive projects, the LSM method could be very effective if two assumptions are true. First, all units/segments are made up of the same network —the activities contained in the units/segments are identical; the durations/productivities of the activities are identical; and the dependency relationships within the unit/segment are identical. Second, work continuity has to be maintained for all activities.

Under these conditions, each repetitive continuous activity could be represented as a straight continuous line, and each repetitive discrete activity could be represented as a straight continuous double line whose “thickness” is the total time needed to finish one unit. The schedule can be quickly developed in a similar way as presented in Section 3.2.1. However, if there are variances among the units/segments, the LSM method would be just as inefficient as the CPM method or even more cumbersome.

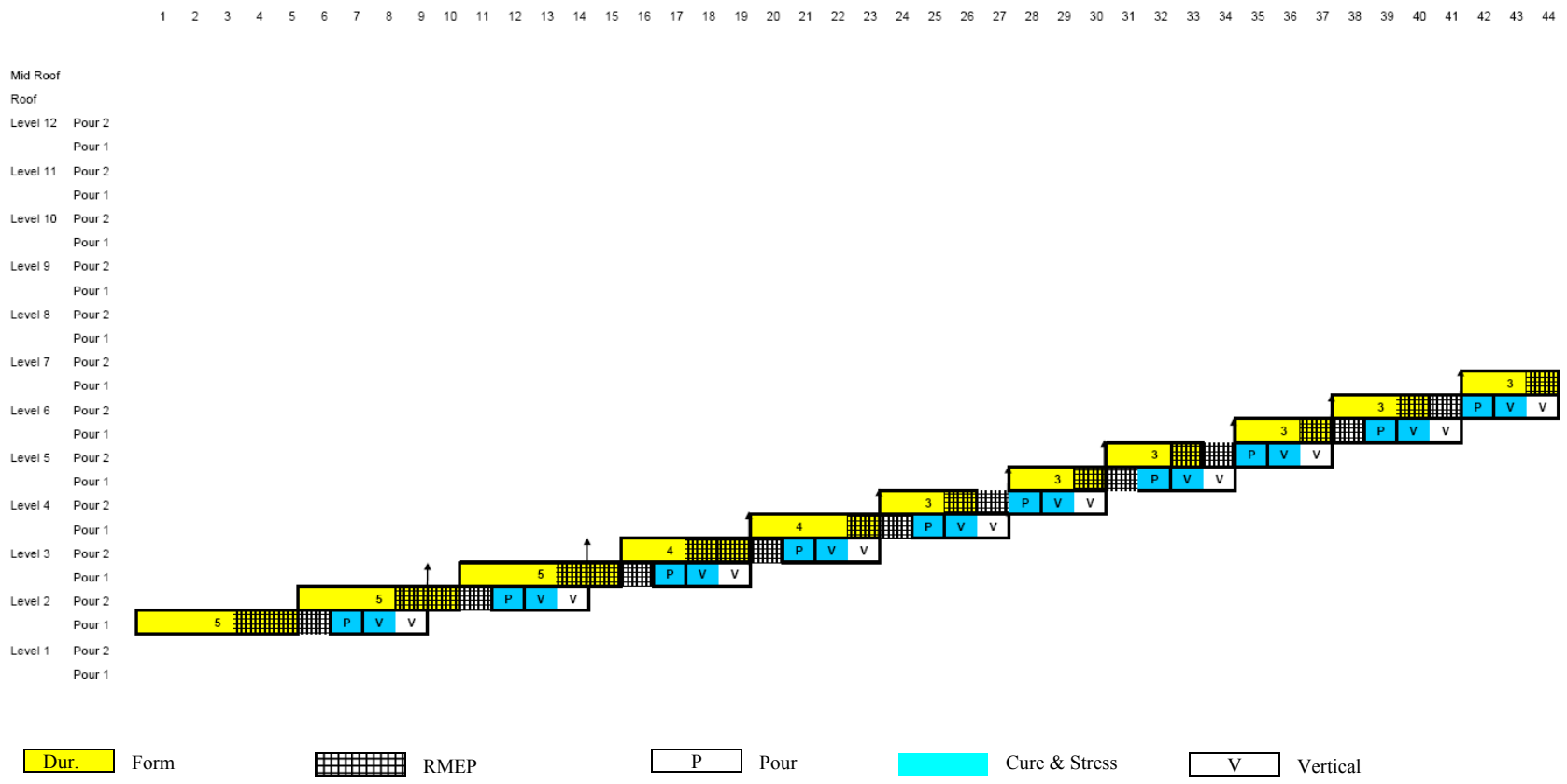


Figure 3-16. The LSM model for Case Three

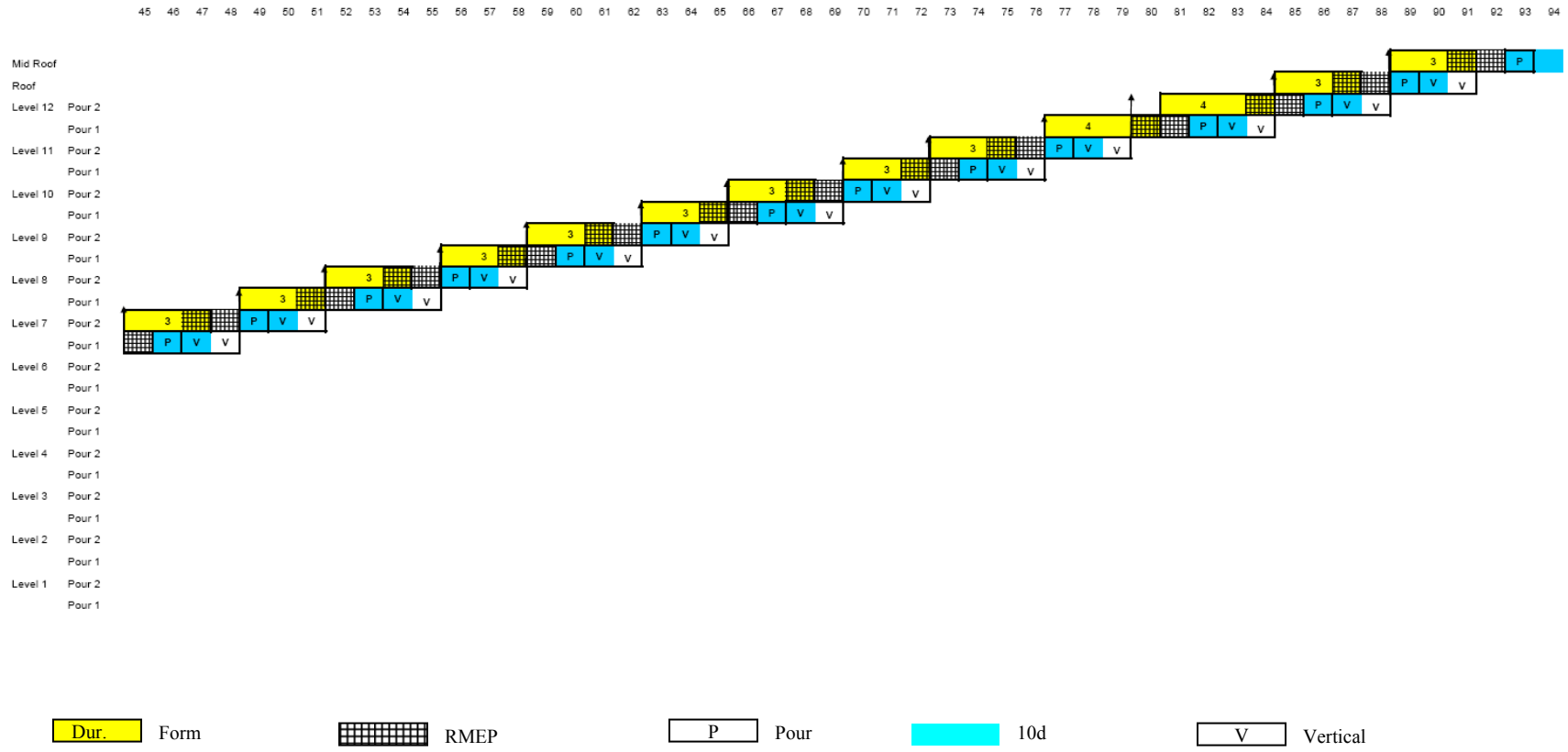


Figure 3-16. Continued

In this project, the time needed to finish the formwork in one pour varied from 5 days to 3 days. Therefore the repetitive activity "Form" had to be represented unit-by-unit as a series of separated blocks (as shown in Figure 3-16). There is no easy way to determine the controlling point on this repetitive activity, and it is very difficult to position the succeeding activities accordingly. Repetitive projects consisting of continuous segments have similar problems if the productivity rates of the activities vary frequently from segment to segment.

- **Scheduling with the LSM method for repetitive projects is very difficult when there are hetero-relationships.**

The term *hetero-relationship* is borrowed from the research on the LSCHEDULER algorithm (El-Rayes and Moselhi 2001). Here it is used to refer to the dependency relationships between two different units/segments of two separate repetitive activities.

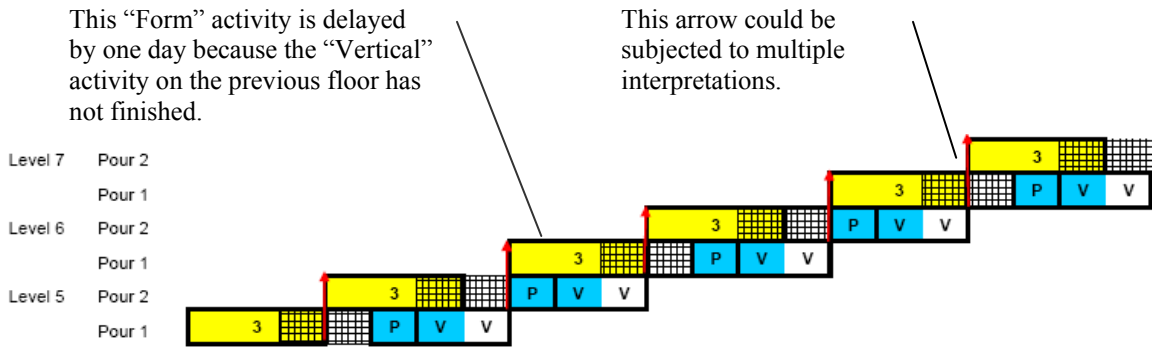


Figure 3-17. Representation of the hetero-relationship in the LSM diagram

In this project, beams and deck formworks were placed on the top of the vertical columns off the lower floor, indicating that the activity "Form" on Floor *n (n = 2 to 14) Pour i (i= 1 or 2)* could not start until the activity "Vertical" on Floor *n-1 (n = 2 to 14) Pour i (i= 1 or 2)* was finished. To model this constraint, we need to add a vertical arrow at the end of each "Vertical" activity (as illustrated in Figure 3-17) in the LSM diagram,

and examine the related “Form” activity against it, If a “Form” activity starts before the arrow, it has to be pushed back.

Because of the hetero-relationships, it is very unrealistic to maintain the work continuity in this project. RMEP, Pouring, Cure & Stress and Vertical have to start as early as possible in every unit so as to avoid that Form on the next level is delayed too long. As shown in Figure 3-17, in each unit, the activities are clustered tightly together, whereas the connections among the blocks of the same repetitive activity are frequently broken, indicating that there are a lot of interruptions along each repetitive activity.

It is difficult to read from the LSM diagram what these arrows actually represent and which activities they are restraining. The arrow as noted in Figure 3-17 could be interpreted as a constraint on the start of “Form” at Floor 7 Pour 2, as it was intended to be, or be misinterpreted as a constraint on the start of “Pour” at Floor 7 Pour 1 or “Cure & Stress” at Floor 7 Pour 1.

3.3.3 The Simulation Model

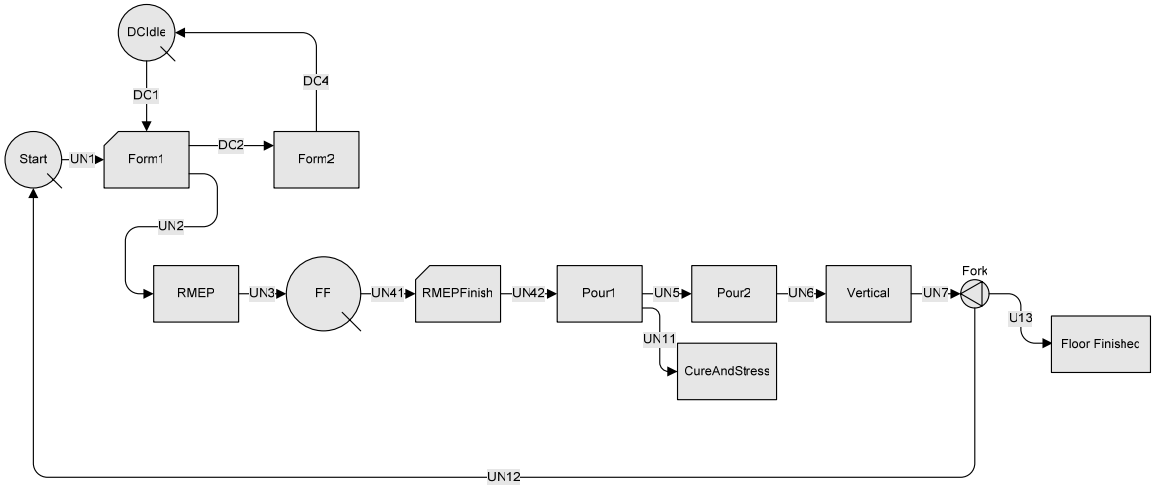


Figure 3-18. The STROBOSCOPE model for Case Three

Figure 3-18 shows the STROBOSCOPE model developed for the Case Three. The first node “Start” queue holds 24 entities of type “UN” (Unit) at initiation, each entity

representing one concrete pour. Since the deck forming crew is the most critical resource for floor construction and it sets the pace of the work, it is also included in the model. One “DCrew” entity is initiated in the “DCIdle” queue, and it is required to start the “Form1” activity.

3.3.3.1 Efficiency

STROBOSCOPE is efficient at modeling repetitive activities. The construction of the 14 floors (26 concrete pours) is represented by just one cycle in the STROBOSCOPE diagram. Compared to the CPM model in Figure 3-15 and the LSM model in Figure 3-16, it is much more concise. The variances among the units are modeled by utilizing the characterized resources. For example, the durations of activity Pour, Cure & Stress, and Vertical are identical in all units, so their values can be directly defined as an attribute of the activity. The durations of the activity Form and RMEP vary across the units, so their values are defined as attribute *FORMDur* and *RMEPDur* of the characterized resource “Unit”. As a “Unit” enters a Form or a RMEP activity, the value of the correspondent attribute is assigned to the duration of the activity. Thereby, STROBOSCOPE achieves efficiency in representing the repetitiveness and at the same time it also provides some flexibility in representing the variances among the units.

3.3.3.2 Resource-imposed dependency relationships

STROBOSCOPE is able to model resource-imposed dependency relationships. In front of the Combi activity “Form1”, there are two queues: the “Start” queue that holds twenty-six virtual tokens — “Units”, and the “DCIdle” queue that holds one formwork crew — “DCrew”. The activity needs to obtain one “Unit” and one “DCrew” to start. To simulate multiple crews on this activity, the project manager can simply initiate more

than one DCrew in the queue. By changing activity priorities and queue disciplines, many different crew utilization strategies can be represented.

3.3.3.3 Hetero-relationships

STROBOSCOPE is difficult to use in representing hetero-relationships.

Representing hetero-relationships with the STROBPSCOPE method usually takes many trials and errors. Figure 3-19 shows one solution.

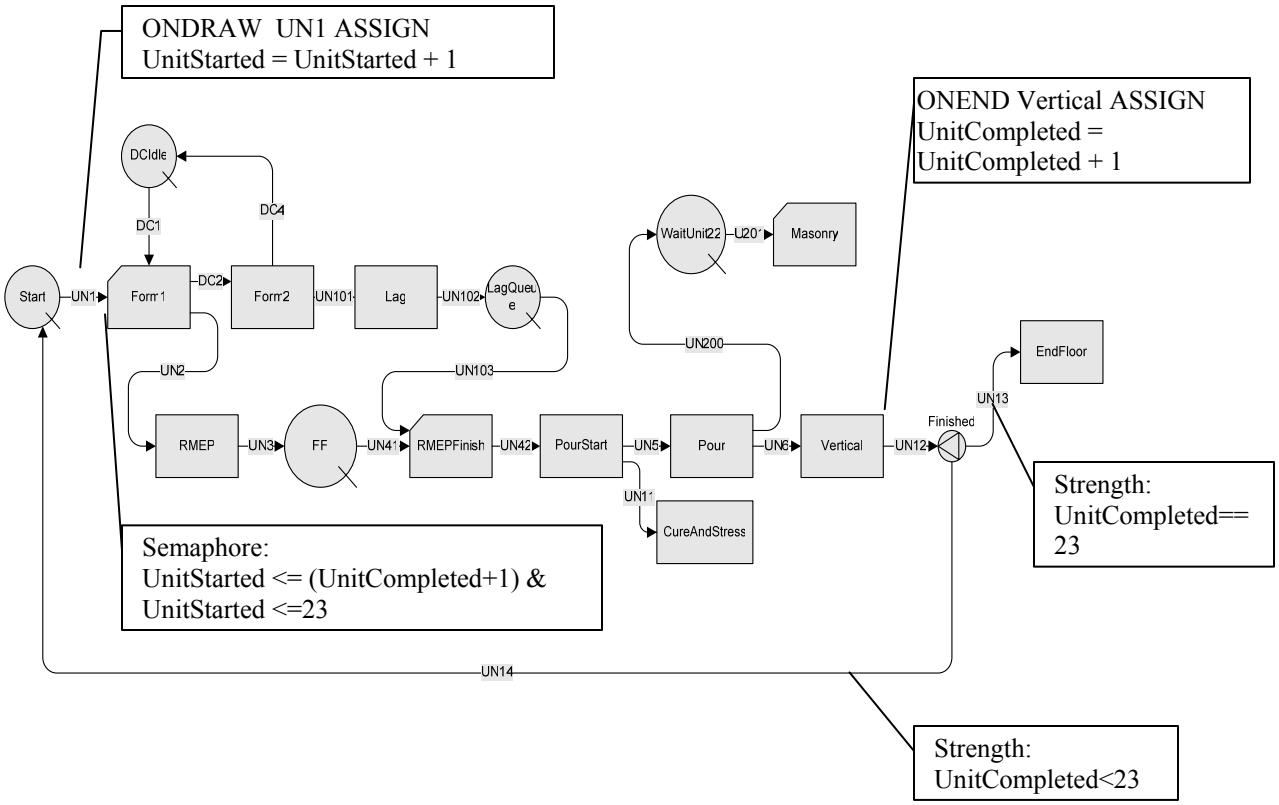


Figure 3-19. Representation of the hetero – relationships in STROBOSCOPE

Two variables are defined in the model: *UnitStarted* which counts the total number of units that have entered the activity "Form", and *UnitCompleted* which counts the number of units that have finished the activity "Vertical". A new "Form" activity cannot begin unless the value of the Semaphore *(UnitStarted<=UnitCompleted+1)* is true. Also,

the units need to be routed back to the "Start" queue in order to trigger the initialization of the next "Form" activity once they exit from the "Vertical" activity. But from the 23rd units, this back flow has to be stopped, otherwise the model will begin to create the "25th" unit. A fork named "Finished" therefore has to be inserted to control when to stop this process.

3.3.4 Summary of Modeling Repetitive Projects

The requirements for modeling typical repetitive construction projects are summarized in Table 3-5.

Table 3-5. Modeling requirements for repetitive projects

	CPM	LSM	STROBOSCOPE
Efficiency	Not efficient	Efficient only when all assumptions are satisfied. Difficult to show parallel activities.	Efficient
Resource-imposed dependency relationships	Not directly modeled. Not flexible.	Show resource flows, but does not model the resources directly.	Yes
Hetero-relationships	Have to be represented one-by-one.	Very difficult	Very difficult

3.4 Case Four: A Project of Mixed Features

In the last case study, the whole construction process of the 14-level condominium's structure system will be examined. This project has fixed features because: (1) it contains both non-repetitive activities (as discussed in Case One) and repetitive activities (as discussed in Case Three); and (2) it contains both discrete activities and continuous activities. A CPM model, a LSM model and a STROBOSCOPE model have been developed and are presented in Figure 3-20, 3-21 and 3-22 respectively.

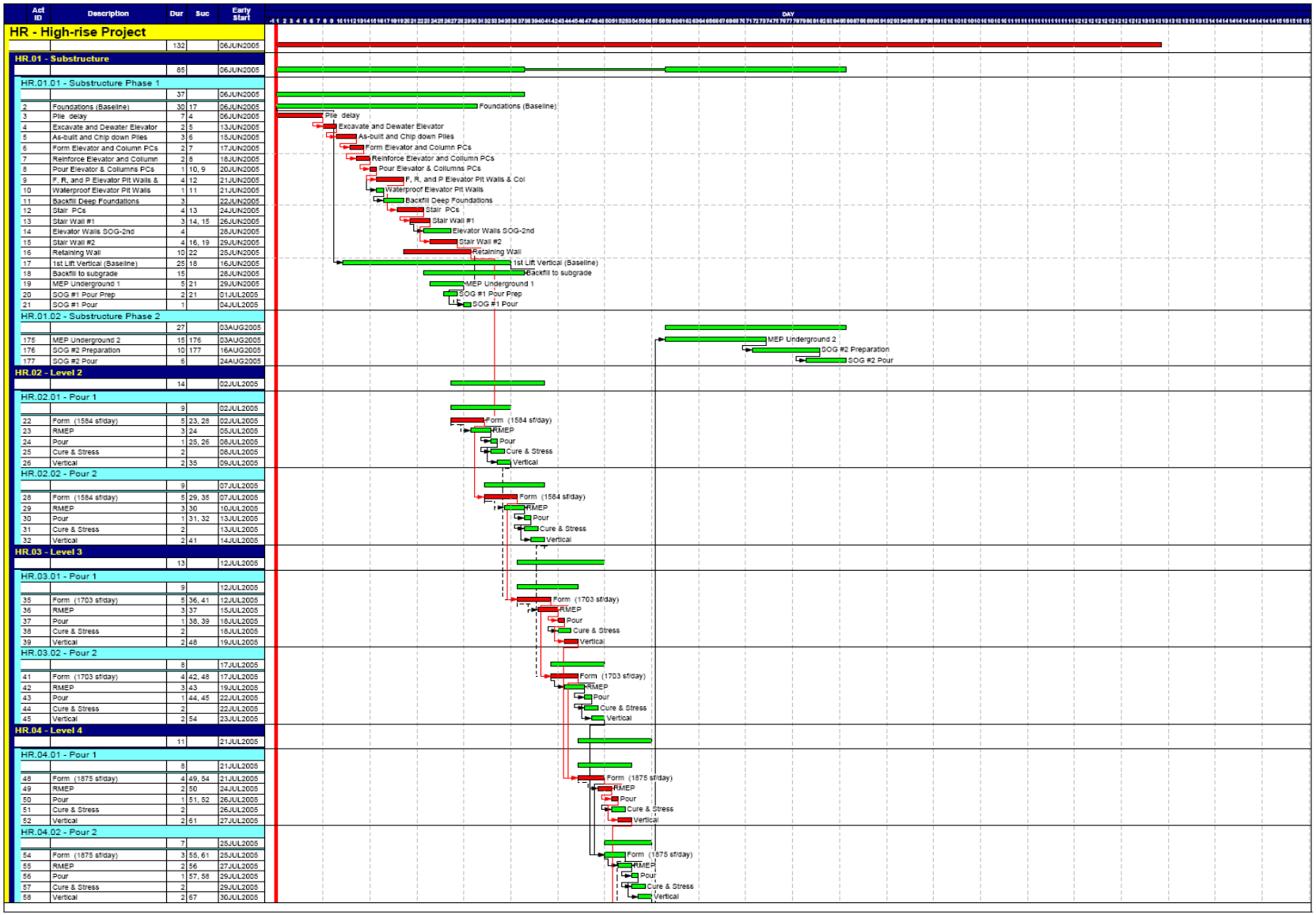


Figure 3-20. The CPM model for Case Four

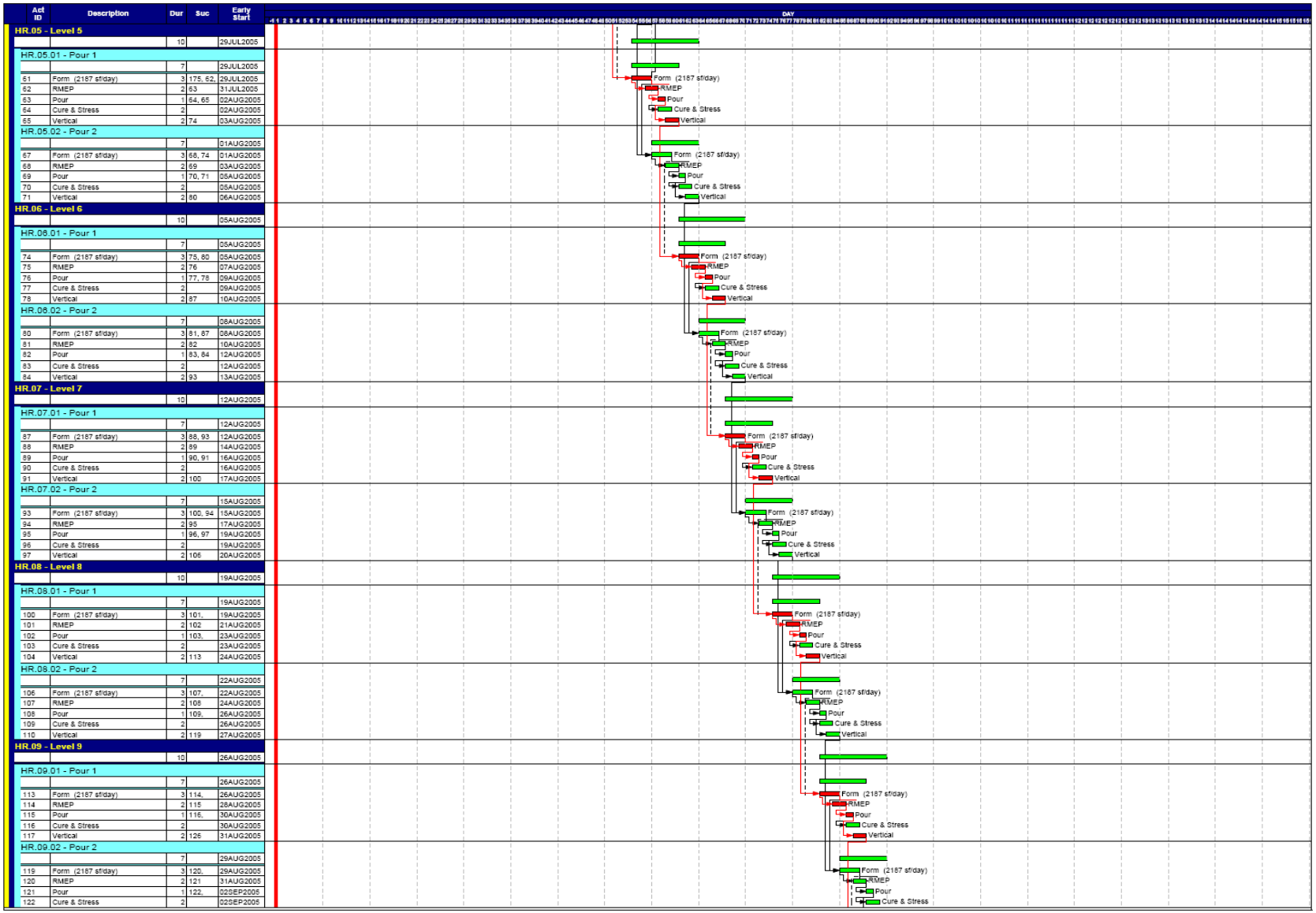


Figure 3-20. Continued

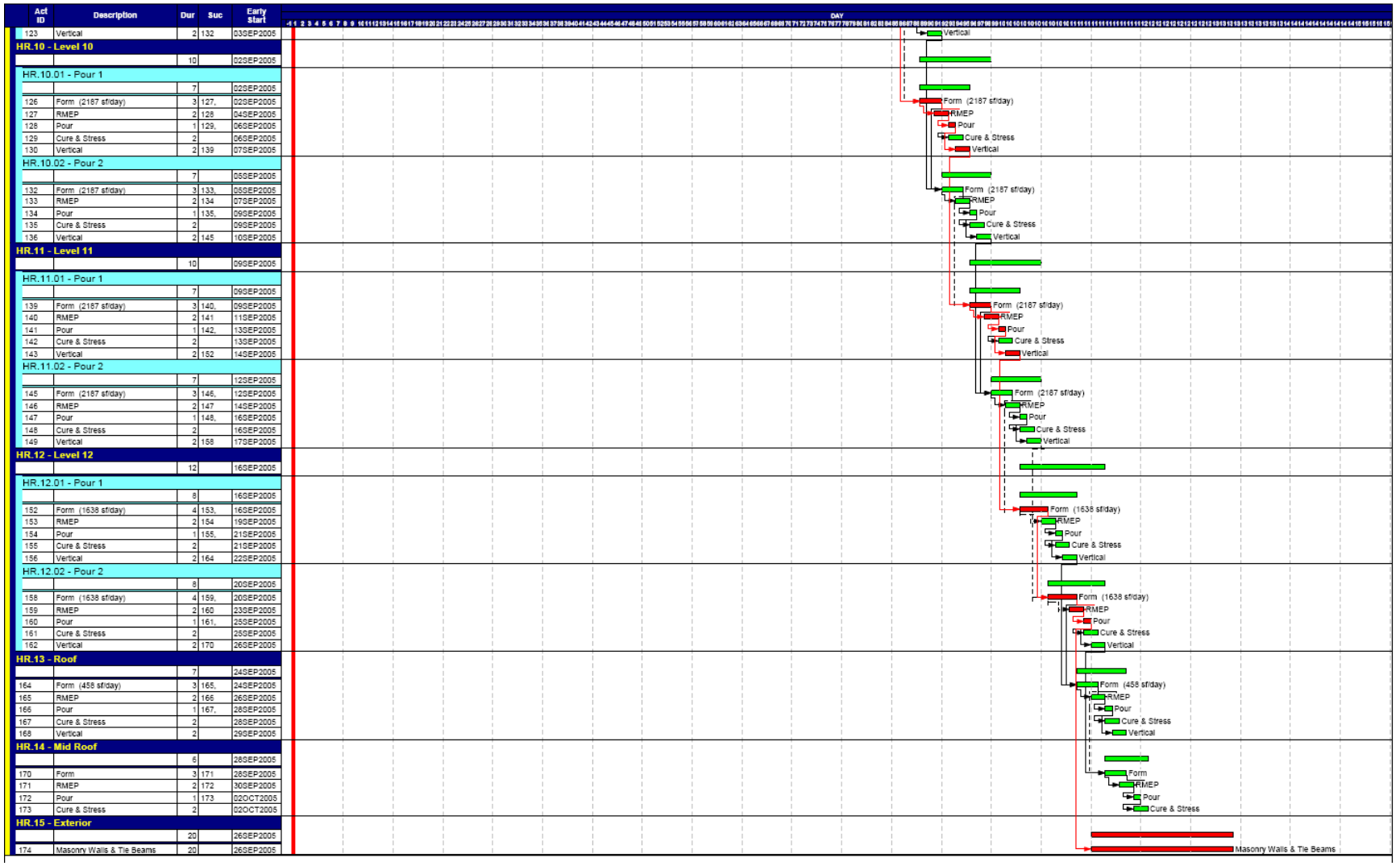


Figure 3-20. Continued

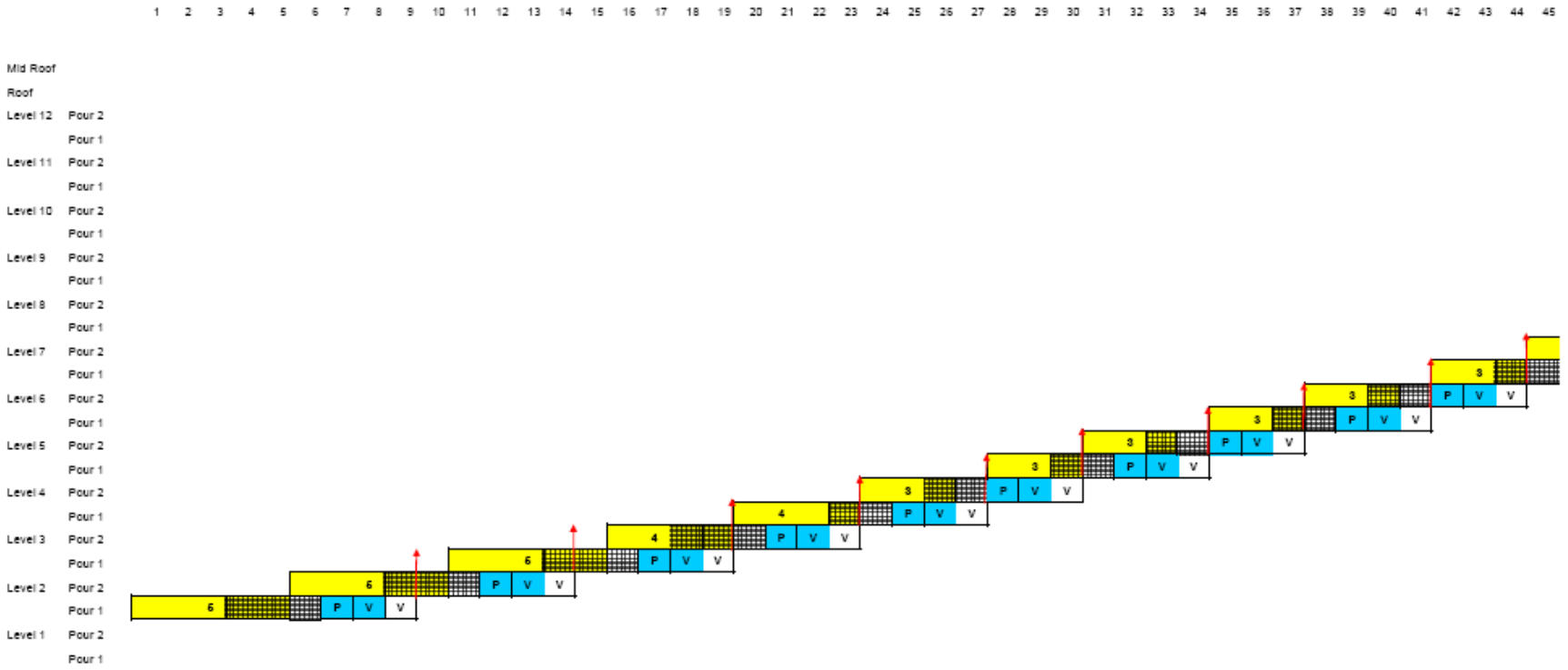


Figure 3-21. The LSM model for Case Four

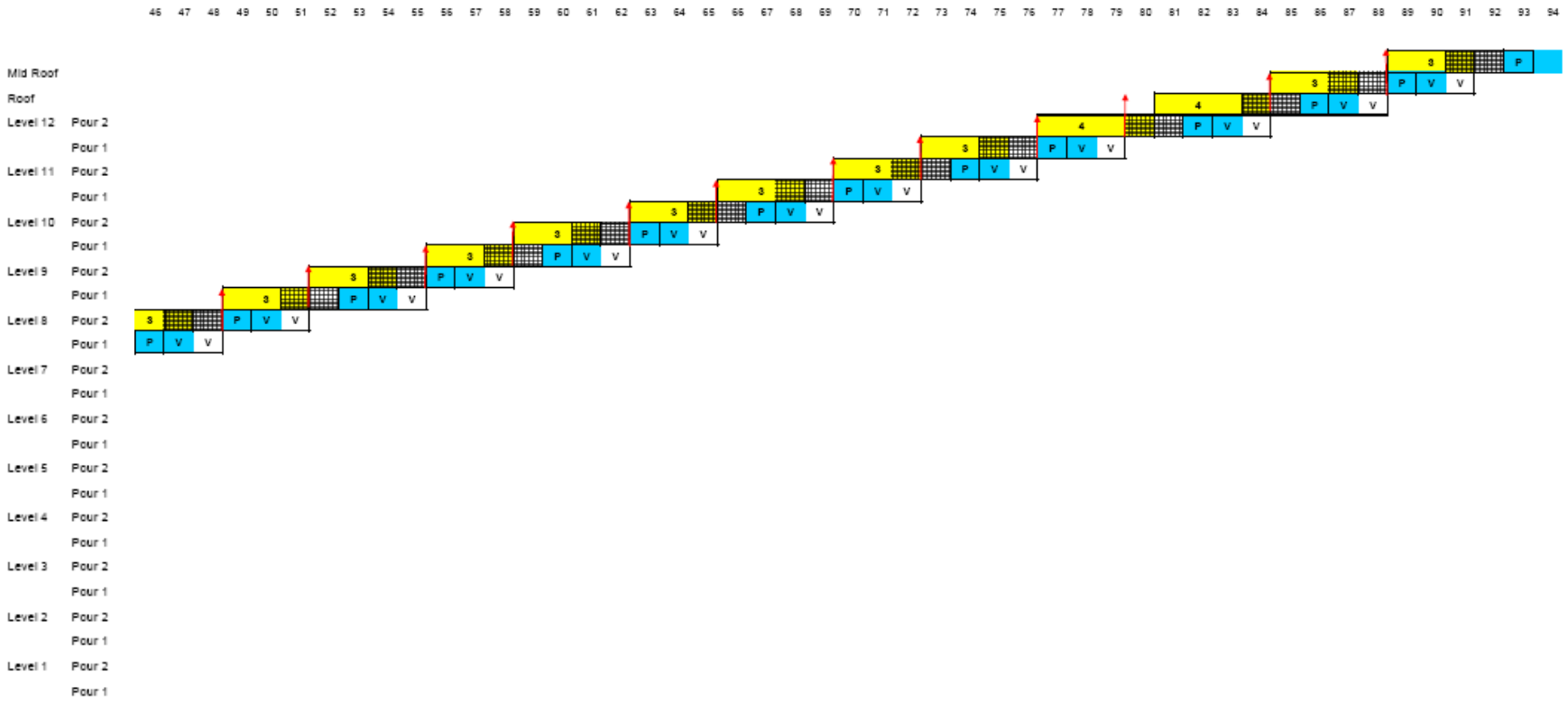
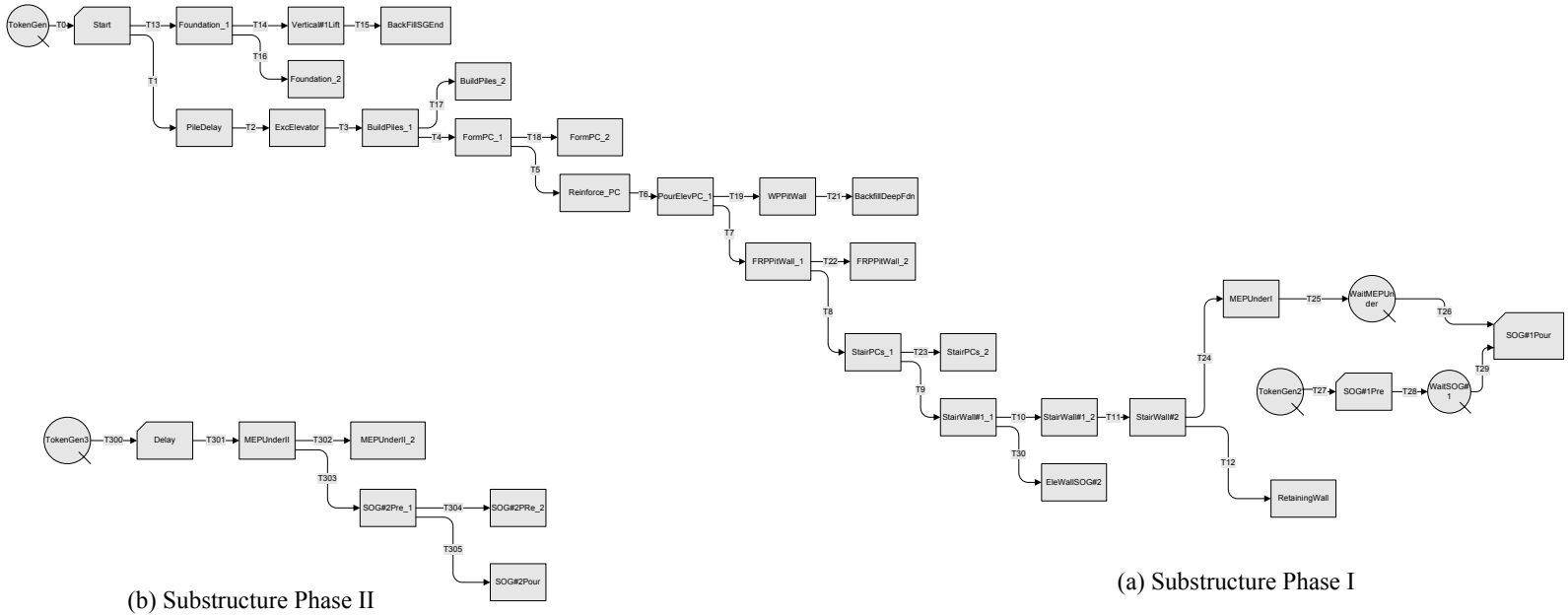
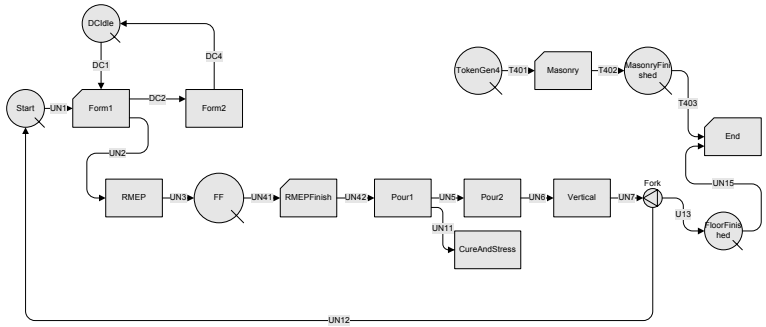


Figure 3-21. Continued



(b) Substructure Phase II

(a) Substructure Phase I



(c) Superstructure

Figure 3-22. The STROBOSCOPE model for Case Four

This case study focuses on integration of different types of activities, including integration of discrete and continuous activities, and repetitive and non-repetitive activities.

3.4.1 Integration of discrete and continuous activities

Among the existing planning and scheduling tools, the LSM method is the only one that allows easy integration of discrete and continuous activities. In Figure 3-23, “Form” and “RMEP” are represented as lines, showing how they progress gradually over time and the continuous buffer is maintained along the whole process. “Pour”, “Cure & Stress” and “Vertical” are represented as blocks, showing only their start and finish times at each pour.

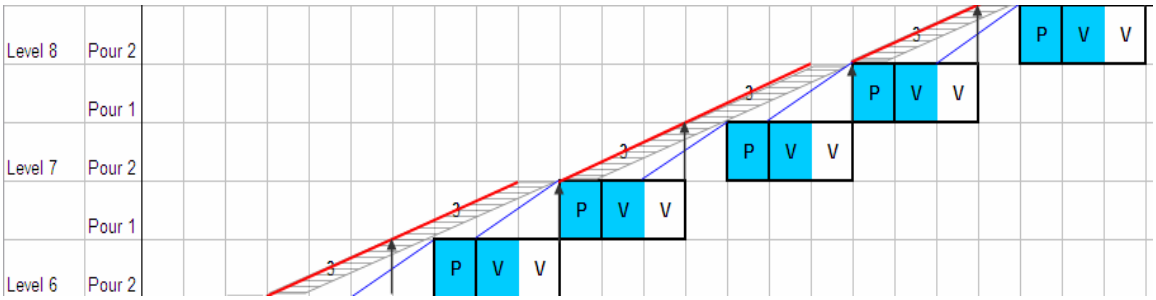


Figure 3-23. Integration of continuous and discrete activities in LSM

The CPM method is not able to represent continuous activities and continuous dependency relationships. All continuous activities have to be divided into discrete segments, and all continuous buffers have to be transformed into an SS dependency and an FF dependency, which may result in mistakes as discussed in Case Two.

Similarly, STROBOSCOPE can only control the startup of the activity — the value of the Semaphore and the availability of the required resources are checked before the activity starts. No constraints can be imposed in the middle of the activity. Figure 3-24 shows part of the STROBOSCOPE model that is used to represent the buffer between the activity “Form” and the activity “RMEP”. This representation is more complex compared to the CPM representation of buffers, while it will result in the same mistake as the CPM method does.

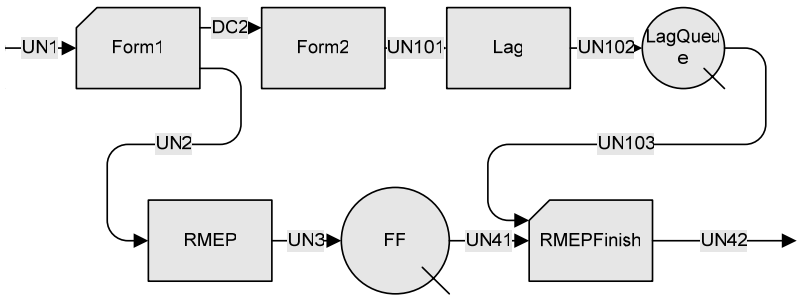


Figure 3-24. Representation of a continuous buffer in STROBOSCOPE

3.4.2 Integration of repetitive and non-repetitive activities.

The LSM method is not able to represent non-repetitive activities. So the schedule only contains the upper structural part.

The CPM method is able to represent both non-repetitive activities and repetitive activities (but only discrete repetitive activities). However,since the CPM method represent each unit/segment of the repetitive activities separately, there is a large amount of repeated and redundant data in the schedule, and the interfaces between the repetitive part and the non-repetitive part are often shrouded and unclear. This can be demonstrated with the CPM schedule as shown in Figure 3-20. The foundation part of this project was constructed in two separate phases. The second phase started from the middle of the construction of the upper structure — after the formwork of the first pour at the 5^{th} level was done. While the link representing this dependency relationship is very important in project management and control, it is buried deeply among the entangled relationship lines among the repetitive activities.

In the STROBOSCOPE model, the repetitive activities usually are linked together by the flow of a physical unit — a pour in this case, while the non-repetitive activities are linked by a virtual token. Sub-networks consisting of different types of activities therefore cannot be directly connected, they need to be “synchronized” with the use of some controlling variables. For example, the second phase of substructure construction, noted as Figure 3-22 (b), cannot start

until 2 days after the formwork on the 1st pour of the 5th floor has been completed. To model this relationship, a variable *PhaseIIStart* is defined and initiated with value 0. A semaphore "PhaseIIStart==1" is added to the first activity in Figure 3-22 (b). When Unit7 flows through "Form1" in Figure 3-22 (b), it changes the value of *PhaseIIStart* from 0 to 1, thus satisfies the semaphore and allows Figure 3-22 (b) to start. The actually implementation is even more complicated than this, as there are time delays involved.

Another example is "Masonry cannot start until the 2nd concrete pour on the 12th floor has ended". As shown in Figure 3-25, this constraint is implemented by duplicating Unit22 when it passes through activity "Pour" and then sending the duplicate to trigger the activity node "Masonry". In addition, code must be written in the Link "UN6" to restrict the number of units it can draw at one time. Otherwise both the original and the duplicate will go to the activity "Vertical".

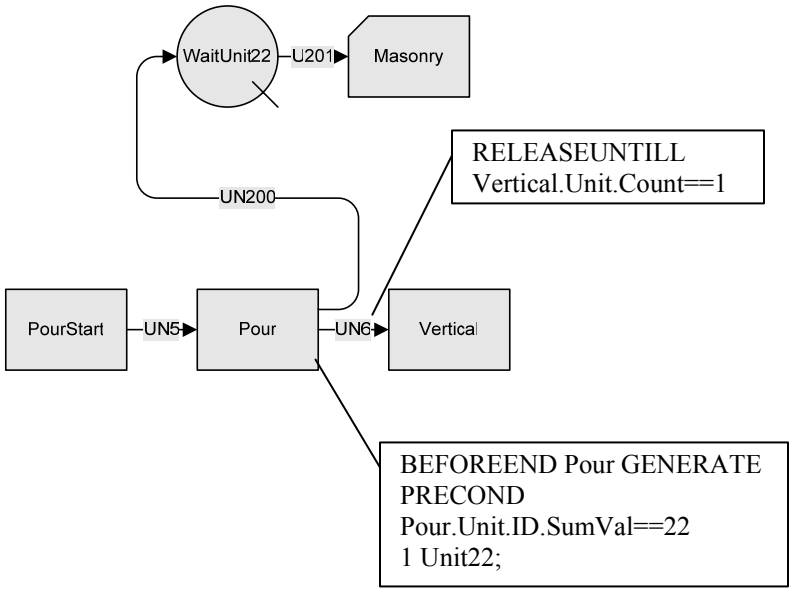


Figure 3-25. Integration of repetitive and non-repetitive activities in STROBOSCOPE

3.4.3 Summary of Modeling Repetitive Projects

The requirements for modeling construction projects with mixed features are summarized in Table 3-6.

Table 3-6. Modeling requirements for projects with mixed features

	CPM	LSM	STROBOSCOPE
Integration of discrete and continuous activities	No	Yes	No
Integration of repetitive and non-repetitive activities	Yes, but not efficient.	No	Yes, but difficult.

CHAPTER 4
REQUIREMENT ANALYSIS

In Chapter 3, we presented four typical types of construction projects and examined the models developed for them with various existing planning and scheduling tools. Each technique shows its strengths in the modeling of certain aspects and certain parts of the projects, yet each also has fundamental limitations making it inadequate for addressing a lot of requirements that arisen from real-world construction projects.

So is it possible to develop a new modeling technique that (1) can be universally applied to any types of construction projects, and (2) incorporates and enhances the best features of the existing techniques? If it is possible, what essential characteristics should such a tool exhibit? This chapter tries to portray an "ideal" modeling technique for construction planning and scheduling, while the question of whether such a technique can be realized and how to implement it will be left for the following chapters.

The desired characteristics will be analyzed from four perspectives: breadth of application, form of representation, accuracy of modeling and levels of modeling.

4.1 Scope of Application

Scope of application refers to the domain that the planning and scheduling method is designed for. Each of the existing tools has a limited scope of application which fits its basic assumptions: the CPM method is designed for regular projects, the LSM method for linear projects, and the CYCLONE-based simulation method methods for cyclic operations. Although these methods can be applied to certain areas outside of their domains, such as the CPM method for repetitive projects, or the CYCLONE-based simulation tools for regular projects, they would become very cumbersome in use. The case studies in Chapter 3 have clearly demonstrated this point.

The desired planning and scheduling method should be able to represent all types of construction projects, namely, regular projects, repetitive projects, linear projects and projects that have mixed features, with sufficient efficiency. An even more ambitious idea is for this method to incorporate the modeling of lower-level operations as well.

From another perspective, the method needs to be able to integrate the discrete and continuous, repetitive and non-repetitive elements together. As shown in Figure 4-1, Regular Projects mainly consist of non-repetitive discrete activities, while Linear Projects consist of non-repetitive continuous activities. Discrete activities, when repeated many times, form the Repetitive Discrete Project; continuous activities, when repeated, form the Repetitive Linear Project — these two types of projects usually are not differentiated, both being referred to as Repetitive Projects. A project of mixed features may contain both the continuous and discrete, repetitive and non-repetitive elements. A method that is able to represent both the discrete and continuous, repetitive and non-repetitive elements and integrate them together would be able to model all types of construction projects.

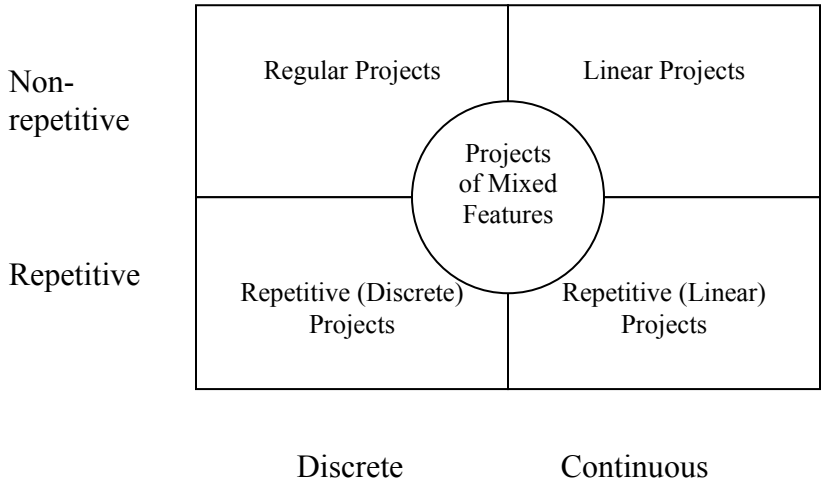


Figure 4-1. Scope of application

4.2 Accuracy of Modeling

The concept of *accuracy* refers to how faithfully a model is able to represent its real-world counterpart. The more properties and behaviors of the real-world system the model is able to represent, the more accurate it is.

In another sense, accuracy of modeling also is a measure of the scope of application. But rather than determining which type of projects can be modeled, it determines how many types of realistic situations it is able to handle. Table 4-1 summarizes the results shown in Table 3-1, 3-4 and 3-5, and provides an overview of the basic requirements for a universally applicable planning and scheduling tool to fairly accurately model all types of construction projects.

However, there are always trade-offs among accuracy, complexity and computing efficiency. As the accuracy of the model increases, the complexity of the model increases too, demanding more computing resources.

A very important point is that for a given objective, a highly accurate model may be just as valid as a less accurate one. For example, when the intermediate progress of an activity is not of concern, representing it as a continuous activity is as valid as representing it as a discrete one, but the continuous representation would require much more resources. An ideal planning and scheduling tool should allow flexibility with regard to the level of accuracy, so that the complexity of the schedule could be appropriate for the objective of the specific planning and scheduling study.

Moreover, no matter how capable the modeling tool is, it is not possible to include all properties and behaviors of its source system — the properties and behaviors of a real-world system are infinite. Therefore, the desired modeling tool should include the most intrinsic properties and behaviors of construction projects, but it must be extensible to incorporate additional properties and behaviors when necessary.

Table 4-1. Summary of the key requirements for modeling construction projects

			CPM	LSM	STROBOSCOPE	SLAM II
Discrete	Activities	Duration	Only real numbers and probabilistic distributions.		Both deterministic and probabilistic. The value can be dynamically determined. Could be resource driven.	
		Progress curve	No		No	
		Interruption	No		No	
		Adjustment	No		No	
	Dependency Relationships	FS,SS,FF, SF relationships	Yes		Only FS relationships can be directly represented.	
		Non-time-based dependencies	No		No	
		Compound constraints	Only "AND" logic.		Yes, but need extra nodes or coding.	
		Branching	No		Support both probabilistic and decision-based branching.	
Continuous	Activities	Productivity		Deterministic		Probabilistic and probabilistic
		Slow-down		Yes		Yes
		Break		Yes		No
		Position		Yes		Could be represented with extensive coding.
		Layout		One dimensional		Could be multi-dimensional with extensive coding.
	Dependency Relationships	Buffers		Yes		Yes
		Buffer types		Distance-based and time-based		All types, theoretically.

Table 4-1. Continued

		CPM	LSM	STROBOSCOPE	SLAM II
Resources	Representation	No	Not directly represented, but the work paths of the resources can be visualized.	Include generic, characterized and compound resources.	Yes. Resources can be defined with type, quantity, and other attributes.
	Activity selecting resources	No	No	Yes, but activities can only select resources in the same queue.	Yes
	Resource selecting activities	No	No	Yes, but an activity can only be assigned one priority value for all the resources it shared with other activities.	Yes
Environmental Factors		No	No	Yes	Yes, but these factors need to be modeled with separate sub-networks.
Repetitive	Representation	Not efficient	Difficult to show parallel activities. Not efficient when certain assumptions cannot be fulfilled.	Efficient	Not efficient, each segment has to be represented as a separate sub-network
	Resource-imposed dependency relationships	Not directly modeled. Not flexible.	Show resource flows, but does not model the resources directly.	Yes	Yes
	Hetero-relationships	Have to be represented one-by-one.	Difficult	Difficult	Difficult
Integration	Discrete and continuous	No	Yes	No	Yes
	Non-repetitive and repetitive	Yes	No	Yes, but difficult	Yes, but difficult

4.3 Form of Representation

Form of representation refers to how the user is going to describe a project. The CPM method uses the AOA or AON network, the LSM method uses the time-location diagram, and most of the existing simulation method uses the ACD diagrams. Besides graphical representations, the representation can also take the form of dialogue windows in computer programs, simulation languages, general purpose languages, or mathematical formulations.

The following characteristics are considered essential for good representations:

- **It should be *easy* to learn.**

The popularity of the CPM method in a great degree is owing to its advantage in this perspective. The simulation method, on the other hand, has not been well-accepted by the construction industry till now primarily because it takes a lot of time to learn the tricks about how to use modeling elements, modeling rules, and the program-specific simulation languages.

- **It should be *efficient* in use.**

In planning and scheduling, efficiency is required in two areas: the development of a new schedule and the editing of an existing schedule.

Construction projects are characterized by their strict deadlines and dynamic natures. The project manager usually can only devote very limited time to scheduling. Under a lot of occasions he or she has to quickly develop a draft schedule within hours, and editing the schedule within minutes.

As for the development of new schedules, the CPM method is very efficient in use for typical regular projects, but not for repetitive projects. When there are *n* units, every repetitive activity needs to be represented *n* times in a CPM schedule (refer to the repetitive part of the CPM model in Case Study One). The simulation model is not as efficient as the CPM method for regular projects — several nodes often have to be used in order to represent a simple logical

relationship (refer to the non-repetitive part of the simulation model in Case Study One). But the simulation model is very efficient for repetitive projects — it can use the network of one unit to represent the work of *n* units (refer to the repetitive part of the simulation model in Case Study One). The LSM method is most efficient for repetitive projects and linear projects, but only when the projects satisfy a lot of assumptions.

A good planning and scheduling tool should be efficient for all types of construction projects.

As for editing, the efficiency of a planning and scheduling tool is mostly determined by how it handles repetitive activities and resource-imposed dependency relationships. As discussed in Section 3.3.1, resource-imposed dependency relationships are very flexible and subject to change with different resource utilization strategies.

- **It should reflect the *natural* way that people look at the project.**

One of the most important functions of a schedule is to facilitate communication. There are many parties involved in the planning and scheduling process, including the project manager, the foreman, the owner, the architect, the subcontractors and the suppliers, who all need to understand each other's needs and concerns.

A modeling tool is a language that people involved in planning and scheduling use in their communication — the modeling elements are the words, and the modeling rules are the grammars. The modeling elements and rules should reflect the natural way that people talk about the project. They should not require too much interpretation; otherwise people will have to focus on the "words" and "grammars" rather than the meanings and ideas.

Examples of unnatural representations can be found in the two simulation models in Chapter 3: a virtual token has to be generated, duplicated and routed to represent the start/finish

logical dependency relationships between the activities; a sub-network, a detector node and a subroutine together represent a continuous activity; etc.

The time-location diagram the LSM method, on the contrary, is very natural and intuitive. The lines, blocks, bars and buffers resemble the layout of the activities in the real world, and thus facilitate understanding and visualization of the situation.

4.4 Levels of Modeling

There are many different levels at which a construction project could be described. For example, at the top level, the construction project has the constraint on when the whole project can start, the location of the project, the total resources assigned to the project, etc.; at the secondary level, the activities have the constraint on when the activity can start, the location of the activity, and the resources available to the activity; at the third level, the activities can be decomposed into sub-activities, each of which has their individual constraint, location, and resources.

When developing a schedule, sometimes we follow a top-down approach — we start from the top level to consider the overall situation and then focusing on the details; sometimes we follow a bottom-up approach — we start from the details and then integrate them to figure out the whole. And often we need to alternate between these two approaches to ensure that the schedule satisfies all the requirements at all levels.

With the existing planning and scheduling tools, constraints, resources and other properties cannot be added on aggregated activities, which means that scheduling can only be performed on one level (although the scheduling results can be summarized to high-level views). Consequently, it is often necessary to develop multiple schedules for one project at each planning level. This results in repeated input and possible conflicts among the many different levels of schedules.

Allowing modeling on multi-levels is also required to improve the efficiency of representation. Constraints, resources and properties that are shared among similar activities can be defined only once at a high level; properties that are unique to individual activities can be defined separately on the low level.

Actually, the way in which the SCOBOSCOPE method models repetitive activities provides an example in this aspect. Assuming that the dependency relationships among the activities (which is a type of constraints) in all units are the same, only one unit network is represented, and then the durations of the activities at each unit are separately defined through the attributes of the tokens (refer to the repetitive part of the simulation model in Case Study One). And this is why the STROBOSCOPE model is much more "compact" than the CPM model for the repetitive part. However, the STROBOSCOPE method only enables modeling on two levels (one on the general activity level, one on the unit level) in a very limited way.

An ideal planning and scheduling tool should allow modeling on multiple levels with complete flexibility.

CHAPTER 5
A NEW THEORY FOR PLANNING AND SCHEDULING CONSTRUCTION PROJECTS

Chapter 4 has identified the desired characteristic of an ideal modeling tool — universally applicable; providing adequate accuracy with enough flexibility and extensibility; simple, efficient and natural in representation; and supportive of multiple-level modeling. None of the existing construction planning and scheduling tools is able to provide these characteristics.

This chapter proposes a new planning and scheduling philosophy that is aimed to deliver the desired characteristics as summarized above. The theoretical foundation for this new philosophy — system theory and DEVS (Discrete Event System Specification) will be introduced first, and then the proposed theory will be explained in details.

5.1 Theoretical Foundations

5.1.1 System Theory

Before we can develop a method that can be universally applied to "model" different types of projects, activities, resources, etc., we need to, first of all, establish a unified concept framework that can be used to "view" various types of entities involved in different type of projects. System theory provides such a unified view.

Systems theory is a trans-disciplinary field that studies the abstract organization of systems, independent of the substance, type, or spatial or temporal scale of existence. It investigates both the principles common to all complex systems, and the models which can be used to describe them. It was founded in the 1940's by the biologist Ludwig von Bertalanffy, and furthered by Ross Ashby. It has been developed into diverse branches and has been applied in numerous areas including engineering, computing, management and ecology (Klir 1991).

A *system* is a set of elements which interact with each other within the system's boundaries (form, structure, organization) to function as a whole. The central concept of system theory is

that the properties of the whole is always different from, and more than, the sum of its unassembled collection of elements.

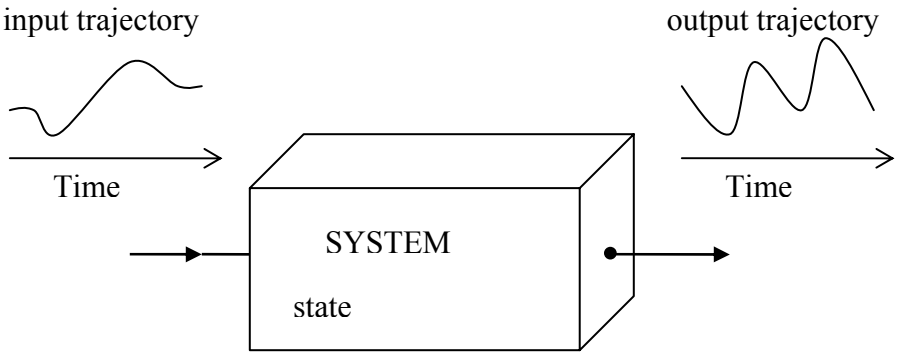


Figure 5-1. Basic system concepts. [Adapted from Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of Modeling and Simulation*, Harcourt India Private Limited, New Delhi, India. (Page 4, Figure 1)]

As shown in Figure 5-1, a system has both internal *structure* and external *behavior*. Viewed as a black box, a system's function is to transform received input time histories and generate output time histories. The relationship it imposes between the inputs and outputs is called the *external behavior*. Inside the black box, a system has its *internal structure* which include three components: (1) the *state* of the system; (2) the *transition mechanism* that dictates how inputs from the outside cause the current state of the system to transform into the next state; and (3) the *state-to-output mapping*, i.e., how the current state will lead to production of outputs.

Knowing the internal structure of a system allows one to simulate and analyze its external behaviors. From the system theory's point of view, the task of modeling and simulation is to develop an abstract model to mimic the internal structure of a system for the purpose of predicting and studying the external behaviors of the system.

As shown in Figure 5-2, a system may be broken down into *component systems* through decomposition, and component systems may be coupled together to form a *resultant system*

through composition. The ability to continue to compose larger and larger systems from previously constructed components leads to *hierarchical construction*. Furthermore, if the structure and behavior of a resultant system can be expressed in the original formalism of its component systems, this system is considered *closed-under-composition*. Such a system is considered to have well-defined structure and behavior in the system theory (Zeigler et al. 2000).

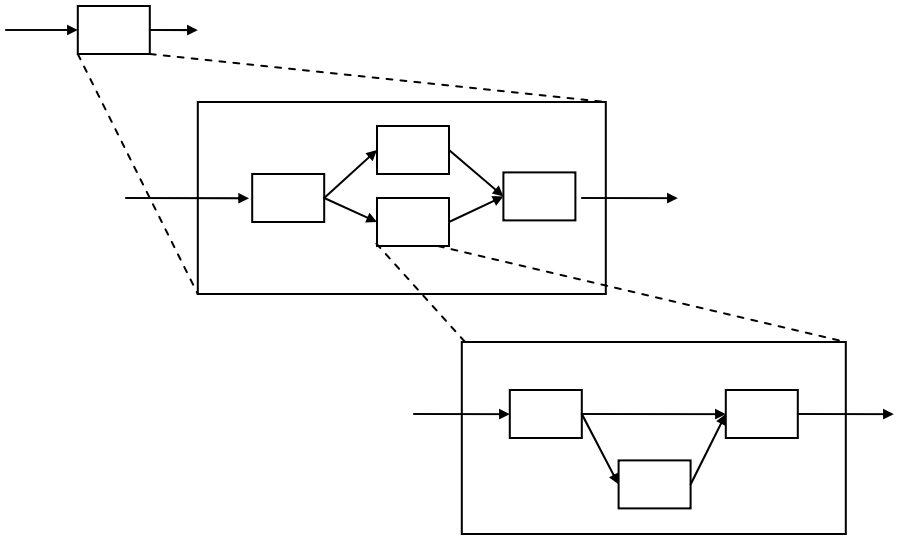


Figure 5-2. Hierarchical construction of a system. [Adapted from Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). Theory of Modeling and Simulation, Harcourt India Private Limited, New Delhi, India. (Page 5, Figure 2)]

5.1.2 The DEVS Formalism

The structure of a system may be expressed in a mathematical language called formalism. DEVS (Discrete Event System Specification) is a fundamental, rigorous formalism for representing dynamic systems. The first version of DEVS was conceived by Zeiger (1976) and referred to as TMS76. It has been improved continuously through the years and the current version is TMS99 (Zeigler et al. 2000). Now, research and development groups on DEVS are reported to be found throughout the world, including the US, Canada, Korea, Japan, UK, Portugal, France, Austria, Germany, Argentina and Mexico.

Two concepts are at the core of the DEVS paradigm:

- ***System specification formalisms*** – the modeler can use different formalisms, such as continuous or discrete, to build component models, and integrate them in one system.
- ***Levels of system specification*** – the behavior and the structure of a system can be described at various levels.

With the DEVS formalism, at the bottom level, atomic models are developed to describe: (1) how a system makes deterministic transitions between sequential states autonomously; (2) how this system reacts to external input; and (3) how this system generates output. The structure of the atomic model is defined as

$$M = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta) \qquad \text{(Equation 5-1)}$$

where

$X = \{(p,v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values;

$Y = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and values;

S is the set of state variables and parameters;

$\delta_{int}: S \rightarrow S$ is the *internal state transition function*;

$\delta_{ext}: Q \times X \rightarrow S$ is the *external state transition function*, where

$Q = \{(s, e) \mid s \in S, 0 < e < ta(s)\}$ is the *set of total states,*

e is the *time elapsed* since last transition;

$\lambda: S \rightarrow Y$ is the output function,

ta(s): $S \rightarrow R^{+}_{0,\infty}$ is the *time advance* function.

The behavior of the atomic model is illustrated in Figure 5-3. Suppose that at the current time the system is in some state S. If no external event occurs the system will stay in state S for time $ta(s)$. The value of $ta(s)$ could be a positive real number, as well as 0 or ∞. In the first case, the stay in state S is so short that no external events can intervene – therefore S is called a

transitory state. In the second case, the system will stay in *S* forever unless an external event interrupts its slumber, so *S* is called a *passive* state. When the resting time in state *S* expires, i.e., when the elapsed time $e = ta(s)$, the system outputs the value $\lambda(s)$, and changes to state $\delta_{int}(s)$. Note that outputs can only be generated just before internal transitions.

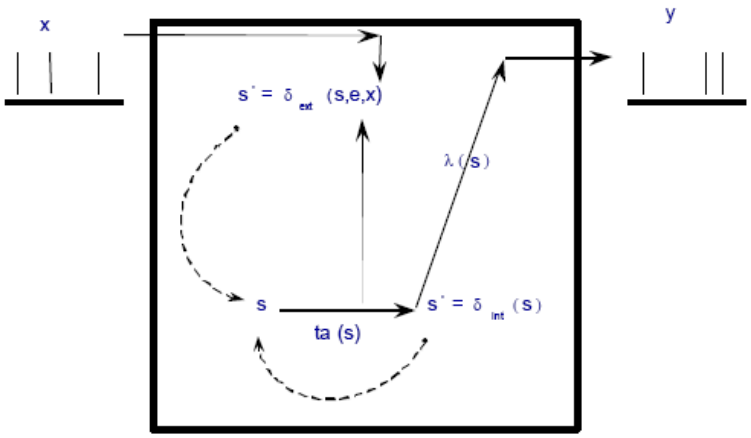


Figure 5-3. Behavior of the DEVS atomic model. [Adapted from Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). Theory of Modeling and Simulation, Harcourt India Private Limited, New Delhi, India. (Page 76, Figure 1)]

If an external event $x \in X$ occurs before this expiration time, i.e., when the system is in total state (s, e) with $e \leq ta(s)$, the system changes to state $\delta_{ext}(s)$. Thus the internal transition function dictates the system's next state when no events interrupt the system since the last transition; while the external transition function dictates the system's next state when an external event occurs – this state is determined by the input *x*, the current state *S*, and how long the system has been in this state when the external event occurred, *e*.

In both cases, the system is then in a certain new state S' with a new resting time $ta(s')$ and the same story continues.

Higher level models are constructed by coupling component models. The components can be atomic DEVS models as well as coupled DEVS in their own right. The specification of a coupled model includes the external interface (input and output ports and values), the components (which must be DEVS models), and the coupling relations:

$$N = (X, Y, D, \{Md \mid d \quad D\}, EIC, EOC, IC, Select) \qquad \text{(Equation 5-2)}$$

where

$X = \{(p,v) \mid p \in IPorts, v \in X_p\}$ is the set of input ports and values;

$Y = \{(p,v) \mid p \in OPorts, v \in Y_p\}$ is the set of output ports and values;

D is the set of the component names;

Components are DEVS models (i.e., for each $d \in D$),

$M_d = (X_d, Y_d, S, \delta_{ext}, \delta_{int}, ta)$ is a DEVS

with $X_d = \{(p, v) \mid p \in IPorts_d, v \in Xp\}$

$Y_d = \{(p, v) \mid p \in OPorts_d, v \in Yp\};$

External input couplings connect external inputs to component inputs:

$$EIC \subseteq \{((N, ip_N),(d, ip_d)) \mid ip_N \in IPorts, d \in D, ip_d \in IPorts_d\};$$

External output couplings connect component outputs to external outputs:

$$EOC \subseteq \{((d, op_d), (N, ip_N)) \mid op_N \in OPorts, d \in D, ip_d \in OPorts_d\};$$

Internal couplings connect component outputs to component inputs:

$$IC \subseteq \{((a, op_d), (b, ip_N)) \mid a,b \in D, ip_a \in OPorts_a, ip_b \in OPorts_b\};$$

Select: $2^D - \{\} \rightarrow D$, the tie-breaking function.

Zeiger (1984) have proved that the DEVS formalism is closed-under-composition: for each pair of coupled DEVS, a resultant atomic DEVS can be constructed. As such, any DEVS model, atomic or coupled, can always be replaced by an equivalent atomic DEVS. As a coupled DEVS

may have coupled DEVS as its components, hierarchical modeling is supported. The constitution of the resultant atomic DEVS also makes it possible to implement a simulator capable of simulating any DEVS models.

5.2 The Proposed Theory

In the proposed method, each activity, each type of resources and each group of environmental factors is regarded as an independent system, represented in the DEVS formalism. The state of these systems will change according to their internal transition functions if no external event intervenes, and outputs can be generated just before the occurrence of each internal transition. The internal transitions will be interrupted, however, when the system receives an input, in which case the state of the system will change as dictated by the external transition function triggered by the received input.

There are three types of atomic activity models in the proposed system: the non-resource-driven discrete activity model $Activity_d$, the resource-driven discrete activity model $Activity_{dr}$, and the continuous activity model $Activity_c$. Multiple atomic activity models can be assembled into one compound activity model, which could be a non-resource-driven discrete model $Compound_d$, or a resource-driven discrete model $Compound_{dr}$, or a continuous model $Composite_c$. Each compound activity model has its own input and output ports, and can be used just as an atomic activity model in the construction of higher-level compound activities. The repetitive activity model *Repetitive* is a special type of compound activity model.

5.2.1 Atomic Activity Models

5.2.1.1 Non-resource-driven discrete models

With the non-resource-driven discrete activity model $Activity_d$, the duration of the activity is independent of the resources, and the activity will progress according to the predefined progress curve (if no external event intervenes). This model allows the duration of the activity to

be adjusted and the progress to be interrupted in the middle by external events, provided that such events are rare. If the activity is influenced by any environmental factor that is constantly changing, the activity is better represented with a continuous activity model *Activity*$_c$.

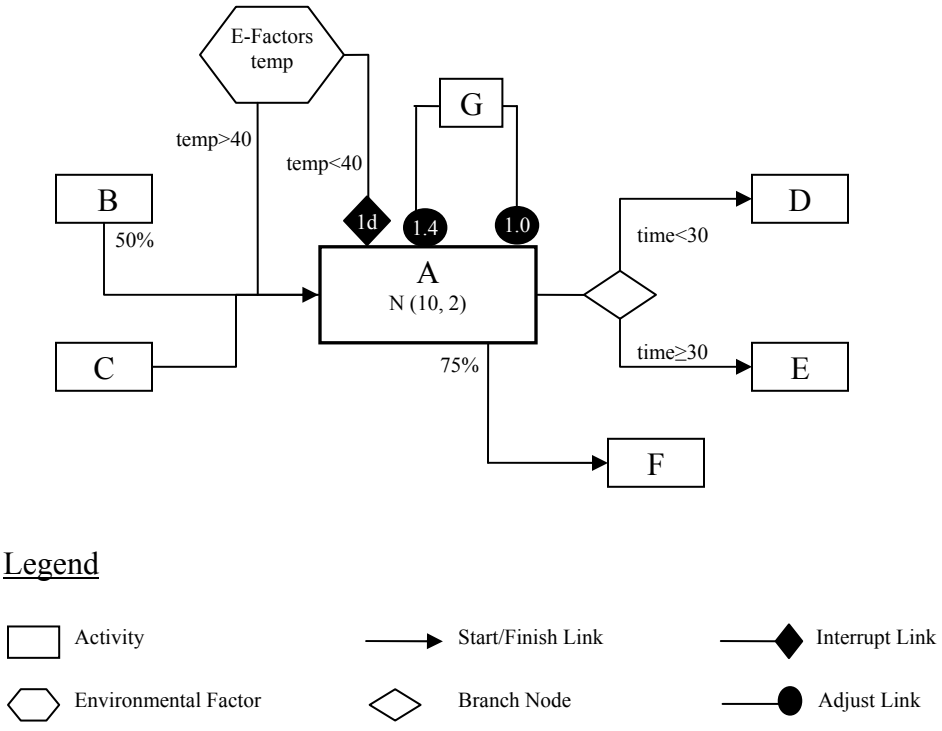


Figure 5-4. Non-resource-driven discrete Activity A with many realistic factors

Figure 5-4 shows an example of a non-resource-driven discrete activity which has incorporated a number of realistic factors. In this example, Activity A-Placement Concrete cannot start until Activity B is 50% finished and Activity C is completely finished, plus that the temperature has to be above 40 degrees. When Activity A is progressing, if the temperature falls to below 40°F, it will have to pause for 1 day. The construction of a nearby Activity G will also impact Activity A, causing its productivity to decrease to .7 of the normal speed. After Activity A is 70% finished, Activity F can start. And after Activity A is completely finished, Activity D will be performed if the time is early enough (<30 days), otherwise Activity E will be performed

as a remedy plan. The estimated duration of Activity A is N (20, 4) [1], and its progress curve is depicted in Figure 5-5.

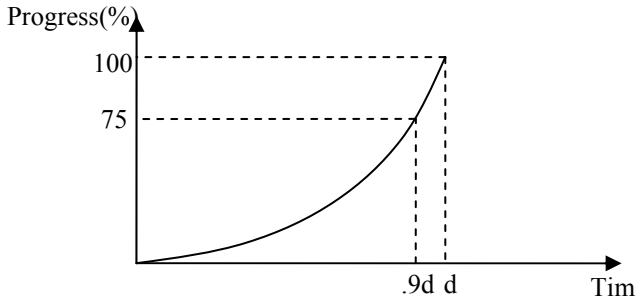


Figure 5-5. Progress curve of Activity A

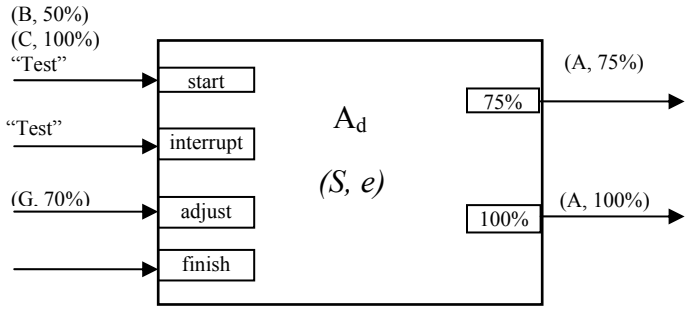


Figure 5-6. Non-resource-driven discrete model of Activity A

The structure of the non-resource-driven discrete model for Activity A is illustrated in Figure 5-6. This structure can be described as:

$$A_d = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta), \qquad \text{(Equation 5-3)}$$

in which the elements are defined as:

- ***State S.*** The state of the non-resource-driven discrete activity is represented with a set of parameters and state variables. The *parameters* include:

  name. This is the unique identifier of the activity.

[1] N (10, 2): Sample from a normal distribution with the mean of 20 days and the standard deviation of 4 days.

work_quantity. The total quantity of work involved.

duration. The estimated duration for completing the activity from start to finish, without any interruptions.

progress_function. The progress function *progress = p (duration, time)* describes the progress curve of the activity, i.e., how the activity would proceed from start to finish over time without external interruptions.

start_constraint. This is a compound logical expression that represents the total constraints on the start of the activity. Each start dependency relationship is associated with one logical variable *con (predecessor_name, predecessor_progress),* whose initial value is set to "False". The time constraints and the environmental constraints on the start of the activity are represented as logical expressions. These logical variables and logical expressions are connected with the "AND" and "OR" operators in the *start_constraint*. For example, the *start_constraint* for Activity A is "con(B, 50%) AND con(C, 100%) AND (tem>40)", whose initial value is "False AND False AND (tem>40)".

finish_constraint. A compound logical expression representing the total constraints on the finish of the activity.

interruptions. Interruptions are defined by *{(interrupt_condition*, *interrupt_length*, *resume_condition)}*, where *interrupt_condition* represents a condition that will trigger an interruption, *interrupt_length* specifies the length of that interruption, and *resume_condition* specifies the condition under which the activity will be resumed if the length of the interruption is not provided. The *interruptions* for Activity A is {(tem<40, 1, ,)}.

adjustments. Adjustments to the duration of the activity are defined by *{(adjust_condition, adjust_factor)}*, where the *adjust_condition* represents a condition that will cause the duration of the activity to change, and the *adjust_factor* specifies the adjustment factor that will be applied. The adjustments for Activity A is represented by {(con(G, 0%), 1.4), (con(G, 100%), 1.0)}.

markers. A marker is a point on the activity's progress where it needs to generate an output. Each dependency relationship imposed by the activity is associated with a marker. The markers of the activity are sorted in ascending orders in a list called the *markers.* Activity A' *markers* are {75%, 100%}.

The *state variables* include:

phase. A discrete activity could be in one of the four phases at any point in time:

"starting": the activity is waiting for the start constraints to be satisfied.

"progressing": the activity is proceeding according to the specified progress curve.

"interrupted": the activity has been interrupted during the middle of its progress, waiting to be resumed again.

"finished"*:* the activity has been 100% completed, and all constraints on the finish of the activity have been satisfied.

transition_progress. A percentage that records the progress of the activity when it enters into the current phase.

a. This is the adjustment factor which is going to be applied to the *duration* of the activity to determine the *adjusted duration*.

δ. A non-negative real number that records the time remaining in the current state — with the absence of any external events. In other words, the activity will stay in the current state for the time given by the state variable *δ*.

The initial state of the activity is set to S *(phase, transition_progress, a, δ)* = ("starting", 0%, 1, ∞).

- ***Total State Q***. The total state of the activity is represented by $\{(s, e) \mid s \in S, 0<e<ta(s)\}$, where *e* is the time elapsed since the last state transition.

- ***Progress.*** The *progress* of the activity at any point in time can always be derived from its total state Q.

$$progress = \begin{cases} 0\% & \text{if } phase = \text{"starting"}; \\ transition\_progress & \text{if } phase = \text{"interrupted"}; \\ p(a \times duration, (p^{-1}(a \times duration, transition\_progress) + e)) & \text{if } phase = \text{"progressing"}; \\ 100\% & \text{if } phase = \text{"finished"}. \end{cases}$$

When the activity is "starting", its progress certainly is 0%. When the activity is in the "finished" phase, the progress must be100%. If the activity is resting in the "interrupted" phase, its progress will remain unchanged at the value when it being stopped, i.e., the value of *transition_progress*.

If the activity is under the "progressing" phase, its progress can be determined by its *transition_progress*, the adjustment factor *a,* and the time elapsed since it enters the current state, *e*. As shown in Figure 5-7, suppose that Q = {"progressing", 20%,1.4, 2}, i.e., starting from 20%, the activity has proceeded 2 days with a reduced productivity of 70%. According to the progress curve, $p^{-1}(1.4 \times 10, 20\%) = 5.7$, i.e. the starting point corresponds to Day 5.7 on the current progress curve. So if the work has proceeded 2 days without any interventions, the progress should be $p\ (1.4 \times 10, 7.7) = 35\%$ at the current point.

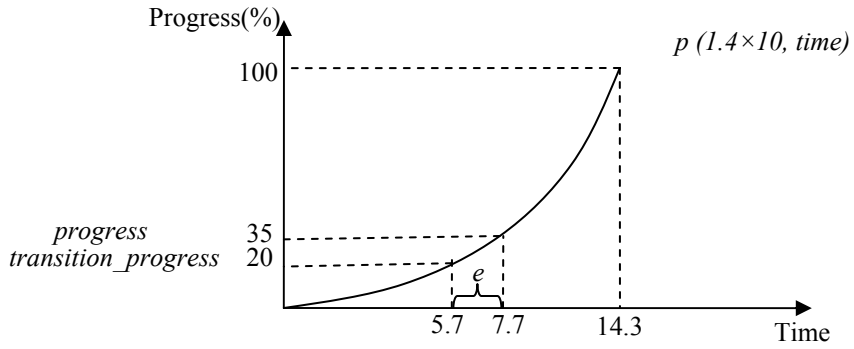


Figure 5-7. Determining the progress during the "progressing" phase

- ***Inputs X***. As shown in Figure 5-6, each non-resource-driven discrete activity model has four input ports, "start", "interrupt", "adjust", and "finish". The input messages could be sent from its predecessor activities in the form of *(predecessor_name, predecessor_progress)*, or the "Test" signals from the influencing environmental factors.

  $InPorts = \{$"start", "interrupt", "adjust", "finish"$\}$,
  $X_{start} = X_{interrupt} = X_{adjust} = X_{finish} =$
  $\{$(predecessor_name, predecessor_progress)$\} \cup$ "Test",
  $X = \{(p,v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values.

- ***Outputs Y.*** The atomic non-resource driven activity model has one output port for every marker. Therefore, Activity A has two output ports: "75%" for the dependency relationship "when A is 75% finished" and "100%" for "when A is 100% finished". The value of the output is a message consisting of the name of the activity and the progress of the activity.

  $OutPorts = \{markers(1), markers(2), \ldots markers(i), \ldots, markers(n)\}$
  $Y_{markers(i)} = \{(name, progress)\}$
  $Y = \{(p,v) \mid p \in OutPorts, v \in Y_{markers(i)}\}$ is the set of output ports and values.

- ***Internal Transition Functions δ_{int}: S→S***. If no external event occurs, the activity will stay in state S for time $ta(s)$. When the resting time in state S expires, i.e., when the elapsed time $e=ta(s)$, the activity outputs the value $\lambda(s)$ and changes to state $\delta_{int}(s)$.

  Under the "starting" and the "finished" phase, $\delta = \infty$, which means that the activity will stay in a passive state until being changed by external events.

  During the "progressing" phase, the activity will proceed from one mark to the next, unless it has reached the 100% marker, in which case it will rest and wait for the finishing constraints to be satisfied.

  δ_{int}("progressing", transition_progress, a, δ)

$= (\text{“progressing”}, next(markers), a, (p^{-1}(a \times duration, next(markers)) - p^{-1}(a \times duration, transition\_progress))$ if $next(markers) \neq 100\%$;
$= (\text{“progressing”}, 100\%, a, \infty)$ if $next(markers) = 100\%$.

If the activity has been interrupted, when the time specified for the “interrupted” phase expires, the activity will return to the “progressing” phase, and continue to proceed towards the next marker.

$\delta_{int}(\text{“interrupted”}, transition\_progress, a, \delta)$
$= (\text{“progressing”}, transition\_progress, (p^{-1}(a \times duration, next(markers)) - p^{-1}(a \times duration, transition\_progress))$.

- ***Output Function $\lambda: S \rightarrow Y$***. Outputs are generated just before the internal transitions. Activities will generate outputs only when they are under the “progressing” phase.

  $\lambda(\text{“progressing”}, transition\_progress, a, \delta) = (\text{“next(markers)”}, (name, next(markers))$

  Suppose that S = (“progressing”, 0%, 1, 9). When $e = 9$, Activity A will generate a message (A, 75%), and post it on the output port “75%”. After that, Activity A will transit into the state δ_{int} (“progressing”, 0%, 1, 9) = (“progressing”, 75%, 1, 1).

- ***External state transition functions $\delta_{ext}: Q \times X \rightarrow S$***. If an input X comes before the expiration time, i.e., when the activity is in total state s (e) with $e = ta(s)$, the state of the activity will changes to state $\delta_{ext}(s)$.

  When a message *(predecessor_name, predecessor_progress)* is received on one of the input ports, the value of the associated logical variable *con (predecessor_name, predecessor_progress)* will change to “True”. After that, the constraint related with that port (*start_constraint* for the “start” port, *finish_constraint* for the “finish” port, *adjustment _condition* for the “adjust” port, *interrupt_condition* for the “interrupt” port during the “progressing” phase, and *resume_condition* for the “interrupt” port during the “interrupted” phase) will be evaluated.

  If the input is a “Test” message sent by an environmental factor, the activity will directly evaluate the value of the constraint.

  For example, if Activity A receives (B, 50%) on its input port “start” during the “starting” phase, the value of *con(B, 50%)* will be turned to “True” and thereby the *start_constraint* “con(B, 50%) AND con(C, 100%) AND (tem>40)” will become “True AND con(C, 100%) AND (tem>40)”. Suppose that Activity A receives a “Test” message on its “start” input port, the *start_constraint* of Activity A will not change; the value of the variable *temp* need to be read dynamically every time when the *start_constraint* is evaluated.

  Messages sent by the predecessor activities and the environmental factors are treated differently in this procedure. The progress of the activities cannot move backwards, therefore once the predecessor activity reaches the specified progress, the constraint it imposed on its successor activity can be permanently removed, which means that the value

of the associated variable *con (predecessor_name, predecessor_progress)* can be changed to "True". Whereas the values of the environmental factors may vary in both directions: "tem>40" at this moment does not guarantee it will still be true at the next moment. Therefore the constraints imposed by the environmental factors need to be re-evaluated every time.

If the evaluation result of the associated constraint is "False", the activity will remain in its current state, except that δ, the time remaining, will be updated to $(\delta\text{-}e)$; otherwise, the activity will transit into a different phase as defined in the following external transition functions:

δ_{ext} *(phase, transition_progress, a,* δ*, e, X)*
= *("progressing", 0%, a,* $p^{-1}(a \times duration, next(markers))$
if *phase = "starting"* and *InPort = "start"* and *start_condition = "True";*
//Transit from the "starting" phase to the "progressing" phase, reaching the first marker in time $p^{-1}(a \times duration, next(markers))$.

= *("interrupted", progress, a, interrupt_length)*
if *phase = "progressing"* and *InPort = "interrupt"* and
interrupt_condition(i) = "True" and *interrupt_length(i)* $\neq 0$*;*
//Transit from the "progressing" phase to the "interrupted" phase, set *transition_progress* to the current value of *progress*, and rest for time *interrupt_length(i)*.

= *("interrupted", progress, a,* ∞*)*
if *phase = "progressing"* and *InPort = "interrupt"* and
interrupt_condition(i) = "True" and *resume_condition(i)*$\neq \phi$ *;*
//Transit from the "progressing" phase to the "interrupted" phase, set *transition_progress* to the current value of *progress*, and rest in a passive state.

= *("progressing", transition_progress, a,* $p^{-1}(a \times duration, next(markers)) - p^{-1}(a \times duration, transition\_progress)$ *)*
if *phase = "interrupted"* and *InPort = "interrupt"* and *resume_condition(i) = "True"*
//Transit from the "interrupted" phase to the "progressing" phase, without changing the value of the *transition_progress*, continuing to proceed toward the next marker.

= *(phase, progress, adjust_factor(i),* δ*-e)*
if *phase* $\neq$ *"progressing"* and *InPort = "adjust"* and *adjust_condition(i) = "True"*
// Remain in the original phase, set the value of *a* to the *adjust_factor(i)*, update the value of *transition_progress* and δ.

= *("progressing", progress, adjust_factor,* $p^{-1}(adjust\_factor \times duration, next(markers)) - p^{-1}(adjust\_factor \times duration, progress)$*)*

if *phase = "progressing"* and *InPort = "adjust"* and *adjust_condition(i) = "True"*

// Remain in the original phase, apply the *adjust_factor(i)* to *a*, update the value of *transition_progress* and then recalculate the time to the next marker.

= ("finished", 100%, a, ∞)

if *phase = "progressing"* and *InPort = "finish"* and *finish_condition = "True"* and *transition_progress = 100%*

// Transit into the "finished" phase, and rest in a passive state (or can be terminated after this point).

- ***Time Advance Function ta(s).*** The time remaining in current state is defined as the value of the state variable δ.

5.2.1.2 A simulation of a non-resource-driven discrete model

A simulation of Activity A's behavior is shown in Figure 5-8.

(1) At initialization, Activity A is in a passive state *(phase, transition_progress, a, δ)* = ("staring", 0%, 1, ∞); and the values of all constraints associated with its input ports are set to "False". Then the message (B, 50%) is received on its "start" input port. The external transition function δ_{ext} is triggered and the value of the logical variable *con (B, 50%)* is changed from the initial "False" into "True". The *start_constraint* "con (B, 50%) AND con(C, 100%) AND temp >40" thus becomes "True AND False AND temp>40", whose value is still "False", so the activity will continue to stay in the "starting" phase with $\delta=\infty$.

Later the temperature rises to above 40°F and a "Test" message is received on the "start" input port. Again the external transition function is evoked, and the *start_constraint* is evaluated. Since the value is still "False", the state of the activity does not change.

After a period of time, the message "(C, 100%)" is received on the "start" input port, the external transition function changes the value of the logical variable *con(C, 100%)* to "True" and thereby the value of the *start_constraint* becomes "True AND True and (temp>40)". Suppose at this point the value of the variable *temp* at this point is below 40°F, the activity still needs to wait.

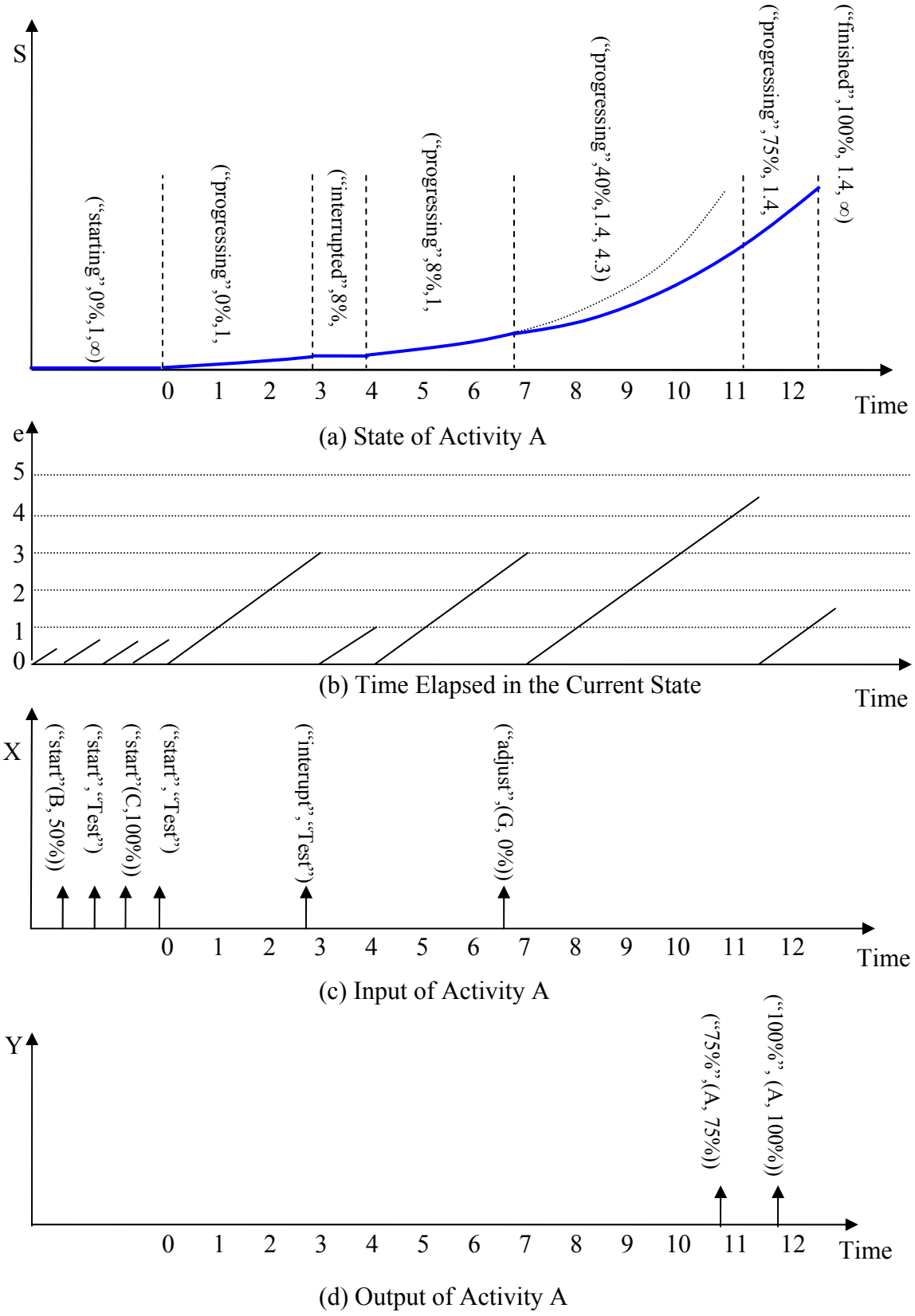


Figure 5-8. Simulation of Activity A's Behavior

(2) At time 0, when temperature rises to above 40°F again, another "Test" message is received on the "Start" port. This time the value of the *start_constraint* is "True" and the state of the activity changes to:

δ_{ext} *("starting", 0%, a, ∞, e, X)*
= ("progressing", 0%, a, p^{-1}*(a×duration, next(markers)*
= ("progressing", 0%, 1, p^{-1}*(10, 75%)*
= ("progressing", 0%, 1, 9),

and the value of *e* is set to 0.

If no external events occur, when the resting time expires, i.e., $e = 9$, the state of Activity A will change according to the following internal transition function:

δ_{int}*("progressing", 0%, 1, 9)*
= ("progressing", next (markers), a, (p^{-1}*(a×duration, next (markers) -* p^{-1} *(a×duration, transition_progress))*
= ("progressing", 75%, 1, (p^{-1} *(100%)-*p^{-1}*(10, 75%)).*

(3) However, at time $t = 3$ days before this resting time expires, the temperature drops to below 40 degrees and a "Test" message is received on the activity's "Interrupt" input port. The internal transition is interrupted, and the values of the *interrupt_conditions* are checked. As the condition "temp>40" have become true, the state of the activity changes according to the following external transition function:

δ_{ext} *("Progressing", transition_progress, a, δ, e, X)*
= ("interrupted", progress, a, interrupt_length)
= ("interrupted", 8%, 1, 1).

Note that the progress 8% is derived from *S*, the state of the activity, and *e*, the time lapsed since the last transition at the point just before the occurrence of this external transition. Now it has become the new *transition_progress* of the new state. The value of *e* is reset to 0.

(4) The activity stays in the state *("interrupted", 8%, 1, 1)* until $e = \delta = 1$, i.e., at time $t = 4$ days, the following internal transition takes place:

δ_{int} *("interrupted", 8%, 1, 1)*
= *("progressing", transition_progress, a,* $(p^{-1}(a \times duration, next(markers)) - p^{-1}(a \times duration,$ *transition_progress))*
= *("Progressing", 8%, 1,* $(p^{-1}(10, 75\%) - p^{-1}(10, 8\%))$
= *("Progressing", 8%, 1, 6),*

which means that the activity has reentered the "progressing" phase, and it will reach the next marker in 6 days if no external events intervenes.

(5) At time $t = 7$ days, when $e = 3$, the above internal transition is stopped because Activity G has started and a message (G, 0%) has been received on the "adjust" input port. The value of the logical variable *con(G, 0%)* is turned into "True"; and after that, the constraints specified in the parameter *adjustment* = {(con(G, 0%), 1.4), (con(G, 100%), 1.0)} are checked. Since one of the adjustment conditions, *con(G, 0%)*, has become true, the following external transition is triggered and the value of e is reset to 0:

δ_{ext} *("progressing", transition_progress, a,* δ*, e, X)*
= *("progressing", progress, 1/adjust_factor,* $p^{-1}(a \times duration, next(markers)) - p^{-1}(a \times duration, progress))$
= *("progressing", 40%, 1.43,* $p^{-1}(1.43 \times 10, 75\%) - p^{-1}(1.43 \times 10, 40\%)$*)*
= *("progressing", 40%, 1.43, 4.3).*

(6) At time $t = 11.3$ days, $e = 4.3 = \delta$, the resting time expires and an internal transition is going to take place. Just before the internal transition, the activity generates an output according to the following output function:

λ *("progressing", transition_progress, a,* δ*)*
= *("next(markers)", (name, next(markers))*
= *("75%", (A, 75%)),*

which means that the message (A, 75%) is posted to the output port "75%". Afterwards, the following internal transition occurs:

δ_{int}(“progressing”, 40%, 1.43, 4.3)
= (“progressing”, 75%, 1.43, (p-1(1.43×10,100%)-p-1(1.43×10, 75%))
= (“progressing”, 75%, 1.43, 1.43).

(7) Finally, at time t = 12.73 days when the activity approaches the last maker 100%, another output is generated:

λ (“progressing”, 75%, 1.43, 1.43)= (“100%”, (A, 100%)),

followed by the internal transition:

δ_{int}(“progressing”, 75%, 1.43, 4.3)= (“progressing”, 100%, 1.43, ∞).

At this point, Activity A is completed and will rest in a passive state.

5.2.1.3 Resource-driven discrete models

The second type of the atomic activity model is the resource-driven discrete activity model $Activity_{dr.}$ With this model, the duration of the activity is determined dynamically according to the type and quantity of the resource that can be obtained. The activity will request resources after its *start_contraint* has been satisfied, and release resources when it has been finished. If the activity is interrupted during the middle, resources can be returned to the resource pool and used by other activities.

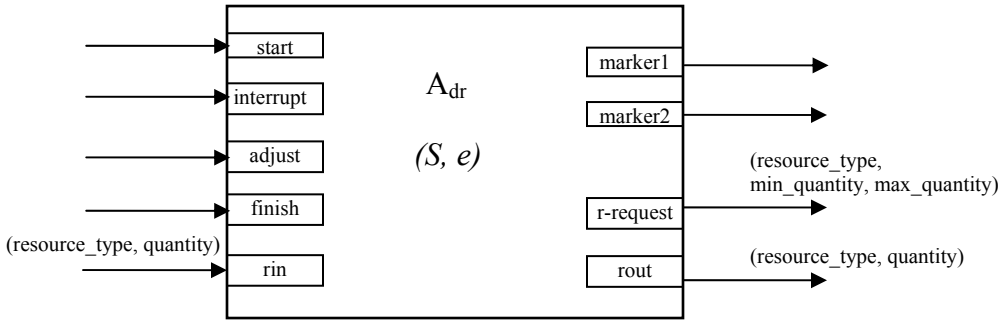


Figure 5-9. Structure of the resource-driven discrete activity model

The basic structure of the resource-driven discrete activity model $Activity_{dr}$ is shown in Figure 5-9. It is similar to the structure of the non-resource-driven activity model $Activity_{d}$, with the following differences:

- ***State S.*** In the $Activity_{dr}$ model, *duration* is not an independent parameter; instead, it is a derived state variable whose value is dependent upon the parameter *work_quantity*, and the *productivity* and *quantity* of the acquired resource.

  The parameter *required_resource = {(resource_type, min_quantity, max_quantity, productivity, priority)}* represents the type, the lower limit and upper limit of the required quantity, the productivity, and the priority of the resource if there are a number of different types of resources that can be used to finish this activity.

  The state variable *resource = (resource_type, quantity)* is used to represent the type and quantity of the resource that has been obtained by the activity.

  The state variable *phase* has three additional values compared to that of the non resource-driven model: "requesting_resource", during which the activity is attempting to obtain the required resources to either start or resume the work; "resource_obtained", during which the required resources have been acquired and the activity is sending messages to clear previously submitted requests that are still waiting in resource request queues; and "releasing_resource", during which the activity is releasing the engaged resources.

  The parameter *interruptions* in the $Activity_{dr}$ model is defined as {*(interrupt_condition*, *interrupt_length*, *resume_condition, release_resource)*}. The parameter *release_resource* is a logical variable that defines whether the engaged resources shall be released when the activity is interrupted by the specified *interrupt_condition*.

- ***Input.*** The model $Activity_{dr}$ has an additional input port "rin" which is to receive resources input *(resource_type, quantity)*.

- ***Output.*** The activity will post the request for resources *(resource_type, min-quantity, max_type)* to its output port "r-request" and release the disengaged resources *(resource_type, quantity)* through the output port "rout".

- ***External Transitions, Internal Transitions and Output Functions.***

  (1) Start-up of the activity.

  When the activity's start constraints have been satisfied, the activity will transit via the following external transition function:

  δ_{ext}*("starting", 0%, a, ∞, e, X) = ("requesting-resources", 0%, a, 0).*

  From this state the activity will take the following internal transition:

δ_{int} *("requesting-resources", 0%, a, 0)= ("requesting-resources", 0%, a, ∞)*,

i.e., the activity will put itself into a passive state, waiting for the arrival of the required resources. But right before the internal transition, an output will be produced:

λ ("requesting-resources", 0%, a, 0)
= ("r-request", (activity_name, resource_type, min_quantity, max_quantity)).

The activity posts the message *(activity_name, resource_type, min_quantity, max_quantity)* to its "r-request" output. This massage contains the name of the activity that is sending the request, the type of the resource with the highest priority in its parameter *required-resource*, and the range of the quantity required.

This message will be send to the resource, which will return a message *(activity_name, resource_type, quantity)* to the "rin" port of the activity in response.

A request that cannot be satisfied outright will be saved in the waiting queue of the resource, and the *quantity* in the returned message will be set to 0. Upon receiving such a message, the activity will check if there are any alternative resources listed in its *required-resource*. If not, it will stay in the passive state, waiting for the requests in the waiting queues to be handled and fulfilled; otherwise, the passive state of the activity will be ended, as δ will be changed from ∞ to 0 as in the following external transition function:

δ_{ext} *("requesting-resources", 0%, a, ∞, e, X) = ("requesting-resources", 0%, a, 0).*

Under this state, the activity can send a new request to the resource with the second highest priority in its required-resource. The above processes will be repeated until the activity receives a message with a non-zero *quantity* on its "rin" port, upon which, the activity will be changed into a temporary "resource_obtained" phase via:

δ_{ext} *("requesting-resources", 0%, a, ∞, e, X) = ("resource_obtained", 0%, a, 0).*

From this state, the activity will start progressing toward the next immediate marker:

δ_{int} *("resources-obtained", 0%, a, 0)*
= ("progressing", 0%, a, p^{-1}*(a×duration, next(markers))-* p^{-1}*(a×duration, progress)*,

in which the duration is calculated as:

$$duration = work\_quantity/(productivity \times quantity).$$

Right before the above internal transition, the activity produces an output to clear all of its previous requests stored in the waiting queues of related resources:

λ ("resource-obtained", transition_progress, a, 0)
= ("r-request", (activity_name, resource_type, 0, 0)).

(2) Finishing of the activity.

When the activity is finished, i.e., after the activity's progress has approached 100% and the finish constraints have all been satisfied, the activity will transit into a temporary state "releasing-resource", during which it will release the engaged resources, and then continue to transit into the "finished" state:

δ_{int}*("releasing-resource", 100%, a, 0) = ("finished", 100%, a, ∞),*
λ ("releasing-resource", 100%, a, δ)= ("rout", (resource_type, quantity)).

(3) Interruptions and resumptions.

If the activity is interrupted with the option "release resources" setting to be true, the activity will first enter into the "releasing-resource" state and produce an output to export the resources before it enters into the "interrupted" state:

δ_{int}*("progressing", transition_progress, a, δ)*
= ("releasing-resource", progress, a, 0);
λ ("releasing-resource", transition_progress, a, δ)
= ("rout", (resource_type, quantity));
δ_{int}*("releasing-resource", transition_progress, a, δ)*
= ("interrupted", transition_progress, a, interrupt_length (or ∞)).

When the condition for resumption is satisfied, the activity will first enter the "requesting-resource" phase and then the "resource-obtained" phase before it can proceed, similar to the procedure at the start up of the activity.

5.2.1.4 Continuous models

As shown in Figure 5-10, the structure of the continuous activity model $Activity_c$ has a lot of similarities with that of the resource-driven discrete model $Activity_{dr}$. They have the same input and output ports, and under most phases their external transition and internal transitions are essentially the same. The major difference between the two lies in the "progressing" phase.

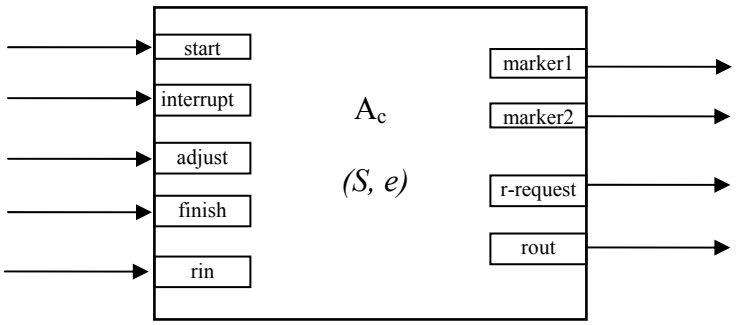


Figure 5-10. Structure of the continuous activity

As shown in Figure 5-11, the continuous activity does not transit directly from one marker to the next during the "progressing" phase, but advances gradually at fixed time steps. At every step, the value of the productivity will be re-sampled, and the *modifiers* (modification factors determined by the values of dynamically changing variables) will be reapplied.

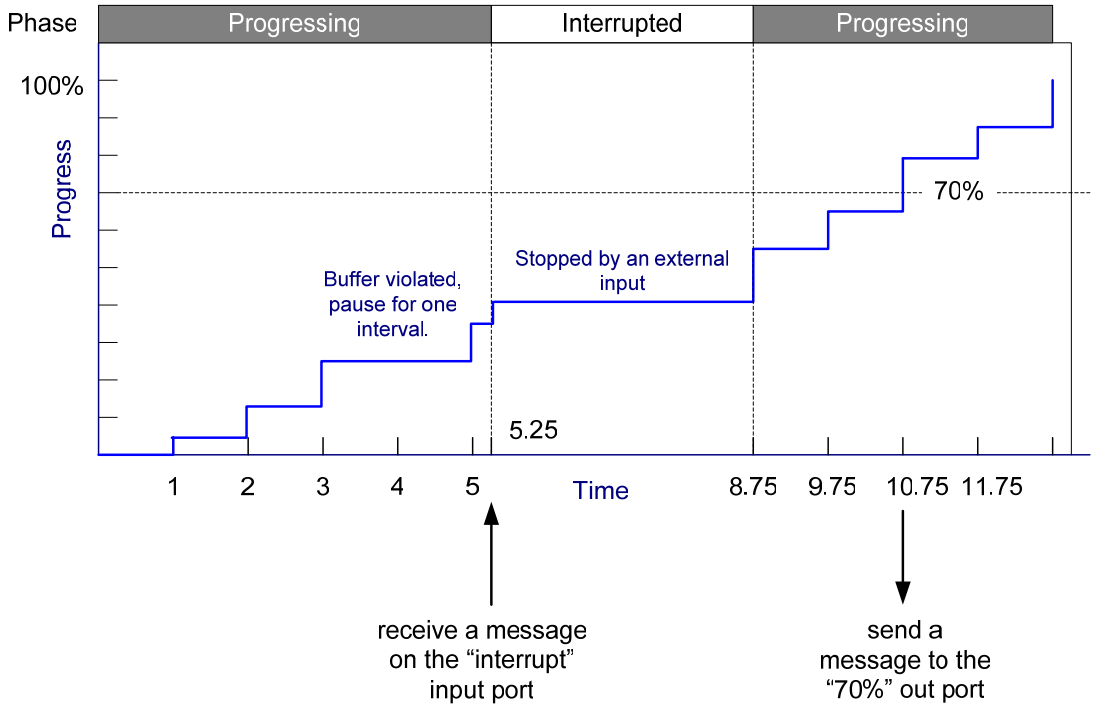


Figure 5-11. Process of a continuous activity

If the activity steps across a marker, it will generate an output at the end of that step. The activity in Figure 5-11 reaches 70% between time 9.75 and 10.75. The output is produced at time 10.75.

If the activity is going to violate the buffer constraint after taking a step, it could pause for that interval (approximating the "slow-down" option) or stop completely (modeling the "break" option). Figure 5-10 shows a pause at time 4. Note that the activity is still under the "progressing" phase, for it will continue to attempt to advance at the next step. If the option

"break" is selected instead of "slow-down", the activity will be put into the "interrupted" phase for a specified length of time or until being resumed by an external input.

External inputs could also cause the continuous activity to stop or to adjust the productivity rate. The external inputs are processed immediately upon their arrival. In Figure 5-11, during the middle of a step at time 5.25, the activity receives a message on its "interrupt" input port and is stopped, with its progress increased by (.25 × length of the interval × productivity). When it is resumed at time 8.75, it continues to advance one interval a time.

Listed below are the state parameters and variables specific to the continuous activity model, together with its internal tansitions and output functions during the "progressing" phase. Since its external transition functions are essentially the same as those of the discrete model, they are not presented here.

- ***State.*** The continuous activity has three unique parameters — Δ, *buffer* and *modifiers*. The definition of the parameter *start_constraint* is also different from the discrete activity models.

  Δ. This parameter *interval* specifies the length of the fixed time step at which the state of the activity is going to be updated under the "progressing" phase.

  buffer. The parameter *buffer* consists of *{(buffer_constraint, pause, pause_period, release_ resource)}*. The *buffer_constraint* is a logical expression that may contain any variables and the "AND" and "OR" operators. When the value of the *buffer_constraint* becomes "True", the activity will pause for the current interval and attempt to continue at the next step (to approximate the "slow-down" option), provided that the value of *pause* is 0; if the value of *pause* is 1, it will stop for the length specified by *pause_period* (to represent the "break" option). The value of *release_resource* specifies whether the engaged resources will be released during the break.

  modifier. The modifier is a function $f(m_i)$ ($i=1$ *to* n) that are dependent upon dynamically changing factors m_i. The modified productivity ($f(m_1, m_2 \ldots m_n) \times$ *productivity)* is denoted as *m_productivity*.

  start_constraint. The *buffer_constraint* will be combined with the constraints explicitly defined as the start constraints to control the start of the continuous activity. If there are more than one buffer constraints, all of them will be connected with "AND" into the *start_constraint*.

- ***Internal Transitions.*** The internal transition under the "progressing" phase for the continuous activity model is completely different from that of other two types of activity models, and it is described as below:

  δ_{int} *("progressing", progress, a, δ) =("releasing-resource", 100%, a, 0)*
  If (*progress* + *a*× *m_productivity*×Δ) ≥ 100%;
  // The progress of the activity is going to exceed 100% after taking a full step. The activity transits into the "releasing-resource" phase, which will be followed by the "finished" phased.

  = ("progressing", progress + a×*m_ productivity*×Δ*, a, Δ)*
  If *buffer_constraint = "False"* and (*progress* + *a*× *m_productivity*×Δ) <100%*;*
  //Buffer not violated, the activity's progress advances by *(a*× *m_productivity*×Δ), and will remain at this level during the next time interval.

  = ("progressing", progress, a, Δ)
  If *buffer_constraint = "True"* and *pause = 0* and (*progress* + *a*×*m_productivity*×Δ) <100%*;*
  //Buffer violated. The option "slow-down" is selected. The activity will hold the current step, but it will remain in the "progressing" phase and continue to attempt to proceed at the next step.

  = ("interrupted", progress, a, pause_period)
  If *buffer_constraint= "True"* and *pause=1* and *release_ resource=0* and (*progress* + *a*× *m_productivity*×Δ) <100%*;*
  //Buffer violated. The "break" option is selected and the parameter *release_resource* is set to "0". The activity's progress will not increase, and it will be put into the "interrupted" phase for the specified *pause_period*. When the *pause_period* expires, the activity will re-enter the "progressing" phase via the internal transition function δ_{int} *("interrupted", progress, a, pause_period) = ("progressing", progress, a, Δ).*

  =("release-resource", progress, a, pause_period)
  If *buffer_constraint= "True"* and *pause=1* and *release_ resource=1* and (*progress* + *a*× *productivity*×Δ) <100%*;*
  //Buffer violated. The "break" option is selected and the parameter *release_resource* is set to "1". The progress will not change. The activity will enter into the "releasing-resource" state and releases the engaged resources via *λ ("releasing-resource", progress, a, 0) = ("rout", (resource_type, quantity))*. Then it will rest in the "interrupted" state for the specified *pause_period*. When the *pause_period* expires, it has to go through the "requesting-resource" and the "resource-obtained" phases before re-entering the "progressing" state.

- ***Output function.*** At the end of every step, the output function will check whether the activity has crossed a marker and produce the outputs accordingly.

  λ ("progressing", progress, a, δ)

= *("marker1", (activity_name, marker1))*
if *progress* < *marker1* and *(progress* + *a*×*m_ productivity*×Δ) > marker 1*;*
("marker2", (activity_name, marker2))
if *progress* < *marker2* and *(progress* + *a*× *m_productivity*×Δ)> marker 2*;*
....
("markerN", (activity_name, markerN))
if *progress* < *markerN* and *(progress* + *a*× *m_productivity*×Δ)> marker N.

To summarize, the progress of the continuous model are controlled by two types of constraints: the constraints that comes through the input ports, which may trigger a one-time response; and the constraints defined as the buffer constraints, which will be evaluated every time the continuous activity advances. By contrast, the discrete models only allow the first type of control.

5.2.2 Compound Activity Models

A group of atomic activity models can be coupled to construct a compound activity model. The compound model could be discrete and non-resource-driven, denoted as $Activity_{cd}$; or discrete and resource-driven, $Activity_{cdr}$; or continuous, $Activity_{cc}$. The compound activity models have the same sets of ports and external behaviors as their atomic counterparts. They can be used just as the atomic models in the construction of higher-level models, thus providing a closed-under-composition hierarchical structure.

5.2.2.1 Compound discrete non-resource-driven models

The structure of the compound discrete non-resource-driven model is defined as:

$$A_{cd} = (X, Y, D, EIC, EOC, IC), \qquad \text{(Equation 5-4)}$$

in which:

- ***X*** is the set of input ports and values. The input ports and the acceptable values on these ports are the same as those of the atomic model $A_{d.}$
- ***Y*** is the set of output ports and values. The output ports and generated values are the same as those of the atomic model $A_{d.}$

- ***D*** is the set of the component names. In the example shown in Figure 5-12, D = {A1, A2}. The components could be of any type of atomic activity models, including A_d, A_{dr} *and* A_c, as well as any type of compound activity models.

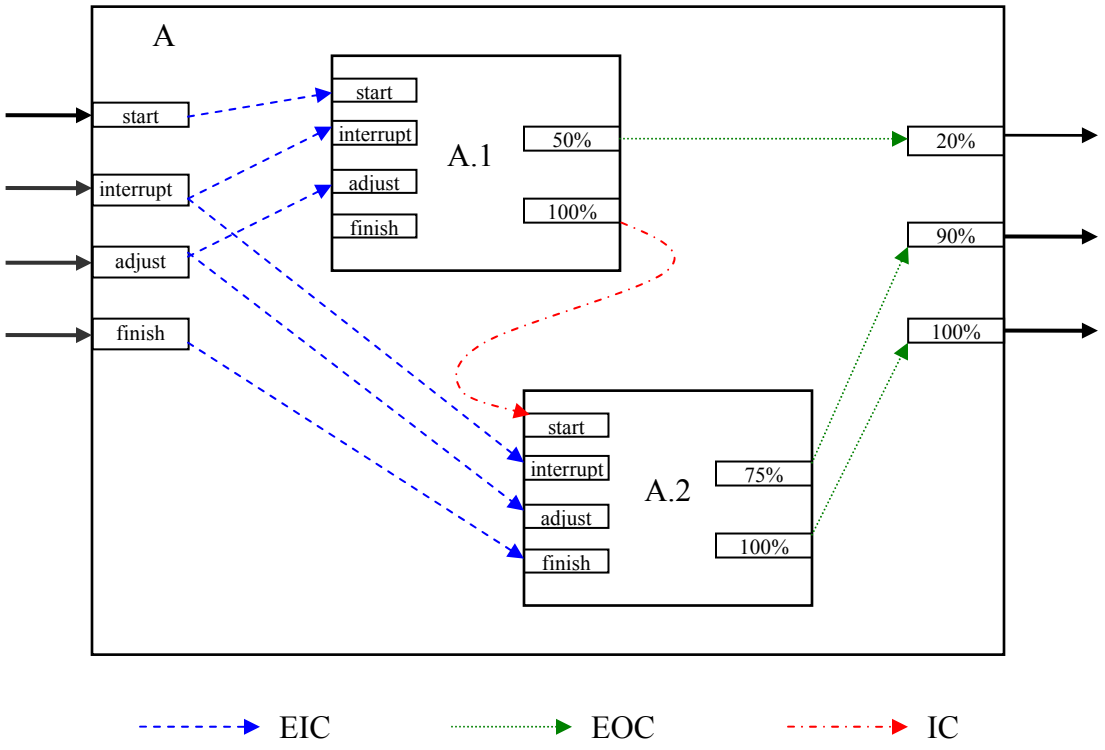


Figure 5-12. Example of the compound discrete non-resource-driven activity

- ***IC***, i.e., Internal Couplings, connect component outputs to component inputs. In Figure 5-11,

  IC = {(A1, "100 %"), (A2, "start")}.

- ***EIC***, i.e., External Input Couplings, connect external inputs to component inputs. The EIC connections are established according to the following rules in the proposed theory:

  External input on the "start" port will be duplicated and routed to the "start" ports of all components that do not have any start constraints;

  External input on the "interrupt", "resume" or "adjust" port will be duplicated and routed to the "interrupt", "resume" or "adjust" ports of all components;

  External input on the "finish" port will be duplicated and routed to the "finish" ports of all components that do not have successors in the components.

  Therefore, for the example shown in Figure 5-12,

EIC = {((A, "start"), (A1, "start")), ((A, "interrupt"), (A1, "interrupt")), ((A, "finish"), (A2, "finish"))}.

- ***EOC***, i.e., External Output Couplings, connect component outputs to external outputs. Only one EOC connection is established by default in the proposed theory — the connection between the "100%" output ports of the components that do not have any successors within the compound activity and the "100%" external output port. Other EOC connections have to be specified by the user. In the above example, the compound activity A needs to generate outputs when it is 20% and 90% completed. The user specifies, based on his or her judgment, that 20% completion of A is best signaled by 50% completion of its component A1, and 90% completion of A by 75% completion of A2.

  So the EOC for A is:

  EOC = {((A1, "50%"), (A, "20%")), ((A2, "75%"), (A, "90%")), ((A2, "100%"), (A, "100%"))}.

  If the progress of the compound activity needs to be defined with the progresses of more than one component, a virtual activity, whose duration is zero, needs to be created to implement the EOC connection. The outputs from the components will be sent to the virtual activity, which will send a message to the external output ports when all component outputs have been received.

5.2.2.2 Compound discrete resource-driven models

Compared to the non-resource-driven model A_{cd}, the resource driven model A_{cdr} has three extra ports: "r-request", "r-request" and "rout". The structures of the two models are similar, except for the EIC and EOC couplings rules involving the three extra ports. Figure 5-13 shows the resource-related couplings in a compound discrete resource-driven activity.

When a component posts a message to its "r-request" port, the message will be forwarded to the external "r-request" port. The message returned by the resource, upon arriving at the external "rin" port, will be routed to the "rin" port of the component according to the name of the activity it contained. After a component is finished, the released resource on the "rout" port will be sent to the external "rout" port.

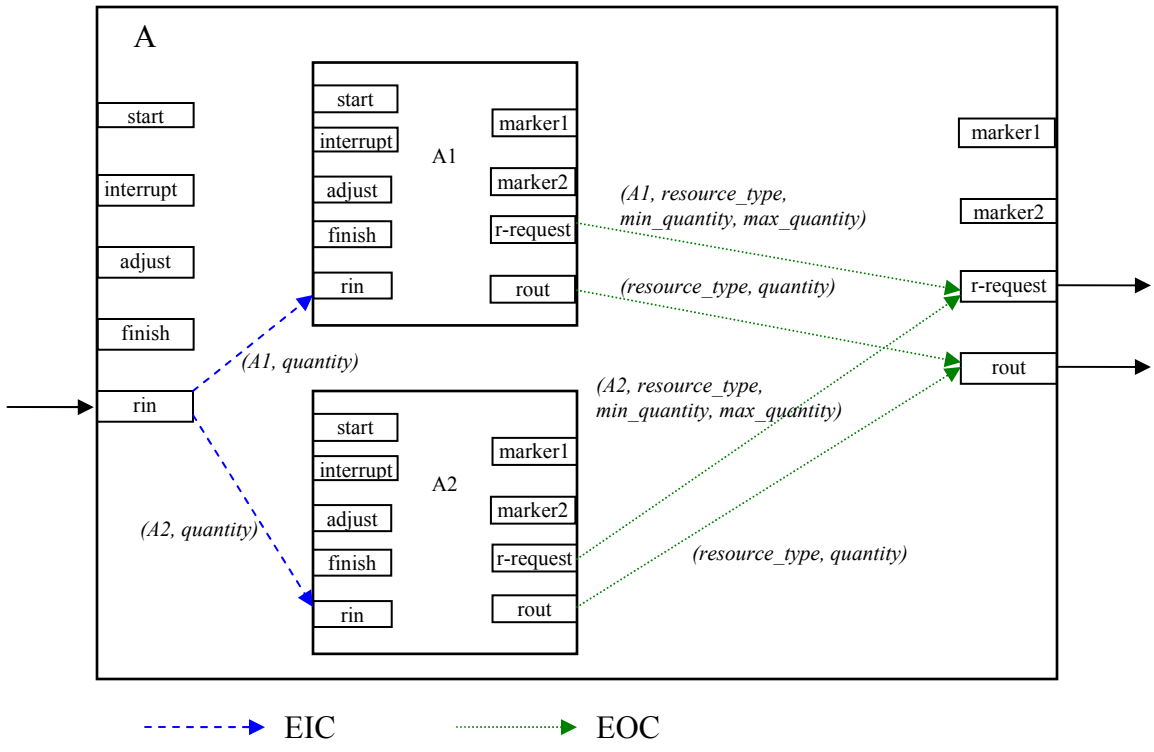


Figure 5-13. Example of a compound discrete resource-driven activity

5.2.2.3 Compound continuous models

As discussed in 5.2.1.4, the behaviors of the atomic continuous models can be impacted in two ways: one is through the external inputs, as with the discrete models; the other is through the variables in its buffer constraints. This is also true for the compound continuous model.

The inputs and outputs of the compound continuous model are routed according to the same rules as with the compound discrete model.

Buffers constraints in the compound continuous models are handled according to the following rules:

- Buffer constraints on a compound continuous activity will be imposed on all of its continuous components — i.e., all of its continuous components will have to maintain the external buffer constraints when they attempt to start and advance;
- Buffer constraints imposed by a compound continuous activity will be compiled to multiple buffer constraints, each imposed by one of its continuous components — i.e., the

successor activity has to maintain the buffer between all the continuous components of the compound activity that has started but not finished.

5.2.3 Resource Models

In the proposed theory, each type of resources is represented as a DEVS model. The structure of the resource model R is illustrated in Figure 5-14.

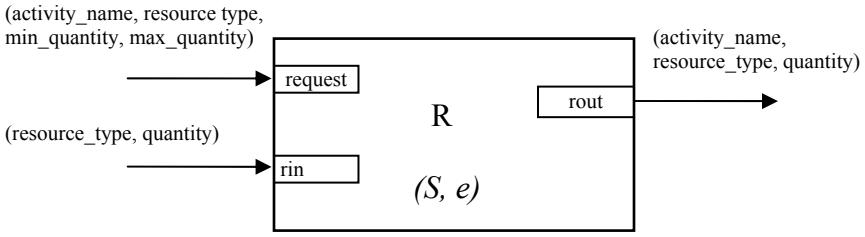


Figure 5-14. Structure of the resource model

$$R= (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta) \quad \text{(Equation 5-5)}$$

- ***Inputs X***. As shown in Figure 5-14, each resource model has two input ports: one is to receive the request sent by the activities; the other is to receive the released resources.

  InPorts = {"request", "rin"},
  $X_{request}$ *= {(activity_name, resource type, min_quantity, max_quantity)},*
  X_{rin} *= {(resource type, quantity)}.*

- ***Outputs Y***. The resource model only has one output port that is to export its responses to the requests from the activities.

  OutPorts = {"rout"},
  Y_{out} *= {(activity_name, resource type, quantity)}.*

- ***State S***. The state of the resource model is represented by the following parameters and variables:

  type. Resource type.

  total. The total amount that has been assigned to the project.

  available. The amount that is in the resource pool and can be used to satisfy the activities' requests.

  phase. The resource could be in one of the two phases: "passive" and "active".

activity_priority. This parameter consists of {(activity_name, priority)}. It stores the names of all the activities that might use the resource, and their respective priorities.

waiting_queue. This queue stores all the requests that have been received but cannot be fulfilled at the moment. It consists of {{(*activity_name, resource type, min_quantity, max_quantity)*}, ±1}. "+1"indicates that these requests are going to be sorted in the ascending order on their priorities, and "-1" indicates the opposite. For activities with equal priorities, they are going to be sorted on a first-come-first-serve basis.

- ***External Transitions, Internal Transitions and Output Functions.***

  At initiation, the resource is resting in the "passive" phase with ($\delta = \infty$). It will not make any internal transitions and outputs under this state.

  When the resource receives a request *(activity_name, resource type, min_quantity, max_quantity)* on its "request" port:

  If the maximum requirement can be fulfilled *(available ≥ max_quantity)*, the variable *available* will be reduced by the required maximum amount, and the resource will transit into the "active" phase, which is temporary ($\delta = 0$). During this phase, the resource outputs *(activity_name, resource_type, max_quantity)* through the "rout" port, and then returns to the passive state via the internal transition δ_{int} *("active", δ) = ("passive",* ∞).

  If the available quantity cannot fulfill the maximum requirement, but exceeds the minimum requirements *(available ≥ min_quantity and available < max_quantity),* the available quantity will decrease to 0 and the resource will transit into the temporary "active" state. After the output (activity_name, resource_type, available) is generated and sent to the "rout" port, it returns to the passive state.

  If the available quantity cannot meet the minimum requirement, the request will be inserted into the *waiting_queue*. The resource remains in the passive state.

  When the resource receives the message *(resource_type, quantity)* on its "rin" port:

  If the message is a "clean-up" message (*quantity* = 0), the corresponding request will be removed from the *waiting_queue*. The resource remains in its passive state.

  Otherwise, the value of the variable *available* will be increased by the released quantity, and the resource will be put into the "checking" phase by the external transition. During the "checking" phase, the following internal transitions take place: the resource finds the first request in its waiting queue (the requests are sorted on the specified priority in either the ascending order or the descending order) whose requirement can now be fulfilled, it then deduct the output amount from the available amount, self-transit into the "active" phase, generate the output, and come back into the "checking" phase to search for the next request that it can fulfill. When the search has reached the end of the queue, the resource will return to the "passive" phase and rest.

5.2.4 Models for Environmental Factors

Factors that change dynamically, such as temperature, precipitation and wind can be modeled with the environmental model *E*. An environmental model may be used to simulate more than one factor, provided that these factors can be updated with the same frequency.

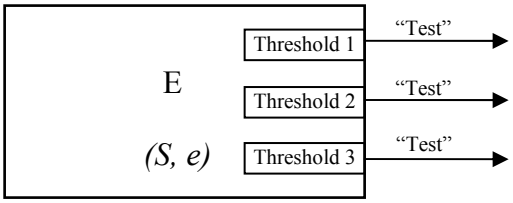


Figure 5-15. Structure of the environmental factor model

As shown in Figure 5-15, the environmental model has no input port, only output ports. The state of the model consists of the parameter *interval*, *variable_list {(variable_name, value)}* and *threshold_list {(variable_name, threshhold, direction)}*. The *values* of all variables are re-sampled and re-calculated at every time step. If the value of a variable exceeds a threshold (when *direction* = +1), or falls below a threshold (when *direction* = -1), a message "Test" will be generated and posted to the corresponding output port, and then sent to the connected activities.

Generating outputs at the thresholds is just one way that the environmental models could influence the activities. The values of the factors in the environmental models could also be read dynamically and thereby affect the duration, productivity and other properties or behaviors of the activities.

Besides the activities, resources and environmental models, there are a few accessory models in the proposed theory. One of them is the Delay model, which can take in any types of input messages, hold them for a specified length of time, and then pass them on to the designated destination. Another is the Branch model, which can route an input to one of its output ports depending on probabilistic selections or the value of a decision variable.

CHAPTER 6
MODELING ELEMENTS AND MODELING RULES

Chapter 5 introduced the fundamental theory proposed in this study — how each type of the models is structured, how they behave and how they interact. This chapter presents a new planning and scheduling method that was developed based on this theory. We will introduce the graphical modeling elements and the modeling rules in this method and show how they can be used to represent various situations in real-world complex construction projects.

The graphical representation of the proposed method is similar to the Activity-On-Node diagrams, but with many enhanced features. The nodes can represent different types of activities, resources and environmental factors; and the links can represent both logical constraints and resource flows.

In the following sections, we will first introduce the modeling of atomic activities, resources and environmental factors, then the links that connecting these basic components, and finally the construction of compound activities and repetitive activities.

6.1 Basic Components

6.1.1 Discrete Activities

As shown in Figure 6-1, in the network, discrete activities are represented as rectangles, with the name and the duration of the activity indicated in the middle. An activity should be modeled as a discrete activity if *all* of the following assumptions are true:

- The duration of the activity can be predicted fairly accurately at the time it starts. Should the duration change because of interruptions or adjustments, such changes should be rare.

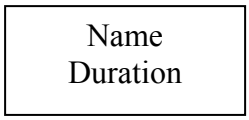


Figure 6-1. Discrete Activity Node

- Only the start and the finish time of the activity are of importance, the intermediate progress is either not of concern or can be satisfactorily approximated with a pre-defined progress curve.
- Except at the start and the finish point, the activity interacts with other activities only sparsely during the progress.
- The activity is not very sensitive to continuously changing factors such as temperature, wind, precipitation, etc.
- The activity is performed at a fixed position or its movement does not need to be tracked.
- Resources engaged in the activity do not need to be closely monitored.

Otherwise, it is more proper to model the activity as a continuous activity.

A discrete activity can be defined through the window-based user interface, as shown in Figure 6-2. The items need to be entered include:

- ***Name***. The name of each activity must be unique.
- ***Work Quantity*** and ***Unit***. The total quantity of work involved in the activity and the unit of measurement. The progress of the activity will be measured with the same unit herein provided. The default value is "100 %", in which case, the progress of the activity will be measured in percentage completed.
- ***Duration*** (for non-resource driven activities). The duration can be defined as a constant number, a probabilistic distribution, or an expression containing any accessible functions and variables. The value of the duration will be determined at the point when the activity starts, and will not change unless being adjusted by external inputs.
- ***Resource Type, Minimum/Maximum Quantity, Priority*** and ***Productivity*** (for resource-driven activities). If the activity is resource-driven, the user needs to specify the type, minimum quantity, maximum quantity and priority of the resources that can be used to finish the work. The productivities of the resources on the activity could be obtained from a pre-established database, and may be a constant number, a probabilistic distribution, or an expression containing any accessible functions and variables. The duration of the activity

is dependent upon which type of the resources and what quantity can be actually acquired, and it will be determined dynamically at the point when the activity starts with the following formula:

$$Duration = \frac{Work\ Quantity}{\Pr oductivity\ of\ the\ Aquired\ \operatorname{Re} source \times Quantity\ of\ the\ Aquired\ \operatorname{Re} source}$$

(Equation 6-1)

- ***Progress Curve***. The progress curve describes how the activity would proceed from start to finish with the passage of time without any external inputs. Three types of progress curves are predefined to be selected from: Type I assumes a constant speed; Type II considers the learning curve effect; Types III reflects a slow-fast-slow pattern (see Figure 6-2). The progress curve can also be customized by defining the progress function *Progress = p (duration, time)*.

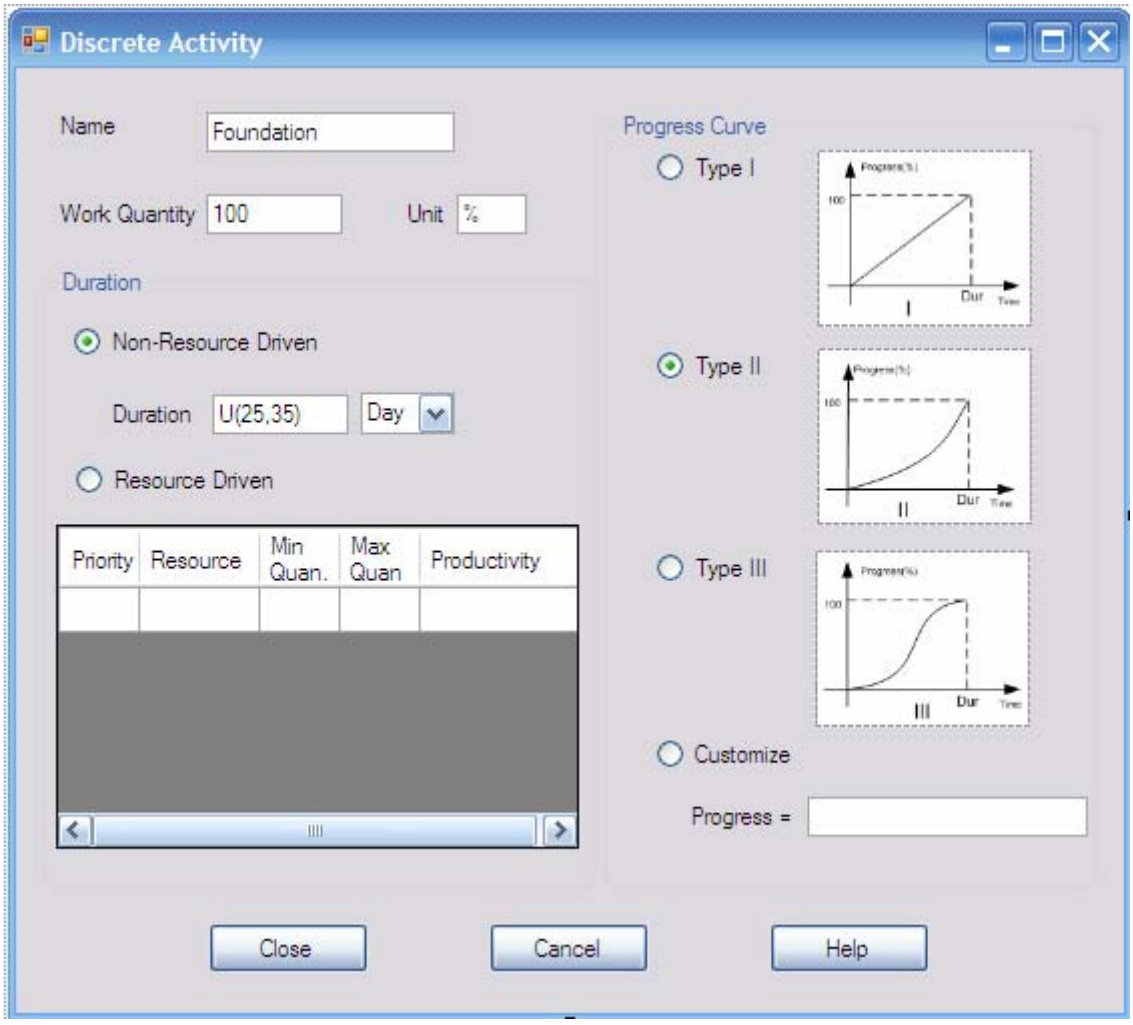


Figure 6-2. Discrete activity dialogue window

A discrete activity node will be compiled into an atomic discrete activity model, either the resource-driven A_d or the non-resource-driven A_{dr}, in accordance with the selected option.

6.1.2 Continuous Activities

Continuous activities are represented as shown in Figure 6-2 as rectangles with a double left border line, with the name and the productivity of the activity indicated in the middle. A continuous activity needs to be defined with:

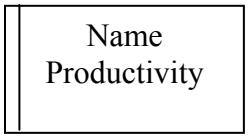


Figure 6-3. Continuous Activity Node

- ***Name***.
- ***Work Quantity***.
- ***Interval***. The state of the continuous activity will be updated at fixed time intervals during the progressing phase. The interval must be defined as a constant number. Depending on the required level of accuracy, different continuous activities in one project can be defined with different lengths of intervals.
- ***Resource Type, Min/Max Quantity,*** and ***Priority.***
- ***Productivity***. The productivity of the continuous activity can be defined as a constant number, a probabilistic distribution, or an expression. Different from the discrete activity, the value is re-sampled (if probabilistic) and recalculated every time the activity advances.
- ***Productivity Modification Factor.*** This factor is used to take account of the effects of dynamically changing variables on the continuous activity. Every time the activity advances, this factor is going to be applied to the productivity to calculate the ***Modified Productivity***, which determines the increase of the progress for the next interval.

Figure 6-4 shows the dialog window used to define the continuous activity Form. The productivity is defined as a uniform distribution U (240, 260) sf/hour. The modification function indicates that the productivity of the activity Form is affected by two continuously changing variables: temperature and weather. Suppose that at time *t*, the temperature is 90°F, then the value of the step function which is dependent upon temperature is .85; the weather is "Cloudy", then the value of the conditional function which is dependent upon weather is 1.0. So the modified productivity of Activity Form at time *t* is: (.85 × 1.0) × U (240, 260) = .85 × 248 = 210.8 sf/day.

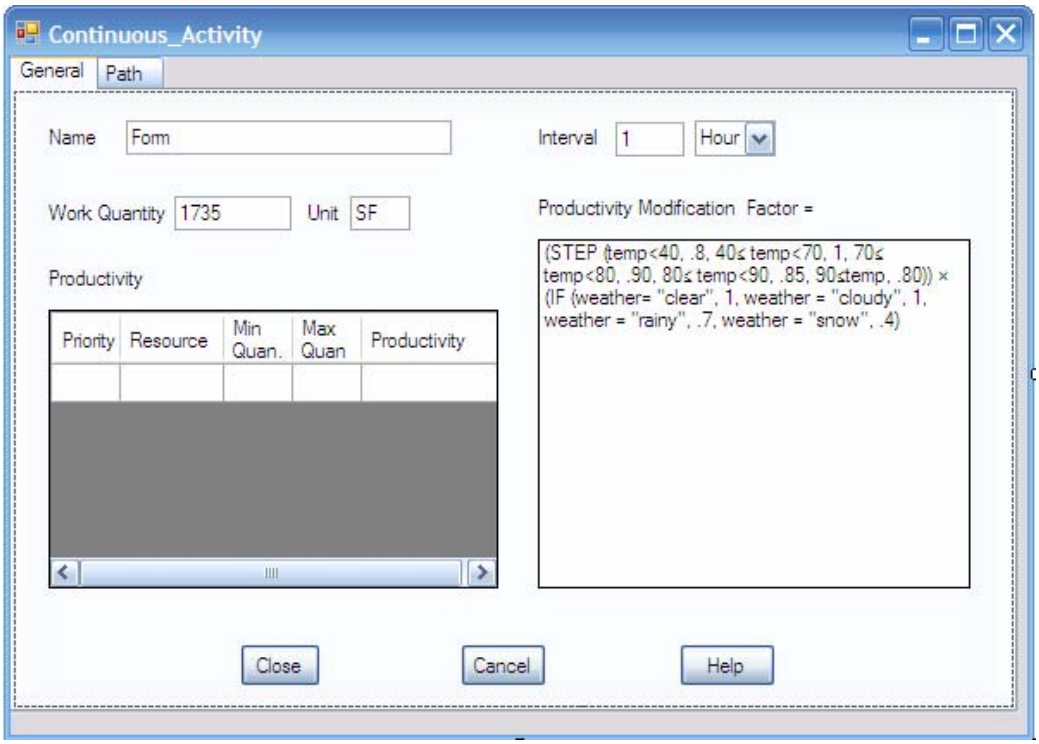


Figure 6-4. Continuous activity dialogue window

Often times it is necessary to track the movement of the crew that is working on the continuous activity. If all continuous activities in the project have the same layout, only two parameters need to be provided:

- ***Start Position.*** The distance between the point where the activity begins and the designated "0" point. The default value is 0, i.e., it is assumed that the activity start from the designated "0" point.

- ***Direction.*** In the one-layout situation, there are only two directions any activity can take: "position" or "negative", with "positive" as the default value.

With these two parameters, the positions of the crews can be derived from the progress of the activity at any time. From example, in Figure 6-5, the start position of both Activity A and Activity B is +150m, A is proceeding toward the "positive" direction, and B toward the "negative" direction. If it is known that at time *t*, A has progressed 400m, and B has progressed 600m, it can be determined that A is at +550m, B is at -450m, and the distance between the two is 1000m.

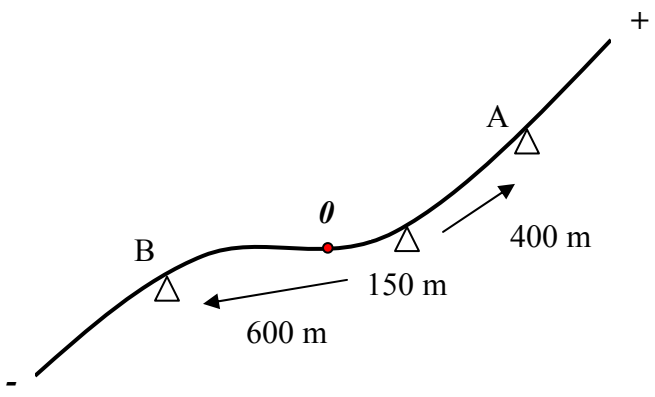


Figure 6-5. Determining activity positions on a one-dimensional layout

If the layouts of the activities are different, a two-dimensional coordinate system needs to be established, and the following parameters need to be provided for each continuous activity:

- ***Layout Function.*** The user needs to specify the shape of the layout with a function *y=f(x)*.

- ***Start Position.*** The (x, y) coordinates of the start point.

- ***End Position.*** The (x, y) coordinates of the finish point.

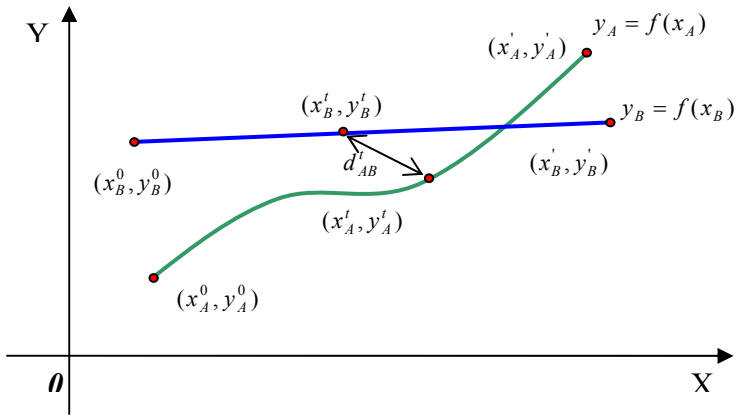


Figure 6-6. Determining activity positions on a two-dimensional layout

In the example shown in Figure 6-6, the layout functions, start positions and end positions of Activity A and B can be specified. If their progress at time *t* is known, their positions at time *t*, (x^t_A, y^t_A) and (x^t_B, y^t_B), can be determined, and the distance between the two activities at time t, d^t_{AB}, can be calculated with:

$$d^t_{AB} = \sqrt{(x^t_A - x^t_B)^2 + (y^t_A - y^t_B)^2} \ . \qquad \text{(Equation 6-2)}$$

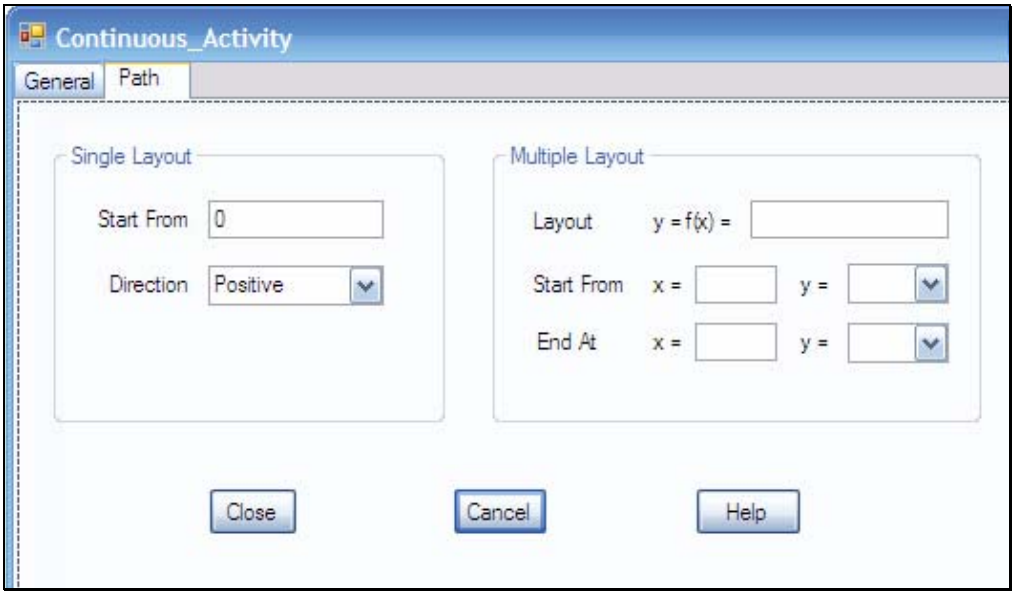


Figure 6-7. Defining the path of a continuous activity

Figure 6-7 shows the dialogue window used to define the path of continuous activities.

6.1.3 Environmental Factors

Environmental factors are used to represent dynamically changing variables such as temperature, precipitation and wind speed. The environmental factor is represented as a hexagon, with the name of the factor and the names of the variables indicated inside.

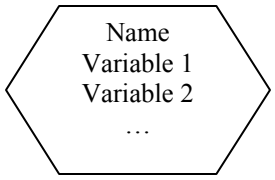


Figure 6-8. Environmental Facor Node

Attributes of the environmental object include:

- ***Name***. A unique identifier of the environmental factor.
- ***Variable Name*** and ***Value.*** The name of the variables and the expressions that determine their values.
- ***Interval***. The interval at which the values of the variables are going to be updated. The interval can be a constant number, a probabilistic variable or an expression.

Variables that require different updating intervals need to be generated by different environmental factors. Therefore one project may include more than one environmental object.

Figure 6-9 displays the dialogue window that defines the environmental object Clock. Two variables, *temperature* and w*eather* are updated every two hours. The value of *temperature* is assumed to have a Poisson distribution Poisson (65, 12). The value of *weather* is assumed to follow the distribution ((“clear”, 0.5), (“Cloudy”, 0.3), (“Rainy”, 0.1), (“Snow”, 0.1)).

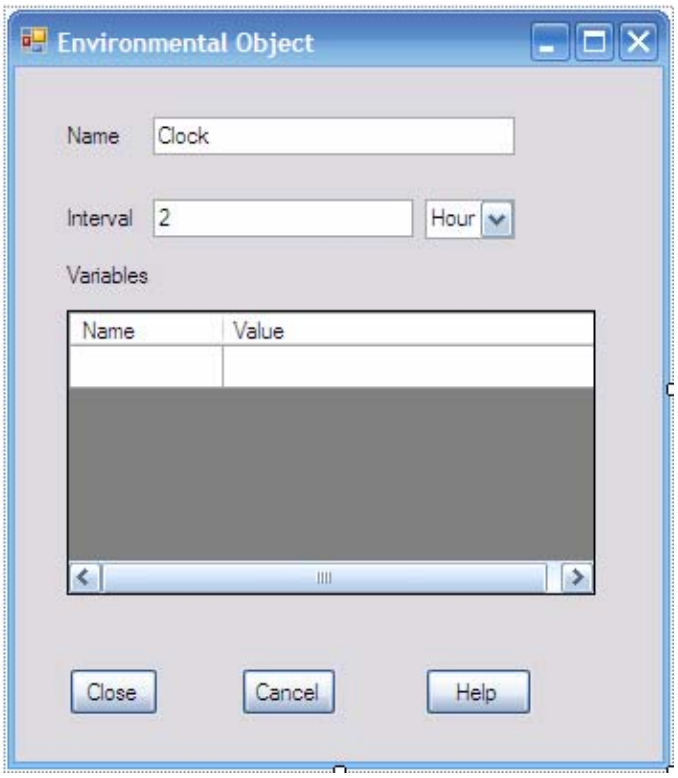


Figure 6-9. Environmental factor dialogue window

6.1.4 Resources

Resources are represented as ovals in the network. The type and the total quantity are indicated in the middle of the oval. A resource has to be defined with the following attributes:

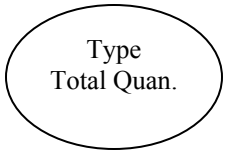


Figure 6-10. Resource Node

- ***Type***. The type should be a standardized name or code for the resource.
- ***Total Quantity***. This should be the total quantity of the resource available to the project.

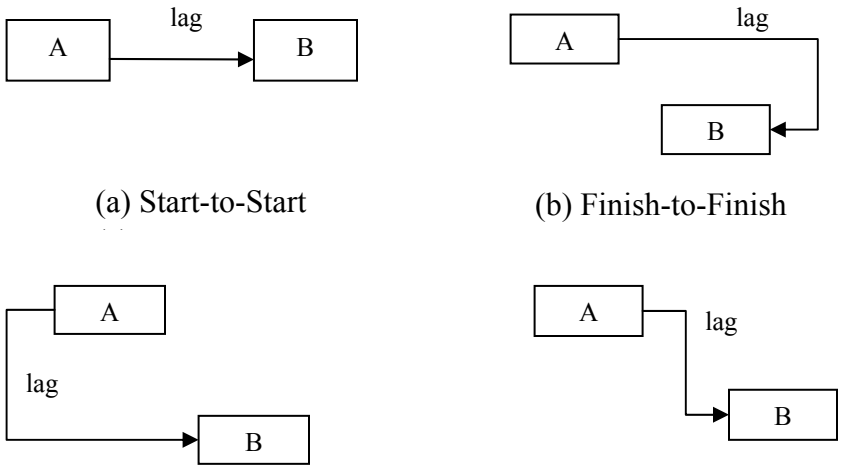
6.2 Links

Links are used to show how the behaviors of an activity are controlled and affected by other activities, resources and environmental factors. The links are divided into start/finish links, interrupt links, adjust links, buffers, and resource links, based on which behavior is being controlled and affected.

6.2.1 Start Links and Finish Links

Start/finish links are represented as arrows. The head of the arrow is connected to the activity being controlled and constrained, which could be either continuous or discrete, resource-driven or not resource-driven; the tail of the arrow is connected to the influencer which could be an activity of any type or an environmental factor.

6.2.1.1 FS, SS, SF, FF dependency relationships



(a) Start-to-Start (b) Finish-to-Finish

(c) Start-to-Start (d) Start-to-Start

Note: the predecessor and successor activities can be either discrete or continuous activities.

Figure 6-11. FS, FF, SS, SF Links

Using the arrows, the Finish-to-Start, Start-to-Start, Start-to-Finish, Finish-to-Finish dependency relationships method are represented just the same as in the AON

network of the CPM method (shown in Figure 6-11). However, they are interoperated differently by the proposed method.

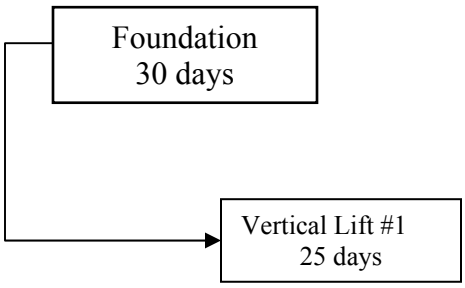


Figure 6-12. Example of an SS link

In Figure 6-12, there is an SS dependency relationship between the activity Foundation and the activity Vertical Lift #1 with a lag of two days. This link will add a marker "0%" in the marker_list of the activity model $Foundation_d$, and insert a logical variable con(Foundation, 0%) into the *start_constraint* of the activity model Vertical_Lift_#1d. When $Foundation_d$ reaches the state (phase = "progressing", progress = 0%), it will generate a message (Foundation, 0%) and post it to its "0%" output port. After a delay of 10 days, this message is routed to the "start" input port of activity Vertical Lift #1, which causes the value of the logical variable con(Foundation, 0%) turn from "False" to "True".

6.2.1.2 Progress-based dependency relationships

The arrows can also represent progress-based dependency relationships as illustrated in Figure 6-13. As discussed in Section 3.1.1, the duration-based SS relationship in the above example is not an accurate description of the real relationship between the two activities. With the proposed method, we can represent "when Activity Foundation is 1/3 completed, Activity 1st Lift Vertical can start" accurately as shown in

Figure 6-14. Note that this start link is connected to the bottom of the predecessor at the position of about 33% length.

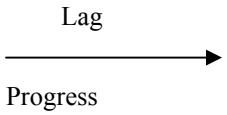


Figure 6-13. Progress-based Link

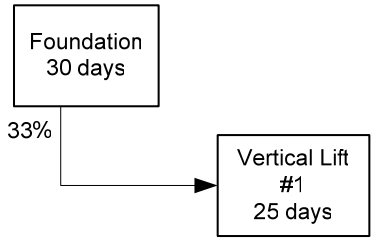


Figure 6-14. Example of a progress-percentage-based link

If the work quantity of the activity Foundation is defined as 3000 CY, then the progress of the activity will be measured in CY instead of the percentage completed, and the above link should be specified as shown in Figure 6-15.

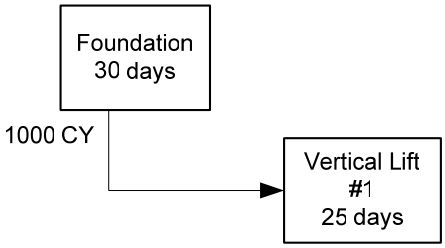


Figure 6-15. Example of a work-quantity-based link

6.2.1.3 Environmental factor imposed start/finish constraints

When the start/finish constraint is imposed by the environmental factor, constraint condition should be indicated on the arrow as shown in Figure 6-16. The example in Figure 6-17 shows that the activity Foundation cannot start if the temperature is below 40°F.

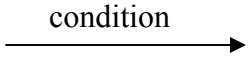


Figure 6-16. Environmental-factor-imposed constraint

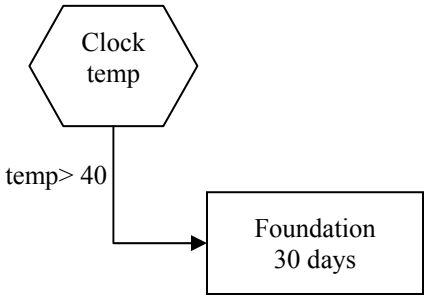


Figure 6-17. Example of an environmental-factor-imposed constraint

The condition "temp>40" will be inserted into the *start_constraint* of the activity model $Foundation_d$, and the *(temp, 40, +1)* will be added into the *threshold_list* of the environmental model $Clock_e$. When the value of the variable *temp* rises above 40 degrees, a "Test" message will be generated by $Clock_e$ and sent to the "start" input port of $Foundation_d$.

6.2.2 Interrupt Links and Resume Links

The interrupt links shown in Figure 6-18 are used to model interruptions caused by ***sudden*** events during the middle of the progress of an activity. The interrupt link is represented as an arrow with a solid diamond head. It is often used in pairs with a resume link, which is represented as an arrow with a hollow diamond head. Similar to the start/finish links, they can be used to control both the discrete and continuous activities, and the influencer can be either an activity or an environmental factor.

Figure 6-18. Interrupt and Resume Links

6.2.2.1 Interruptions caused by other activities

An example of interruption caused by another activity is: when Activity Blasting, starts, Activity B has to stop; after Activity A is finished, Activity B can resume. Figure 6-19 shows the network representation of this example.

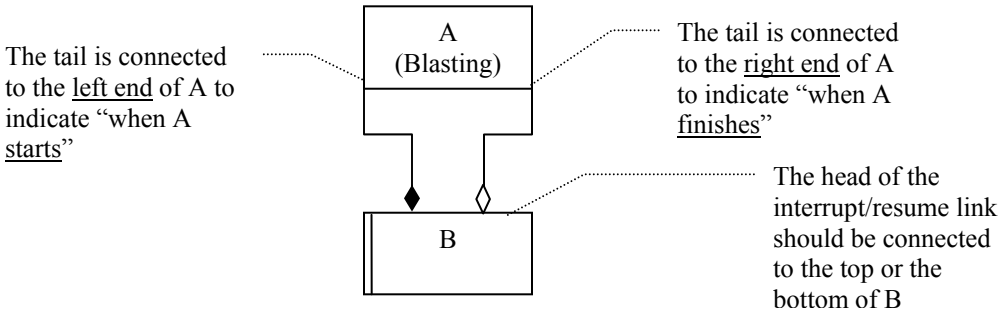


Figure 6-19. Network representation of interruptions and resumptions

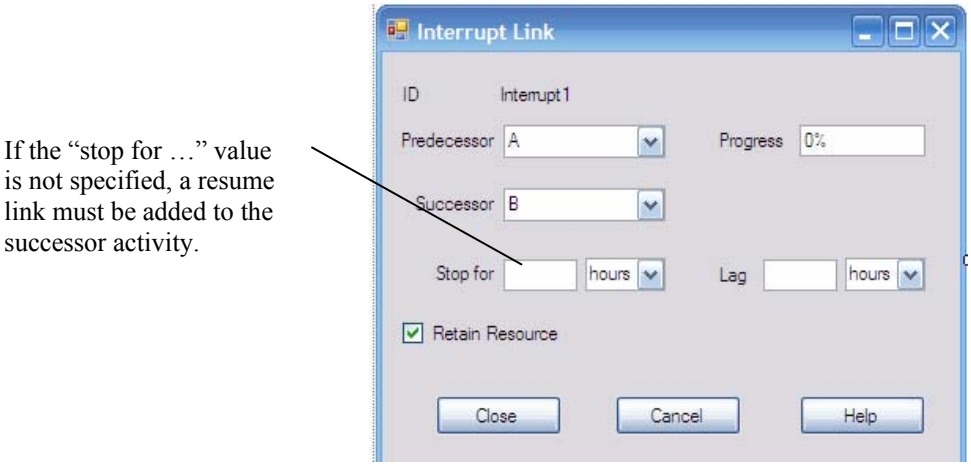


Figure 6-20. Interrupt link dialogue window

Details about the interrupt link need to be further specified in the dialogue window, as shown in Figure 6-20. By default, during the interruption, the resources engaged in the

activity will be retained, so that the activity does not need to re-acquire the resources when it resumes. But the retained resources will be idled during the interruption. To fully utilize the resources, the user should uncheck the "Retain Resource" option in the Interrupt Link dialogue window.

The interrupt and resume links could also be progress-based, i.e., the receiver activity can stop/resume after the sender activity has achieved a certain progress. The value of the progress should be indicated near the connecting point of the sender activity and the link.

Lags can also be represented. If a lag is indicated above the link, the receiver activity will not stop/resume immediately when it receives the message from the sender, but hold for the specified length of time.

6.2.2.2 Interruptions caused by environmental factors

Figure 6-21 shows an example where the interruption is caused by the environmental factor. The activity Foundation will stop whenever a storm comes. It usually requires a period of Poisson (2, 0.5) hours for the storm to pass and for the crew to clear-up the site before the work can resume.

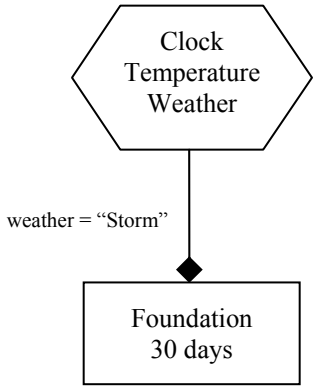


Figure 6-21. Example of an environmental-factor-caused interruption

In this example, the interrupt link is not used in pairs with the resume link. The activity will resume by itself after it has been paused for Poisson (2, 0.5) hours, as specified in the Interrupt Link dialog window in Figure 6-22.

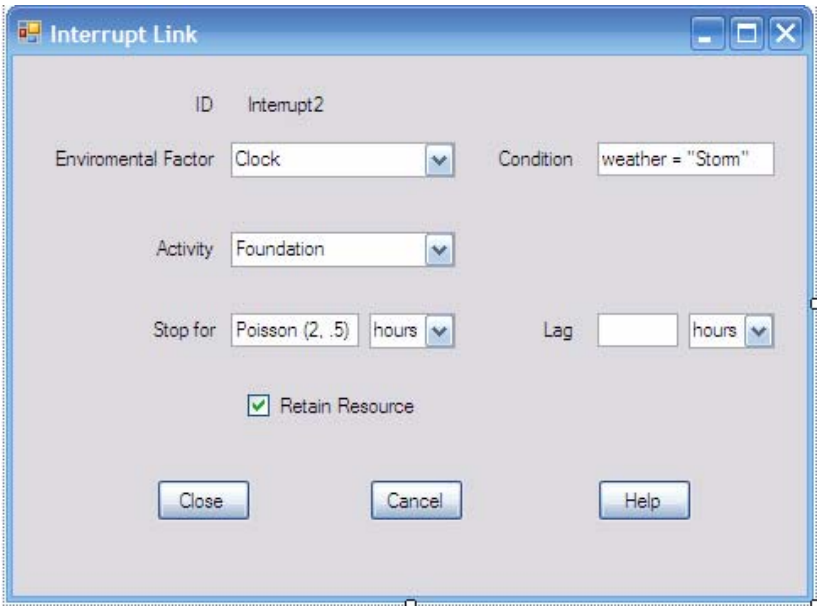


Figure 6-22. Defining breaks in the interrupt link dialogue window

An activity might be under the control of multiple interrupt/resume links, as shown in Figure 6-23. To avoid errors in the simulation process, the interrupt link and the resume link that work in pairs will be assigned with the same number identification in their ID — if the interrupt link that represents "when the temperature is lower than 40°F, the activity Foundation must stop" has the ID "Interrupt*(n)*", the resume link that represent "Activity Foundation will resume when the temperature rises to above 40°F will be identified as "Resume*(n)*".

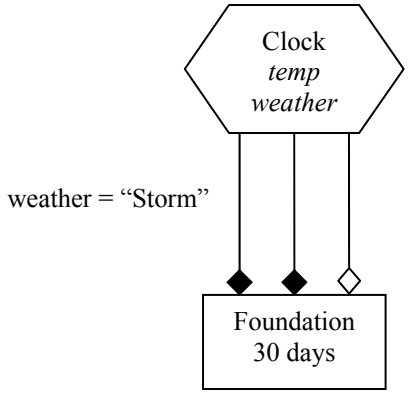


Figure 6-23. Multiple interrupt/resume links on one activity

6.2.3 Adjust Links

Adjust links are used to model duration/productivity changes caused by ***sudden*** events during the middle of the activity's progress. The head of the adjustment link is a circle, with the value of the adjustment factor indicated inside, as shown in Figure 6-24.

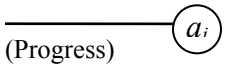


Figure 6-24. Adjust Link

Every activity, either discrete or continuous, has three predefined adjustment factors a_1, a_2 and a_3, whose original values are 1. They are so named to be differentiated from the productivity modification factors of the continuous activities. The productivity modification factors will be updated and applied every time when the continuous activity advances, whereas the adjustment factors will only be applied when the activity receives inputs on its "adjust" input port.

6.2.3.1 Adjustments caused by other activities

Consider the example used in Figure 6-14, which specifies that Activity Lift #1 cannot start until the activity Foundation is 33% finished. We can add an interrupt link to

further describe that when the activity Vertical Lift #1 starts, Activity Foundation will be interfered and its duration will be extended to 1.4 times as long as estimated normal length, as shown in Figure 6-25.

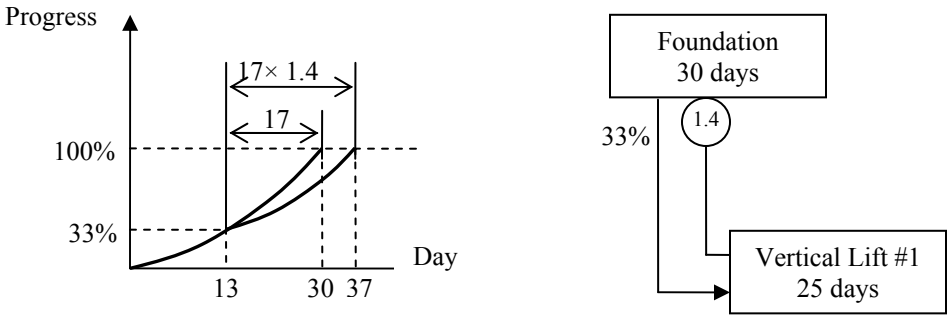


Figure 6-25. Example of an adjustment

The specified value "1.4" will be assigned, by default, to the first adjustment factor a_1. According to the original progress curve of the activity Foundation, it takes 13 days (actual working time) to finish the first 33% of the work, and the remaining 67% will take 17 (= 30 -13) days. The activity Vertical Lift #1 only impacts the remaining 67% work and extends the remaining duration to:

$$\textit{Adjusted Remaining Duration} = \textit{Remaining Duration} \times a_1 \times a_2 \times a_3$$

$$= 17 \times 1.4 \times 1 \times 1$$

$$= 24 \text{ days}.$$

Therefore, Activity Foundation takes 37 (=13+24) days in total to complete. If Activity Vertical Lift #1 starts at a later point (which is very possible as its start is controlled by other activities), the time to complete Activity Foundation will not be extended as much.

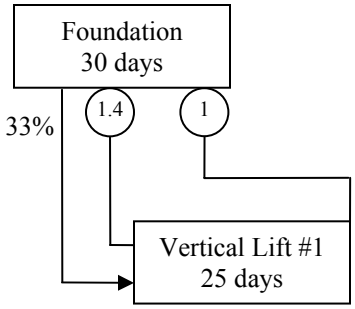


Figure 6-26. Adding additional adjust links

Furthermore, to represent the fact that when Activity Vertical Lift #1 finishes its impacts on Activity Foundation will also end, another adjust link needs to be added as shown in Figure 6-26. Note that it must be ensured that in the dialog window it is the value of the adjustment factor a_1 that will be changed, as shown in Figure 6-27. If the value 1 is given to a_2 or three, the product of $a_1 \times a_2 \times a_3$ will remain as $1.4 \times 1 \times 1 = 1.4$.

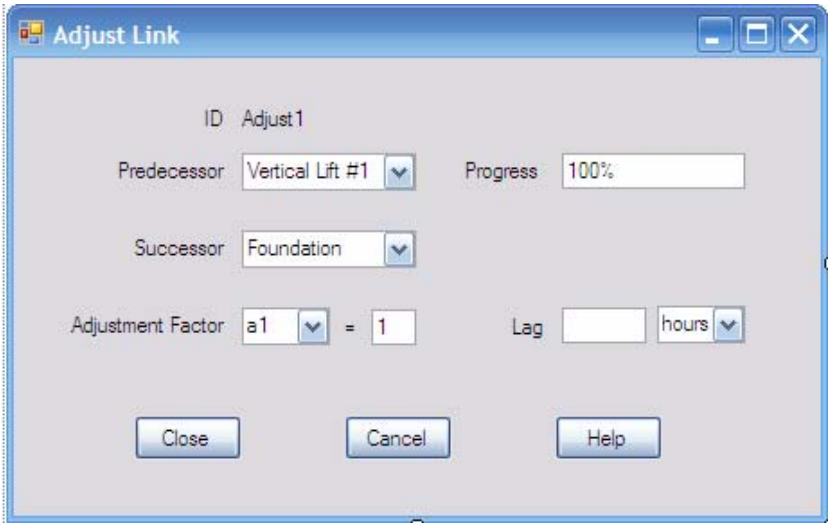


Figure 6-27. Adjust link dialogue window

In the above example, the activity is a discrete activity, and the remaining duration of the activity has been changed by multiplying with the specified adjustment factor. If

the activity is a continuous activity, what will be adjusted is the modified productivity of the continuous activity, and the formula is as follows:

$$Adjusted\ Productivity = Modified\ Productivity \times a_1 \times a_2 \times a_3. \qquad \text{(Equation 6-3)}$$

6.2.3.2 Adjustments caused by environmental factors

Adjustment can also be caused by environmental factors. For example, the activity Foundation involves a lot of concrete placement. When the temperature drops to below 40°F, special construction methods must be taken to avoid concrete cracking. In Figure 6-28, it has been specified that when construction enters the winter season and temperature is below 40°F, the duration of the activity would be extended to 1.7 times. In Figure 6-29, the dialogue window for Link Adjust02, the adjustment factor to be changed is set to a_2. So under the combined impacts of temperature and Activity Vertical Lift #1, the remaining duration of Activity Foundation will be extended by *(1.4 ×1.7)* times. If the adjustment factor is left as the default a_1, the duration will be extended by either 1.4 or 1.7 times, depending on which link has been activated the latest.

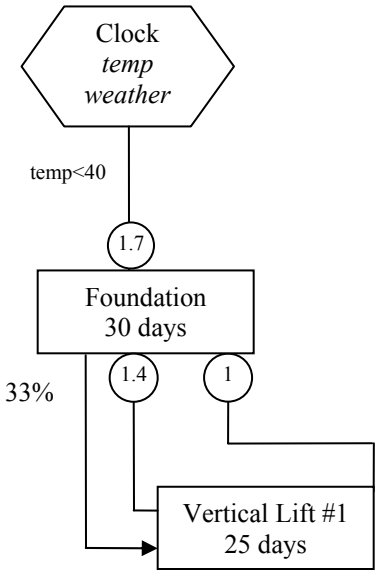


Figure 6-28. Example of an environmental-factor-caused adjustment

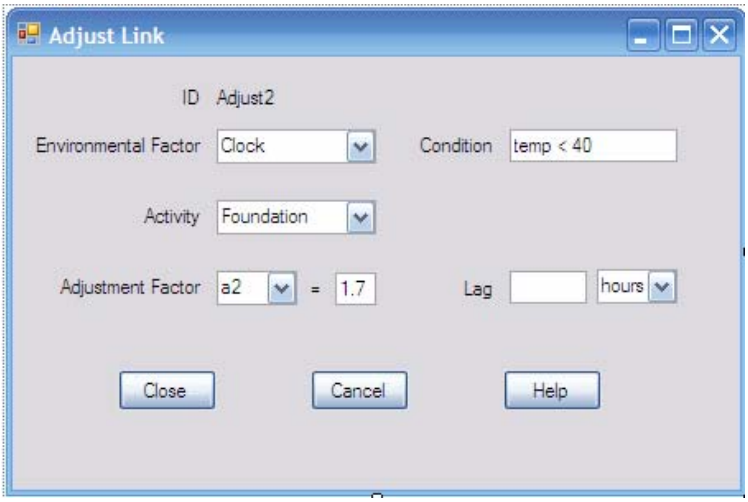


Figure 6-29. Adjust link dialogue window for environmental-factor-caused adjustments

A final caution with the use of the adjust links is: if the activity is very sensitive to environmental variables that are constantly changing, it would be much more efficient to model the activity as a continuous activity and the impacts of the environmental variables should be specified using the productivity modification factors (refer to the example illustrated in Figure 6-4).

6.2.4 Buffers

Buffers can only be defined between two continuous activities. The buffer is represented as a wavy arrow, with the size of the buffer indicated on the top, as shown in Figure 6-30.

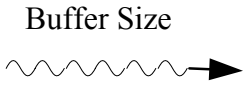


Figure 6-30. Buffer Link

The buffer link will be compiled into a logical condition:

Progress of the predecessor – Progress of the successor ≥ Buffer Size. (Equation 6-4)

This logical condition will be inserted into the *buffer_constraint* of the successor activity. The *buffer_constraint* will be checked when the successor activity attempts to start and every time when it attempts to advance (refer to Section 5.2.1.4). If the *buffer_constraint* is not violated, the successor activity will make one step with the *adjusted productivity*; otherwise, it will slow down to follow the progress of the predecessor activity or stop for a specified period of time, depending on the user's option.

The proposed method also allows the buffers to be measured on the distance between the two activities, provided that the movements of both activities have been defined (refer to Section 6.1.2).

The buffer links can be used flexibly to represent many realistic situations, which will be introduced below.

6.2.4.1 Minimum buffers

The most common type of buffers is minimum buffers — two activities have to maintain a minimum distance from each other. Figure 6-31 shows an example in which the activity RMEP has to maintain a minimum buffer of 2000 SF from the activity Form. When being blocked, RMEP will slow down to follow Form. The buffer dialogue window is displayed in Figure 6-32. During the progressing phase, at every time interval, if (the progress of Activity Form – the progress of Activity REMP) ≤ 2000sf, Activity REMP will slowdown to follow the pace of Activity Form.

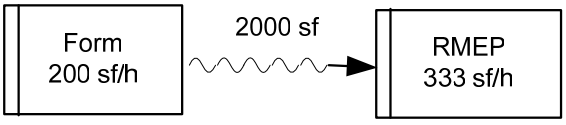


Figure 6-31. Example of a minimum buffer

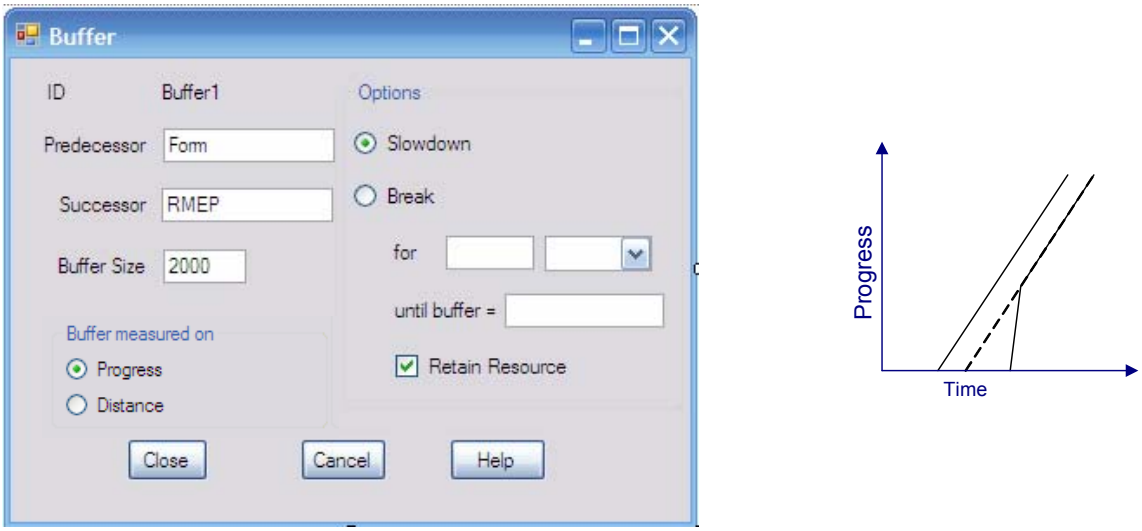


Figure 6-32. Defining a minimum buffer with the "Slowdown" option

We can also change the option in the dialog window as shown in Figure 6-33 to specify that Activity RMEP will have a break of 2 hours when the buffer is violated. Another possible situation is when Activity RMEP runs into the 2000sf buffer, it will stop until Activity Form is 3000sf ahead of it. This situation can be represented as in Figure 6-34. Whether the resource working on the follower activity would be retained during the period of interruption can be controlled through the "Retain Resource" checkbox.

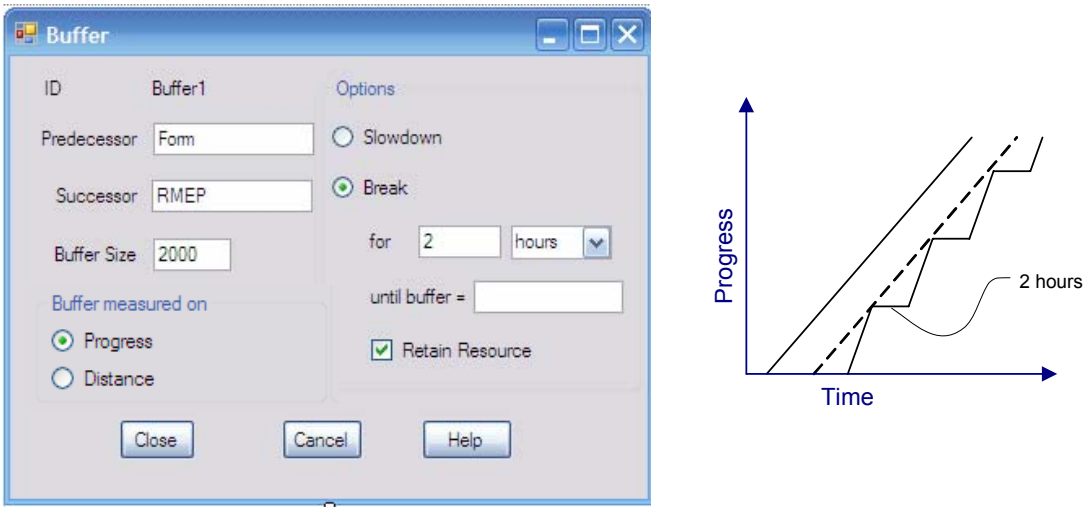


Figure 6-33. Defining a minimum buffer with the "Break for …" option

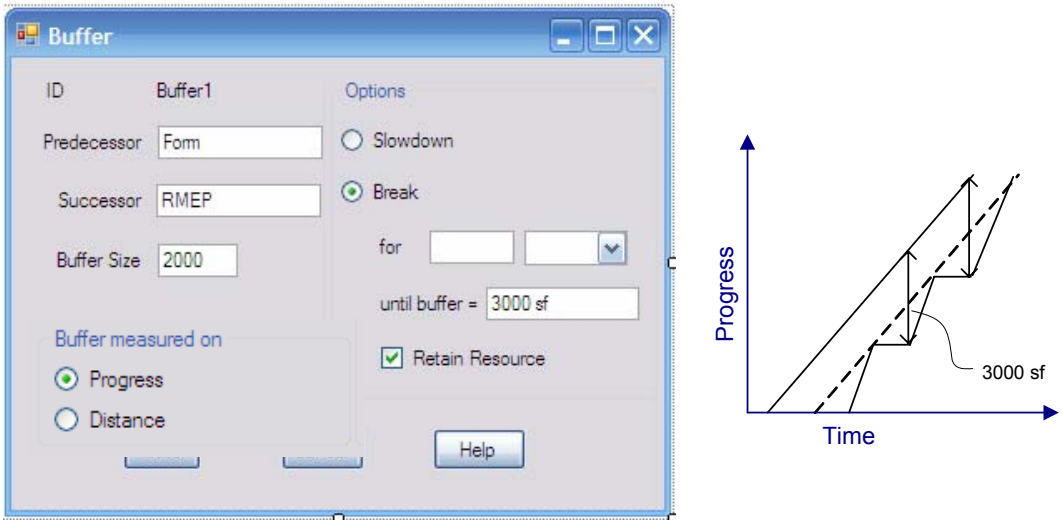


Figure 6-34. Defining a minimum buffer with the "Break until…" option

6.2.4.2 Maximum buffers

Instead of keeping away from each other, sometimes activities need to follow each other closely. When the distance of the two activities exceeds a specified maximum buffer size, the activity that takes the lead has to stop to wait for the activity that follows behind. To represent this type of situation, the head of the buffer link should be connected to the leading activity and the tail of the buffer link should be connected to the follower activity (in this case, the leading activity is the successor in the buffer link, and the follower activity is the predecessor). The size of the buffer must be entered as a negative value.

In the example shown in Figure 6-35, Activity A has a faster pace than Activity B, so the distance between the two activities will increase with time. Every time when Activity A attempts to advance, it will check the value of (the progress of Activity B – the progress of Activity A). If Activity B lags behind by 2100sf, then the value of the difference would be (-2100), which is smaller than the specified buffer size (-2000 sf).

Activity A therefore will slowdown or pause to wait for Activity B to catch up. Figure 6-36 illustrates the situation where A would pause for 2.5 hours every time when it runs "out of" the 2000sf buffer.

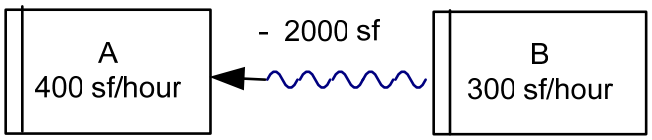


Figure 6-35. Example of a maximum buffer

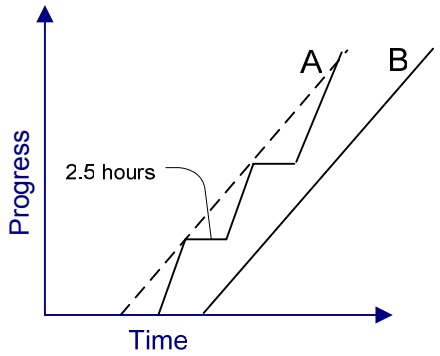
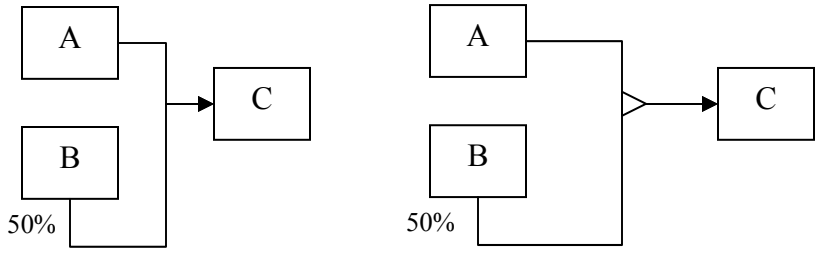


Figure 6-36. Illustration of the maximum buffer

6.2.5 Combining the links

A lot of times, the behaviors of the activity are controlled by joint conditions. For example, when Activity A is finished "AND" Activity B is 50% complete, Activity C can start; or when Activity A is finished "OR" Activity B is 50% complete, Activity C can start. In the proposed method, "AND" logic are represented by simply merging the head of the links, as shown in Figure 6-37(a); "OR" logic are represented by joining the links with a ▷ node, as shown in Figure 6-37(b).



(a) AND logic (b) OR logic

Note: the representation of the "AND" and "OR" logic applies to all type of links, including the start link, finish link, interrupt link, resume link, adjust link and buffer.

Figure 6-37. Representation of "AND" and "OR" logic

Multi-layered constraints can also be quickly constructed with the proposed method. The constraint "(A finished OR B 50% complete) AND temp>40" is represented in Figure 6-38.

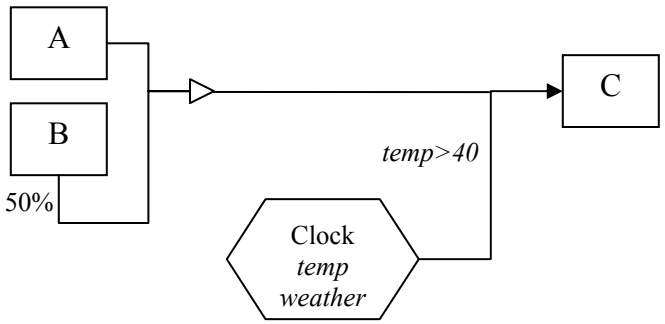


Figure 6-38. Example of a multi-layered constraint

6.2.6 Branching of the links

Branching of the link is represented with a diamond node as shown in Figure 6-39. If the selection is random, the probabilities of taking the branches need to be indicated, and they must sum up to 1. In the example shown in Figure 6-40 (a), the likelihood of executing Activity B is .6 and the likelihood of executing Activity B is .4.

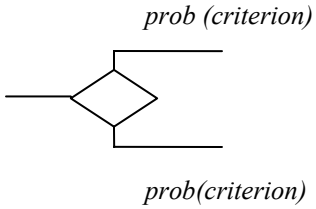
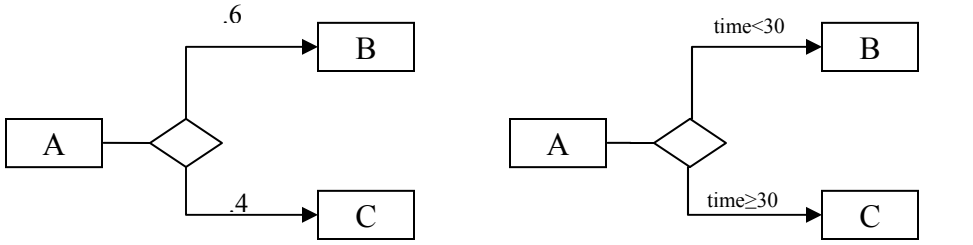


Figure 6-39. Branching Node



(a) Random Selection (b) Decision-based Selection

Figure 6-40. Example of branching

If the selection is decision-based, the criteria of choosing the branches need to be indicated and they must be mutually exclusive. In Figure 6-40(b), when Activity A finishes, if the time is still early enough (time<30), Activity B will be executed; otherwise Activity C, which might be an expensive remedy plan, will be executed.

The branch node will be compiled to a branch model. It can route the received input from the predecessor activity, to one of its output ports according to the value of a random variable or the decision condition, from where the message can be passed on to the selected successor activity.

6.2.7 Resource Links

The resource link is used to represent the activity's requirement for a certain type of resource. As shown in Figure 6-41, it is drawn as a straight line, with the required

quantity indicated on the top. The quantity could be a single positive number or a range expressed as (min, max) showing the upper and lower limit of the requirement.

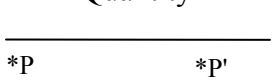


Figure 6-41. Resource Link

The notation *p at the activity's end indicates the priority of the connected resource; if the resource is the only type that the activity is able to use or the first choice among all the alternatives, the notation *1 can be omitted. At the other end, near the resource, the notation *p' indicates the priority of the activity to the resource. If the resource is not shared by any other activities, or if the activity has the highest priority for the use of that resource, the notation can be omitted.

Figure 6-42 shows an example on the use of the resource link. In this example, Activity A could be finished either with two units of Resource S, or with one unit of Resource P. As Resource S has the higher priority (*1 has been omitted) between the two, Activity A's requests for resources will be sent to Resource S first.

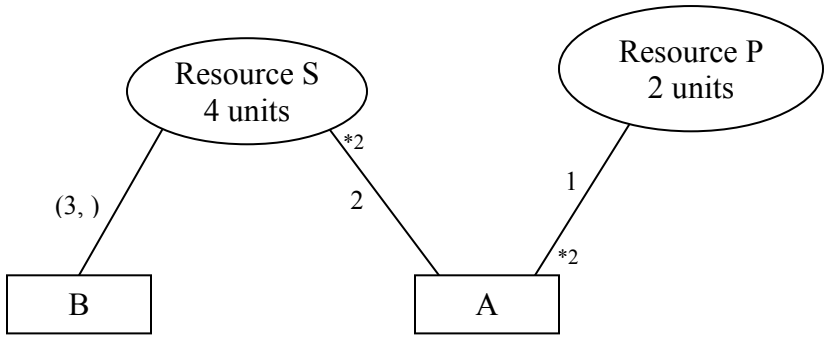


Figure 6-42. Example of resource links

Suppose that by the time Activity A's request arrives, Resource S has also received another request from Activity B. As Activity B has a higher priority over Activity A for

Resource S, Activity B's request will be processed first. All remaining units of Resource S will be assigned to Activity B, whose lower limit is 3, upper limited is omitted, which means the more the better. Activity A's request cannot be fulfilled, and thus will be stored in the request queue of Resource S. Resource S will generate and send a waiting notice to Activity A.

Upon receiving the waiting notice, Activity A will submit another request to Resource P. If this request could be fulfilled by Resource P, Activity A will send a message to remove its previous request stored in the request queue of Resource R. If the request fails again, Activity A will be hold in the "Requesting_Resource" phase until either of the requests is satisfied.

6.3 Compound Activities

One of the most important features of the proposed method is multi-level modeling. With the bottom-up approach, multiple activities (could be of different types) can be coupled together to form a compound activity; the compound activity can be further coupled with other activities —either atomic or compound — in the construction of higher-level compound activities.

With the top-down approach, an atomic activity can be decomposed into a compound activity containing component activities, each of which could be further broken down into lower-level component activities.

A compound activity could be defined as discrete or continuous, and the discrete compound activity could be either resource-driven or non-resource-driven.

Seeing as a black box, the compound activity is used just as its atomic counterparts: any links that can be applied to an atomic continuous activity can be applied to a compound continuous activity; any rules that have to be followed when linking an atomic

continuous activity have to be followed when linking a compound continuous activity. So does the discrete ones.

6.3.1 Compound Discrete Activities

A ***Compound Discrete Activity*** may contain both discrete and continuous components. It is represented as a box with a "+" sign in the upper right corner to differentiate it from the atomic discrete activity. The name of the compound activity is indicated in the upper left corner.

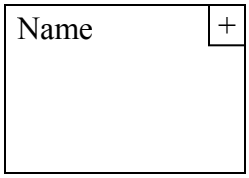


Figure 6-43. Compound Discrete Activity Node

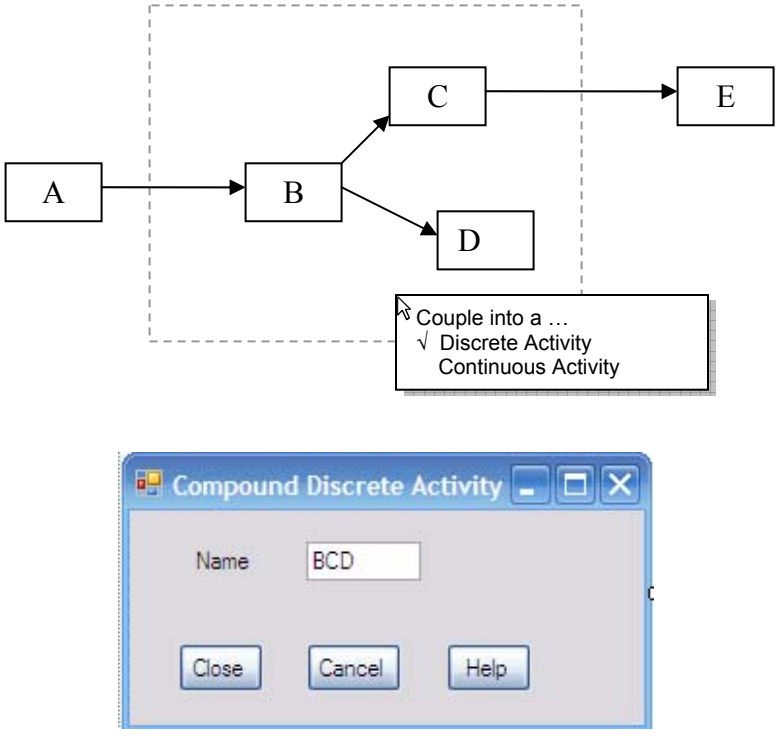


Figure 6-44. Defining a compound discrete activity by coupling

In Figure 6-44, Activity B, C and D are selected and coupled into a compound discrete activity named BCD. We can add start, finish, interrupt, resume, and resource links to the compound activity BCD just as to any discrete atomic discrete activities. An example is shown in Figure 6-45. (For the rules governing the routings of the links between the compound discrete activity and its components, refer to Section 5.2.2.1 and 5.2.2.2.)

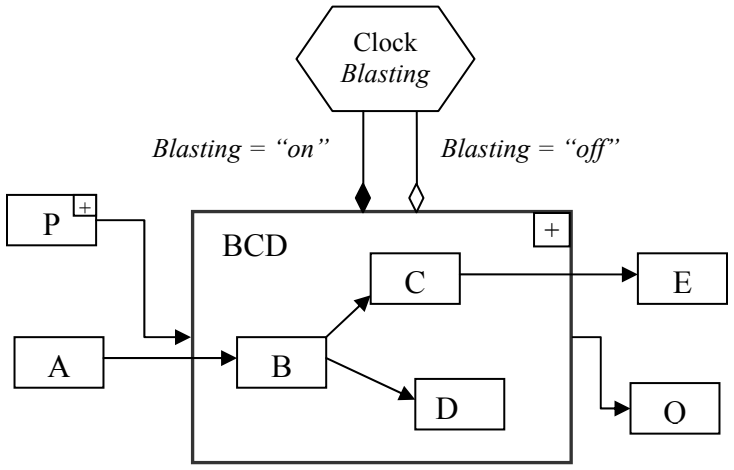


Figure 6-45. Adding links to a compound discrete activity

The only difference when linking a compound activity is: if the link is based upon the partial completion of the compound activity, the user needs to define with the progresses of its components when such a state is considered achieved. For example, when adding a link to represent that Activity S cannot start until Activity BCD is 50% completed, a dialogue window will pop up as shown in Figure 6-46. The user may specify that, according to his or her best judgment, Activity BCD can be considered 50% completed when its components Activity C and Activity D both have completed 50%.

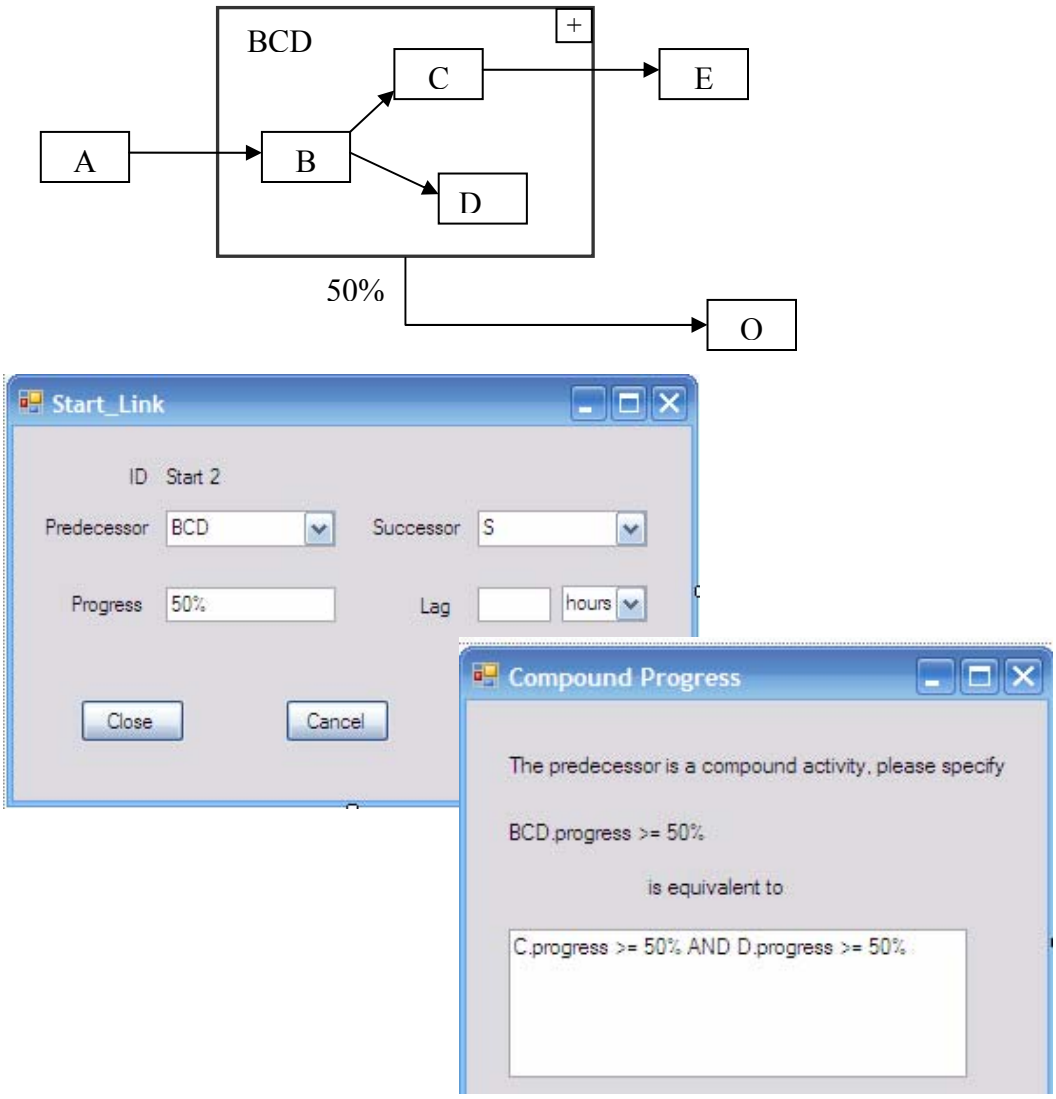


Figure 6-46. Defining a progress-based link on a compound discrete activity

We can also break-down an atomic discrete activity to a compound discrete activity. In Figure 6-47, the atomic discrete Activity D has been decomposed into three components including D1, D2 and D3, each of which needs to be defined separately as an atomic discrete activity or an atomic continuous activity. The Activity D becomes a compound activity and loses its originally defined attributes including the duration, progress curve and resource requirements.

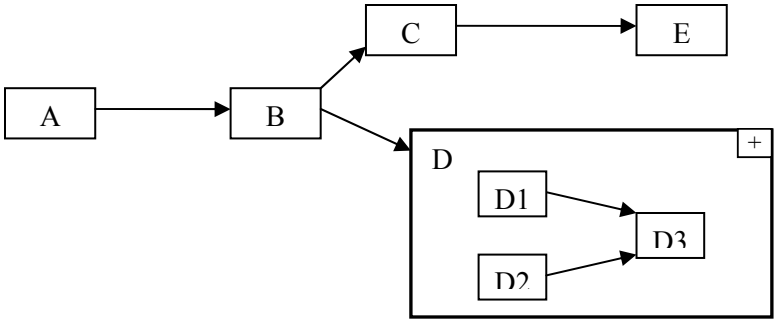


Figure 6-47. Defining a compound discrete activity by decomposition

6.3.2 Compound Continuous Activities

The ***Compound Continuous Activity*** shown in Figure 6-48 has a "+" sign in the right upper corner to differentiate it from the atomic continuous activity. The compound continuous activity could also be formed through coupling or decomposition. However, there must be at least one continuous component in the defined compound continuous activity.

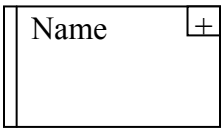


Figure 6-48. Compound Continuous Activity Node

As shown in Figure 6-49, one discrete activity and three continuous activities have been coupled into one compound continuous activity, Activity Line1. In Figure 6-50, Line1 is connected with another two compound continuous activities — Line 2 and Line 3, and a compound discrete activity — Mobilization.

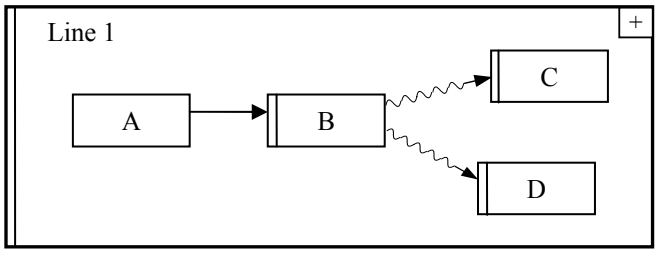


Figure 6-49. Example of a compound continuous activity

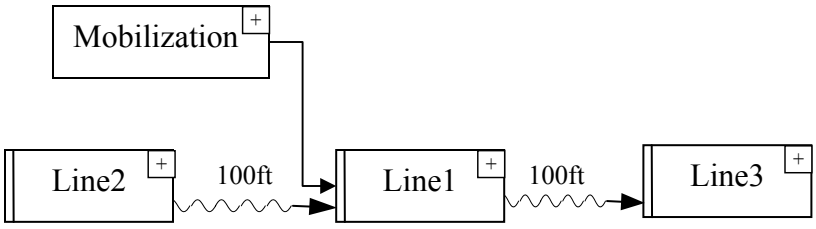


Figure 6-50. Adding links to compound continuous activities

The start, finish, interrupt, resume and adjust links between the compound continuous activity and its components are routed in the same way as with the compound discrete activity (refer to Section 5.2.2.1 and 5.2.2.2 for the routing rules). So the start-up constraints from Mobilization will be imposed on Activity A, the component that does not have any predecessors in Line1.

The rules governing the routings of the buffer constraints have been presented in Section 5.2.2.3. According to these rules, when the component Activity B, C or D attempts to start or advance, they are restrained by the buffer constraints imposed on Activity Line 1. And since this buffer constraint is imposed by Line 2, which is also a compound continuous activity, the buffer distance has to be maintained between all the continuous components in Activity Line 2 that have started but not finished.

6.4 Repetitive Activities

6.4.1 Defining Repetitive Activities

The repetitive activity is a special type of the compound activity. It is divided into the ***Repetitive Discrete Activity***(shown in Figure 6-51(a)) and the ***Repetitive Continuous Activity***(shown in Figure 6-51(b)). Instead of the plus sign, the repetitive activity has the number of the repetitive units/segments indicated in the upper right corner of the activity box.

(a) Repetitive Discrete Activity (b) Repetitive Continuous Activity

Figure 6-51. Repetitive Activities

Figure 6-52 illustrates how to define a repetitive discrete activity. The selected components are duplicated 10 times to form the repetitive discrete activity Phase II shown in Figure 6-53.

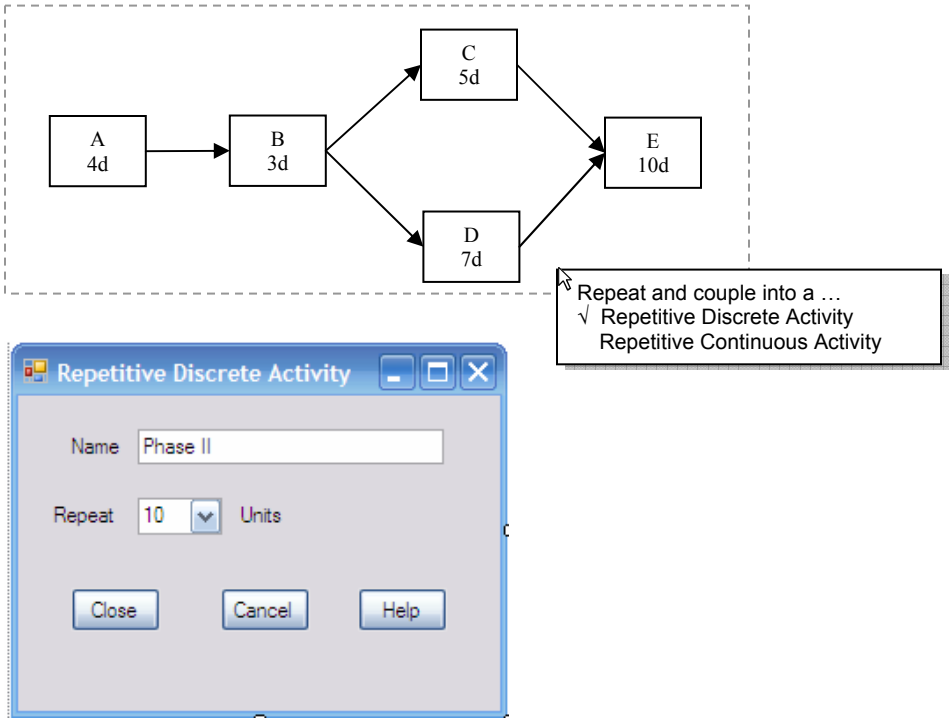


Figure 6-52. Defining a repetitive discrete activity

In the newly formed repetitive activity, all units have the same set of activities, and all activities have the same durations, work quantities, and resource requirements. Therefore, Activity A in all units of Phase II will be assigned a duration of 4 days. Click

on the number "10" in the upper right corner, the repetitive activity will expand, as shown in Figure 6-54, and the attributes of the individual activities could be changed. Activities can also be added to or deleted from each individual unit network in the expanded mode. Click on the "10 units" again, the repetitive activity will collapse and the range of the durations for each activity will be displayed.

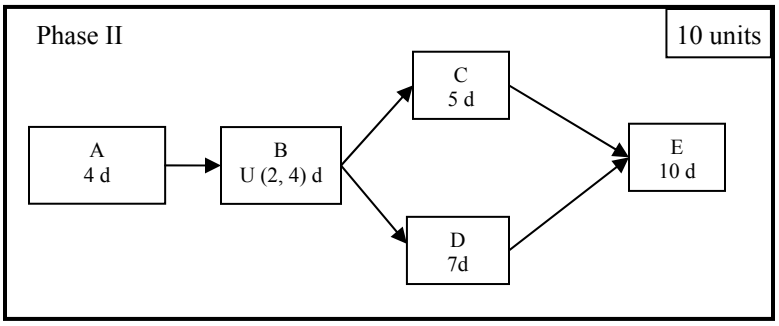


Figure 6-53. Example of a repetitive discrete activity – collapsed mode

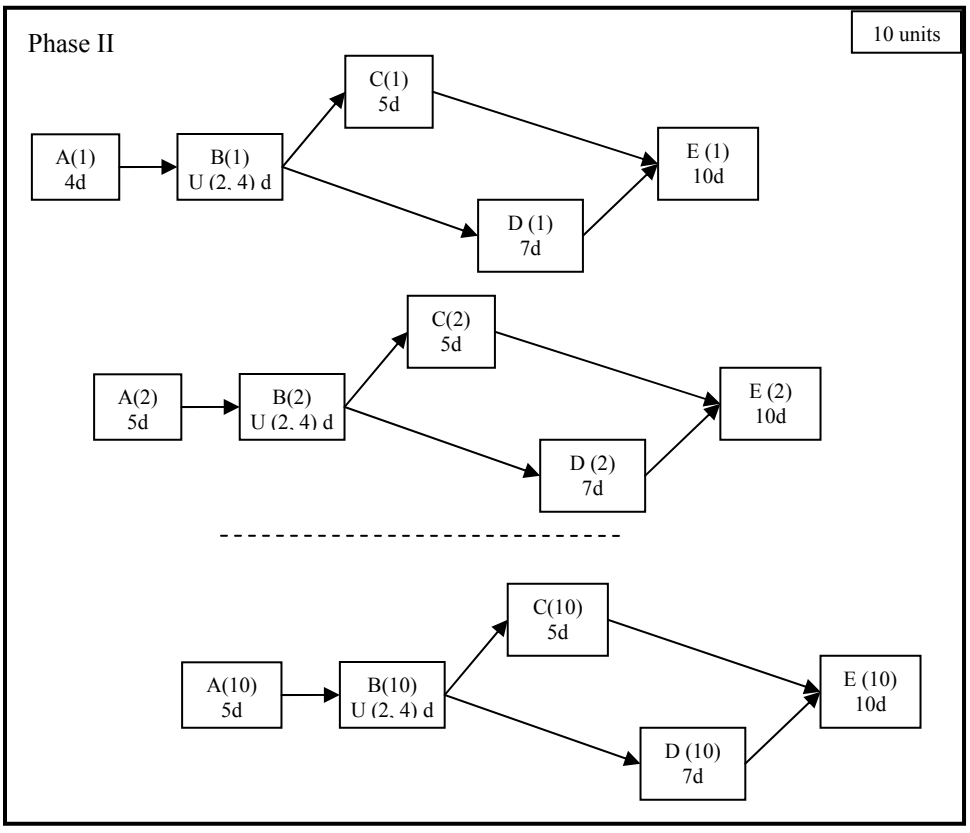


Figure 6-54. Defining a repetitive discrete activity – expanded mode

As can be seen in the expanded mode in Figure 6-54, a variable *i* (*i=1 to 10*) has been assigned to the activities in the repetitive *Unit (i)*. So, for example, Activity C (5) refers to Activity C in Unit 5. This variable *i* is very important when adding links and assigning resources to repetitive activities, as will be discussed below.

Repetitive continuous activities are defined in a similar way, except that a key component (which must be continuous) needs to be designated. Note that all activities duplicated from that designated key component will become key components and thereby be controlled by the buffer constraint imposed on the repetitive activity.

6.4.2 Assigning Resources

Notice that in Figure 6-40, no links have been added among the units, this means that activities of the same type have not been sequenced. When the start constraints on the repetitive activity is satisfied, all A(i) *(i = 1 to 10)* can start at once. With the proposed method, sequencing of activities among the units is modeled through resource assignments. By changing the available number of resources and the priority settings, various resource utilization strategies could be easily represented.

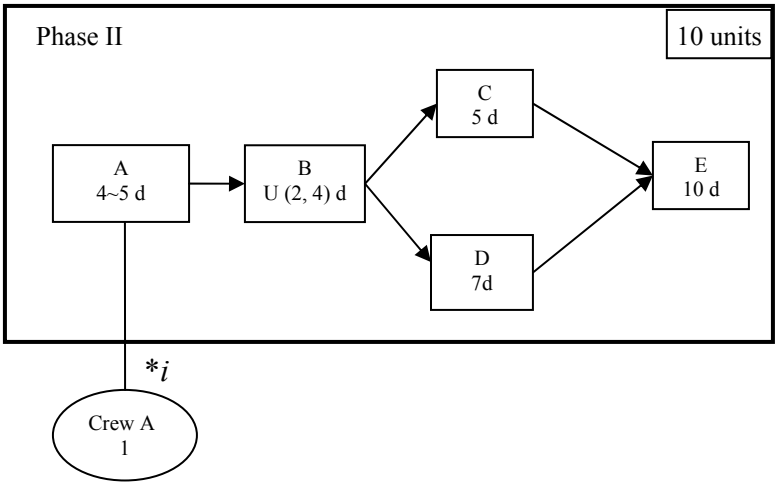
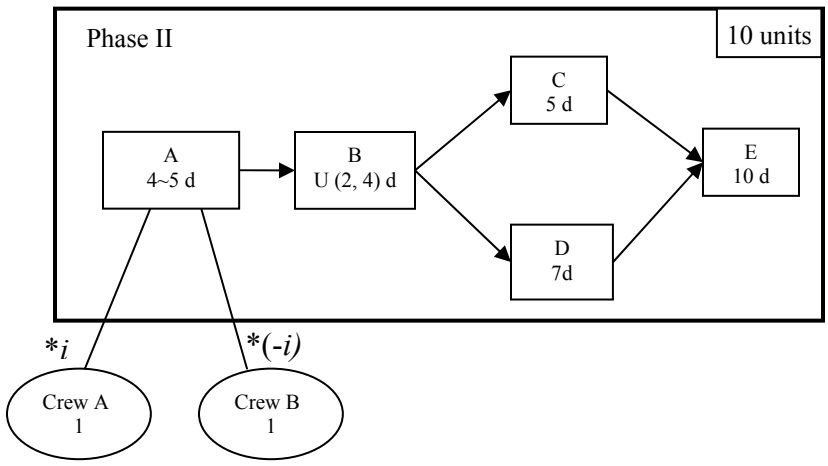
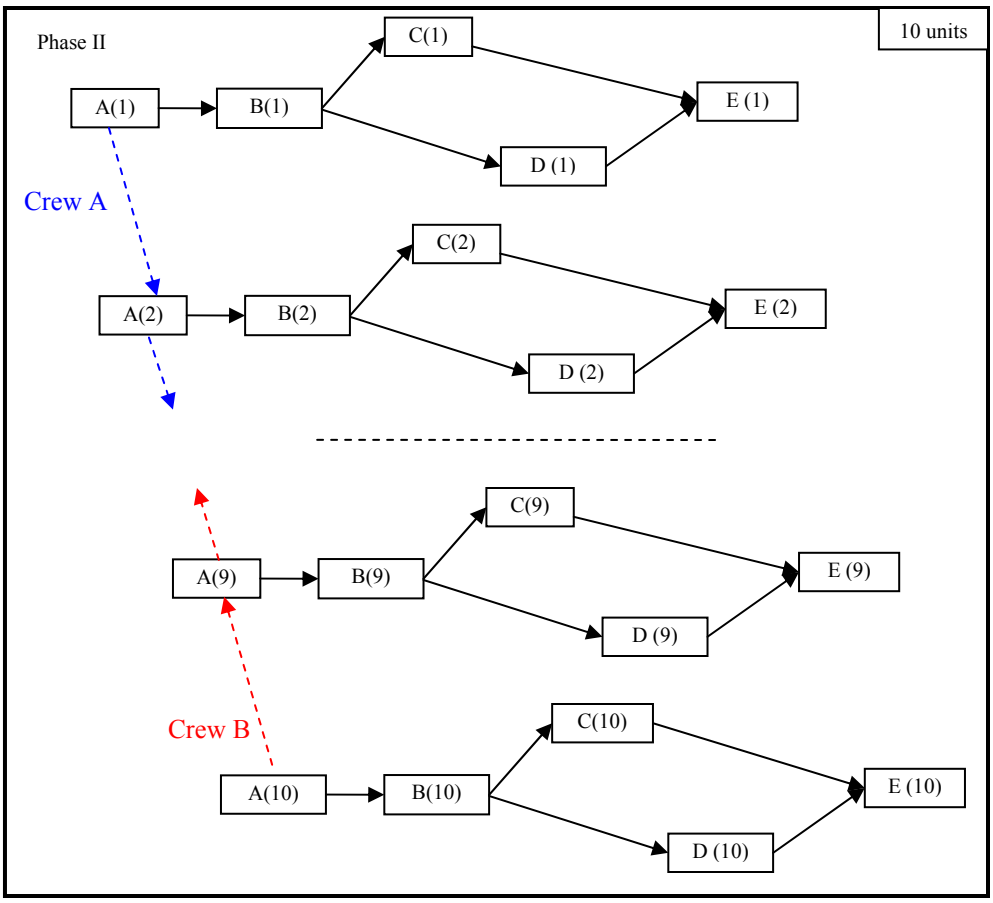


Figure 6-55. Assigning resources to the repetitive activity

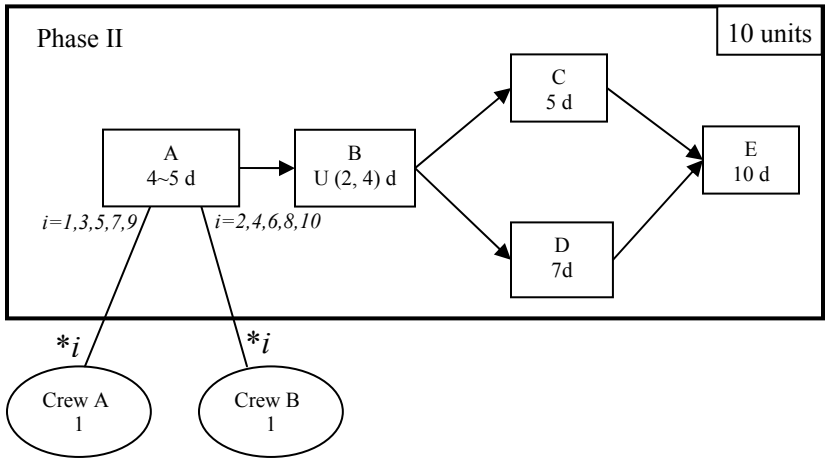


(a) Collapsed Mode

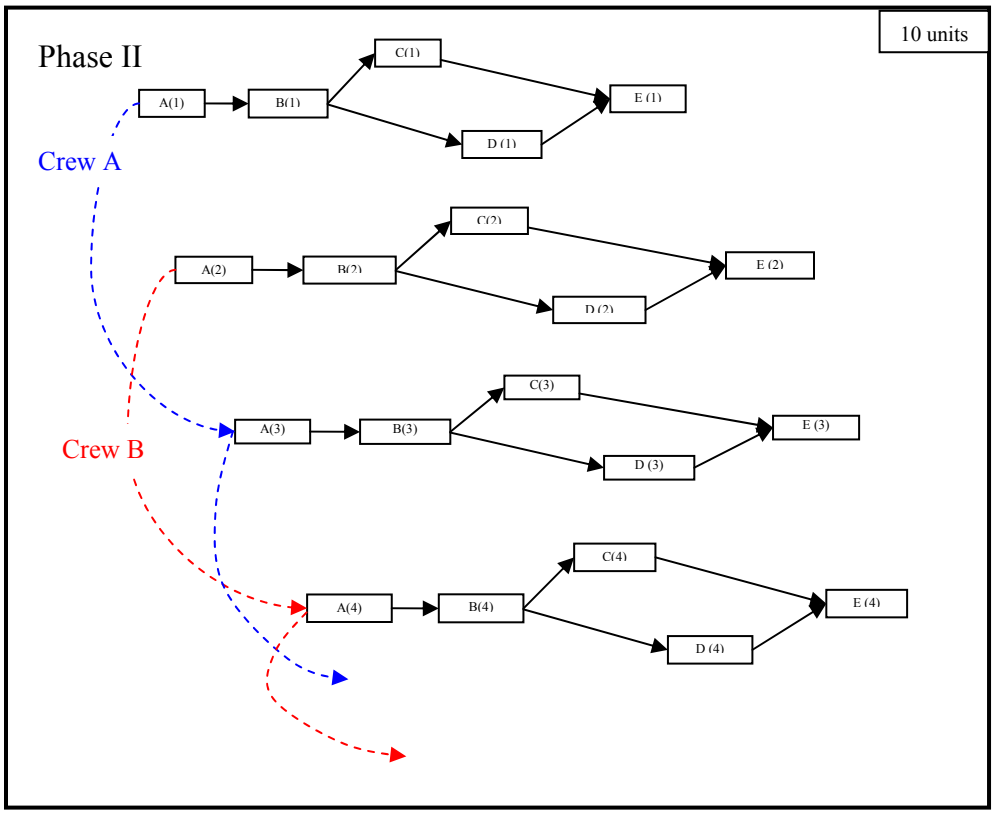


(b) Expanded Mode

Figure 6-56. Resource-imposed sequencing in the repetitive activity: two crews proceed toward each other



(a) Collapsed Mode



(b) Expanded Mode

Figure 6-57. Resource-imposed sequencing in the repetitive activity- two crews in alternating units

In Figure 6-55, one Crew A is assigned to Activity A, so only one A(*i*) could be worked on at any time. As the priority of the activities are set as their unit number *i*, Crew A is going to work in the order of A(1), A(2) … A(10). If the priority is defined as "*(-*i)*", Crew A is going to start from A(10) and finish at A(1).

In Figure 6-56(a), Crew B has been added to work on Activity A with the priority set as "*(-*i*)". So Crew A and Crew B will be working towards opposite directions as illustrated in Figure 6-56(b). Figure 6-57 shows another example where Crew A and Crew B work in alternate units in the same direction.

6.4.3 Identification and Repeating of the Links

As a special type of compound activity, the repetitive activities follow all the rules governing the linking of compound activities, but there are two additional usages about the links that are designed to address their unique repetitiveness.

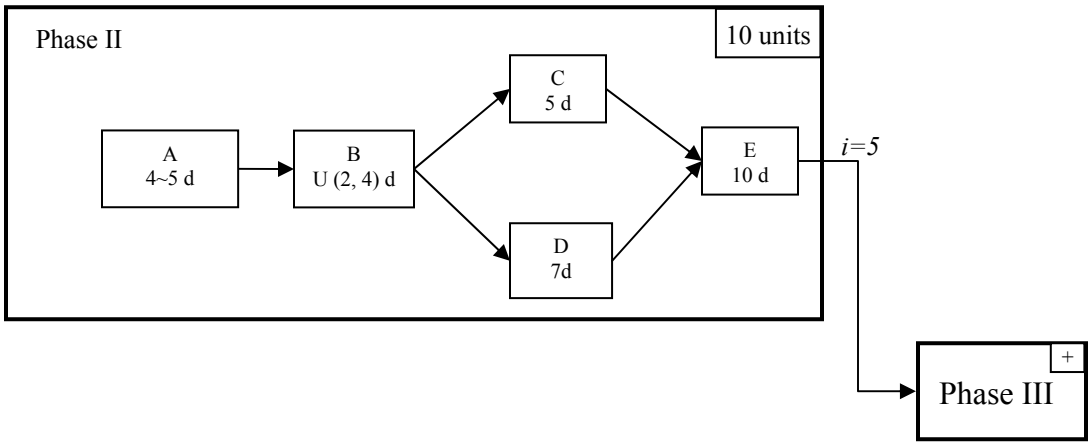


Figure 6-58. Identification of the link in the repetitive activity

One is ***identification of the link***. In the example shown in Figure 6-58, Phase III will start after Unit 5 in Phase II has finished. One FS dependency relationship has been added between Activity E in Phase II and the compound activity Phase III. To indicate

that this link is only connected with Activity E(5), "i=5" is annotated at the end of the link near Activity E. Note that the identification can be used at either the tail end or the head end of the links.

The other is ***repeating of the link***. All links, including the start/finish links, interrupt/resume links, adjust links, and buffers can be used to show repetitive relationships simply by changing their tails from a single line to a double line. Figure 6-59 shows an example of using the repetitive link. In the construction of high-rise buildings, to protect the drywalls from weather, it is often required that dry wall should not be put on until the building envelope on upper floors have been finished. As illustrated below, we can specify that Exterior Wall on the 6th floor has to be finished before Dry Wall on the 1st floor can start, Exterior Wall on the 7th floor has to be finished before Dry Wall on the 2nd floor can start,…, Exterior Wall on the 100th floor has to be finished before Dry wall on the 95th floor can start, with just one repetitive link.

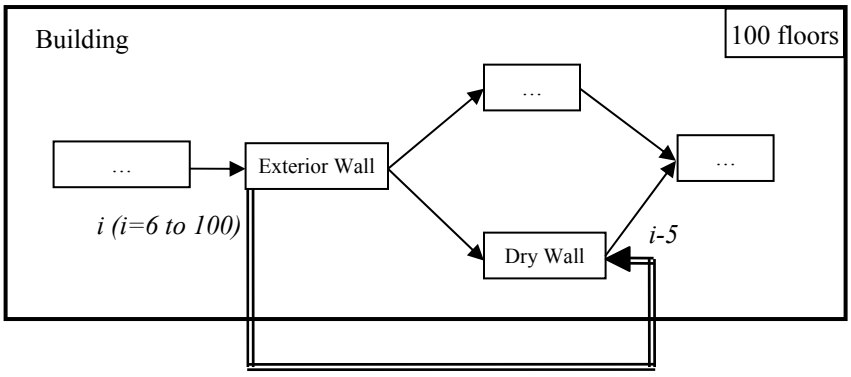


Figure 6-59. Repetitive link within a repetitive activity

The repetitive links can also be used between two repetitive activities. Suppose that Building 1 has 100 floors, and Building 2 has 50 floors. Since these two buildings are connected, Activity P in Building 2 on each floor cannot start until Activity D in building

1 on the same floor has completed. This relationship can be modeled by a simple repetitive link as shown in Figure 6-60. Because this is an "i^{th} to i^{th} " relationship, the "i" identification at both ends have been omitted and only the applicable scope of i has been noted.

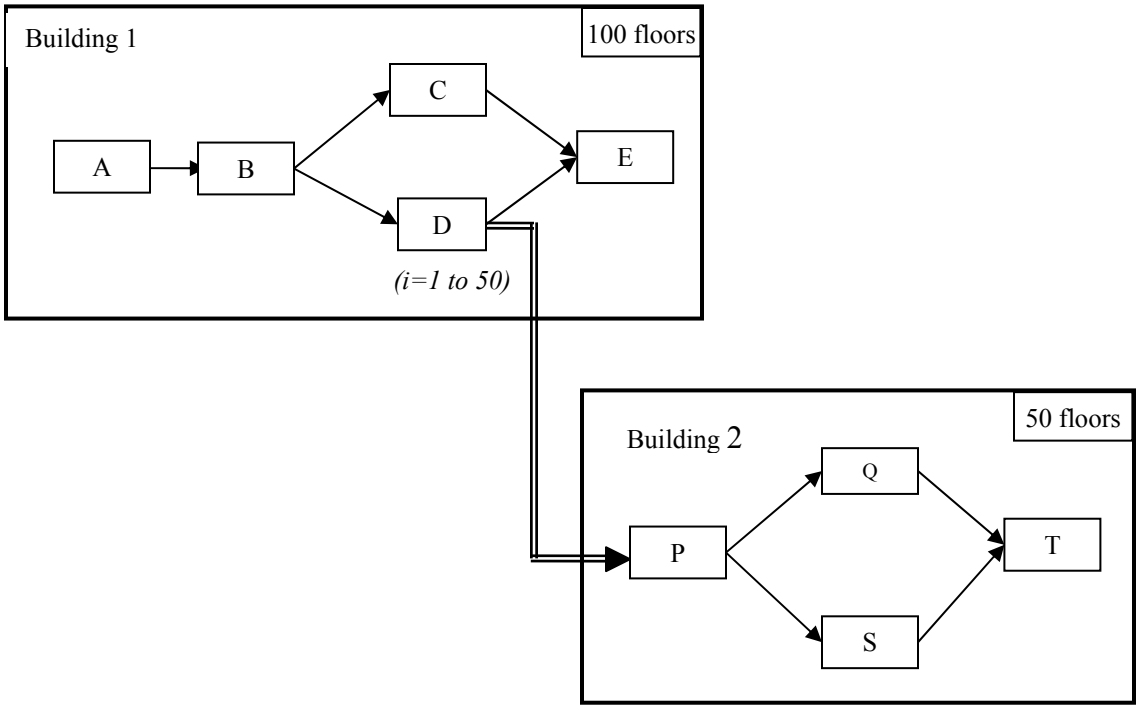


Figure 6-60. Repetitive link between repetitive activities

CHAPTER 7
CASE STUDIES WITH THE PROPOSED METHOD

In Chapter 3, four case studies are presented, each representing a typical class of construction project. In this chapter, the proposed method is applied to those four cases, and the new models are compared with those developed with the existing methods.

7.1 Case One: A Typical Regular Project

The project examined in the first case study has been described in Section 3.1. Figure 7-1 shows the model developed with the proposed method for this case study. This model will be compared with the CPM model (in Figure 3-1) and the STROBOSCOPE model (in Figure 3-5) on the modeling of activities, dependency relationships and environmental factors.

7.1.1 Modeling of Activities

7.1.1.1 Duration

Similar to the STROBOSCOPE method, the proposed method is able to represent both deterministic and probabilistic durations, and determine the values dynamically. In Figure 7-1, the duration of the activity Foundation is defined as a probability distribution of "U (25, 35)[*] days". We can also specify that if the foundation is poured in winter, it would take much longer to get the job done. Defined as "IIF (March< time. month < Nov, U(25,35), U(35,45))" (*time* is a system variable that record when the activity attempts to start), the duration of the Activity Foundation would be U(25,35) if the activity starts between March and November; and U(35,45) otherwise.

[*] U(,): Uniform Distribution

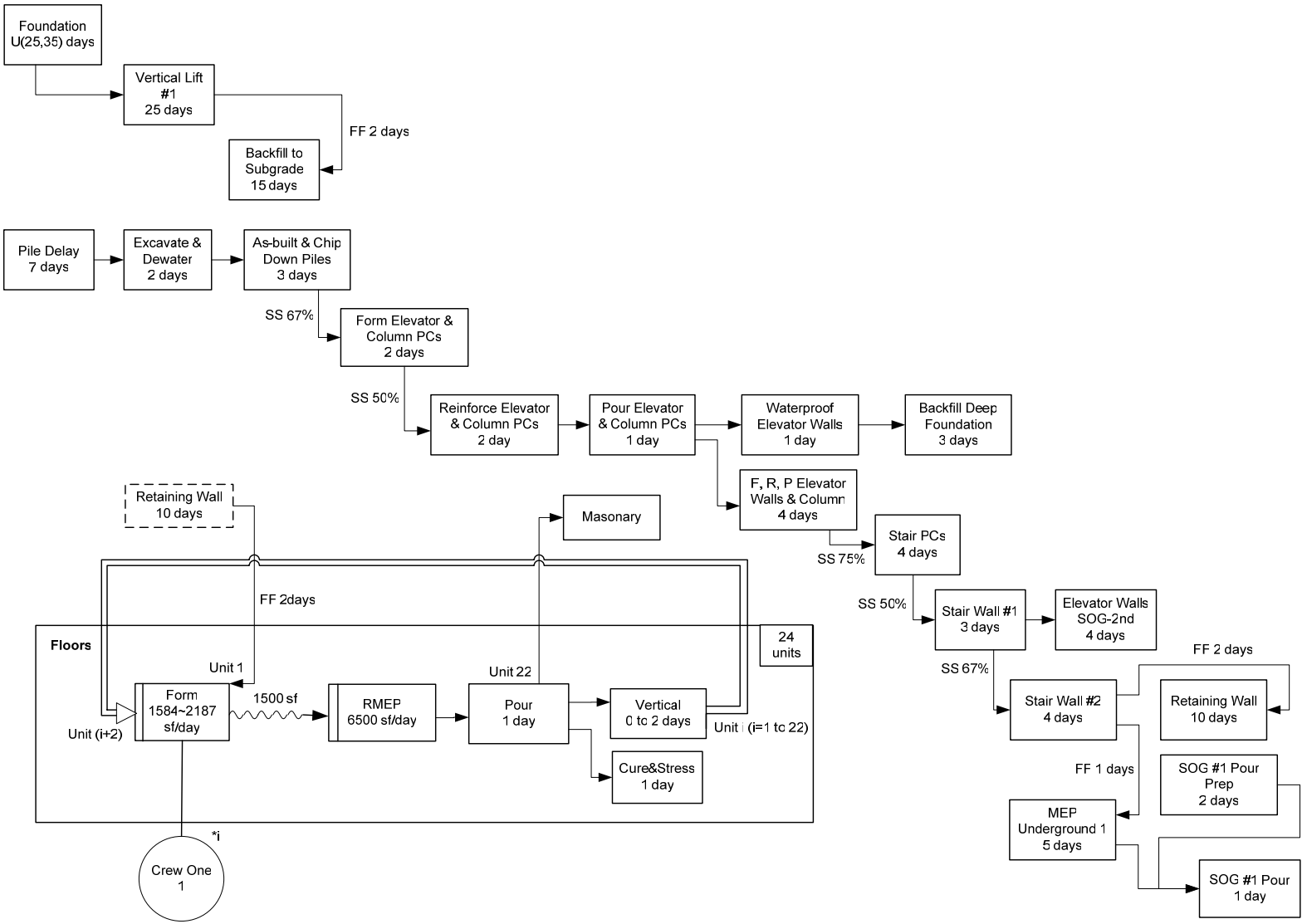


Figure 7-1. The proposed model for Case One

7.1.1.2 Progress curve

The CPM model and the STROBOSCOPE model are not able to represent the progress curves of the activities. In the proposed method, by default, all discrete activities proceed at a constant rate from start to finish, i.e., have a Type I — straight progress curve. The Activity Foundation is defined to have a Type II curve, which takes account of the learning curve effect. The progress curve can also be customized. For example, to show that the foundation is poured in five separate parts and each part is not considered done until completely finished, we could define that the progress function is "STEP (0≤t<.2, 0%, .2≤t<.4, 20%, .4≤t<.6, 40%, .6≤t<.8, 60%, .8≤t<100, 80%, t=1.0, 100%)", as illustrated in Figure 7-2.

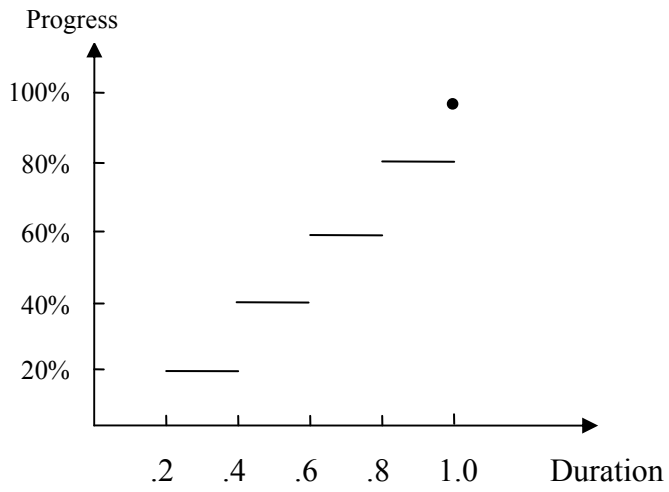


Figure 7-2. Customizing a stepwise progress curve

7.1.1.3 Interruption and adjustment

Different from the other two models, the defined durations of the discrete activities could be changed by interruptions and adjustments in the proposed new model. In Figure 7-1, the Activity Foundation would stop when the temperature drops to below 40 degrees and resume

when the temperature exceeds 45 degrees; and its remaining duration would be extended to 1.4 times as the construction of the Vertical Lifts starts.

7.1.2 Modeling of dependency relationships

7.1.2.1 FS, SS, FF and SF dependency relationships

The representation of the FS, SS, FF and SF relationships and lag times in the new model are exactly the same as in the CPM method. The user does not need to generate virtual tokens, divide activities, or add extra nodes as in the STROBOSCOPE model.

7.1.2.2 Non-time based dependency relationships

Except for the time-based dependency relationships, the proposed method is also able to represent progress-based dependency relationships. So, the requirement that "the vertical lift on the first floor cannot start until the foundation is at least 1/3 finished" can be directly represented as in Figure 7-1, and does not need to be converted into an SS relationship with a 10 day's lag. If Activity Foundation's duration changes, or its progress curve is non-linear, or the activity has been interrupted or elongated, this progress-based relationship will still be accurate.

7.1.2.3 Compound constraints

The representation of the "AND" logic in the proposed method is as easy as in the CPM method, as can be demonstrated with the activity SOG #1 Pour (enlarged in Figure 7-3).

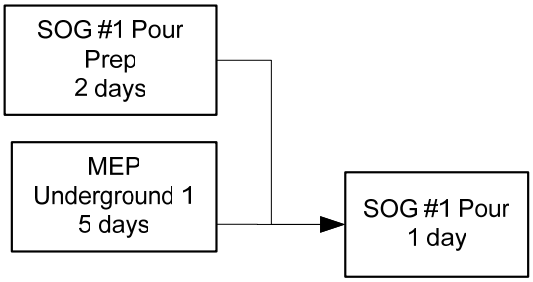


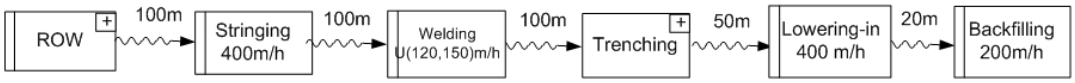
Figure 7-3. Representation of the "AND" logic in the proposed method

7.1.3 Modeling of environmental factors

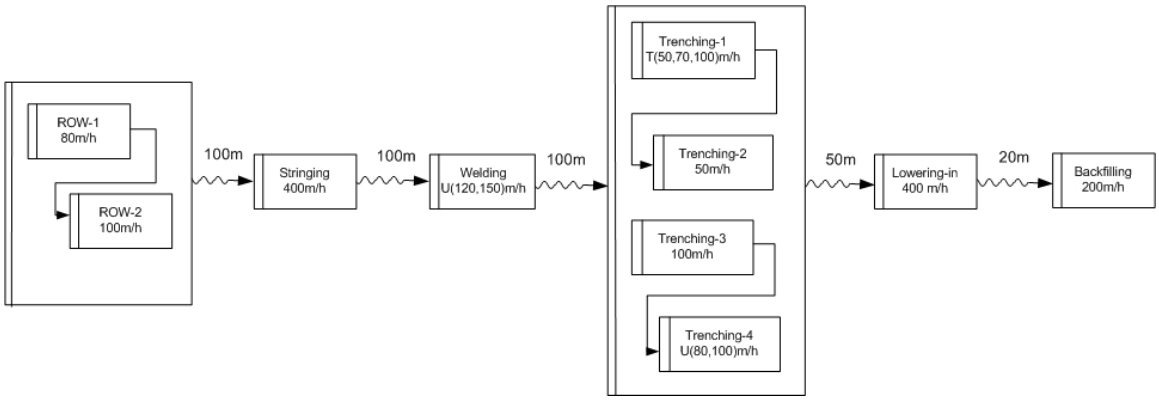
The environmental factor temperature is represented as variable *temp = p (50, 15)* [*] in the environmental factor "Clock 1", whose interval is 2 hours. The impacts of weather and wind speed can be included in similar ways.

7.2 Case Two: A Typical Linear Project

In Section 3.2, the construction of a gas line has been discussed as the example for the linear projects. A model has been developed with the proposed method for this project. Figure 7-4 (a) shows this model with the compound activities collapsed; in Figure 7-4 (b), the compound activities have been expanded.



(a) Collapsed Mode



(b) Expanded Mode

Figure 7-4. The proposed model for Case Two

In this project, the buffers need to be measured on distance, and the paths of the activities need to be tracked. The point where the project starts is designate as Point "0", and the direction

[*] P (,): Poisson distribution.

toward which all activities are progressing is taken as the "positive" direction. The paths of the activities are defined as shown in Table 7-1. Since the default values (0 for the "Start Position", positive for the "Direction") apply to most of the activities, only a few activities need to be defined specifically.

Table 7-1. Defining the path of the activities for Case Two

Activity	Start position	Direction
ROW-1	0	Positive
ROW-2	6000	Positive
Stringing	0	Positive
Welding	0	Positive
Trenching-1	0	Positive
Trenching-2	3000	Positive
Trenching-3	3500	Positive
Trenching-4	7500	Positive
Lowering-in	0	Positive
Backfilling	0	Positive

Compared with the LSM model in Figure 3-10 and the SLAM II model in Figure 3-13, the proposed model has the following characteristics:

7.2.1 Modeling of Activities

7.2.1.1 Productivity

As SLAM II, the proposed method allows both deterministic and probabilistic productivities. But the proposed method is also able to represent the impacts of dynamically changing factors on the productivities. For example, the productivity of Activity Welding is sensitive to temperature. This can be modeled by defining the Productivity Modification Factor as a function on the variable *temperature*.

7.2.1.2 Slow-down and break

With the proposed method, it can be easily modeled whether an activity will slow-down or break when being blocked by its preceding activity. Figure 7-5 shows the dialogue window defining the buffer between the activity ROW and Stringing. With the LSM method, if these options changes, the whole diagram needs to be redrawn.

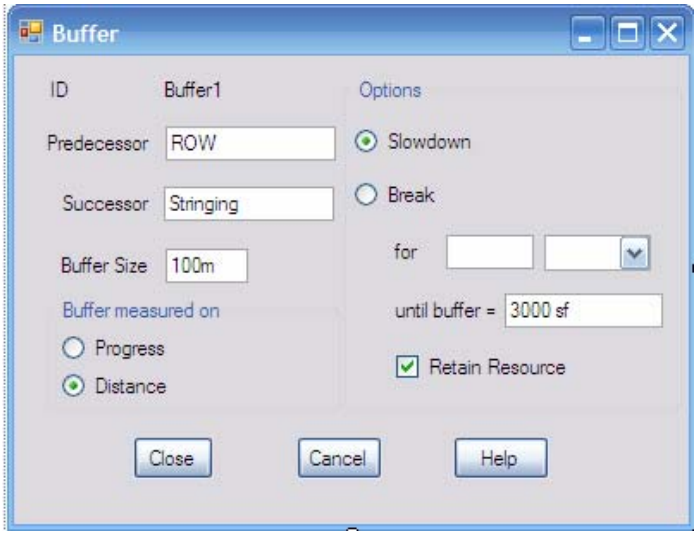


Figure 7-5. Defining the buffer between ROW and Stringing

7.2.1.3 Position of activities

With the defined paths and directions of the activities, the proposed method can determine positions of the activities at any time based on the simulated progress values. For example, if at time *t* the progress of Activity Trenching-3 is 1000m, it can be determined that it is (3500 + 1000 =) 4000m away from the project start point. Should the activity goes toward the negative direction, it would be at the (3500 -1000 =) 2500m position at this point. The result can be plotted as a LSM diagram (if the layout is one-dimensional) for further graphical analysis.

7.2.2 Modeling of Buffers

The proposed method can maintain the buffer between two activities measured on progress or distance. The buffers are represented as links, similar to the discrete type of dependency

relationships. Though this representation is not as graphically intuitive as the actual "buffers" shown on a LSM diagram, it is much easier to draw and change. Compared to the SLAM II simulation method, where the buffer constraint has to be coded in subroutines, the advantages of the proposed method in this area are even more obvious.

7.2.3 Modeling of Resources

The proposed model can represent resources directly when necessary. In this project, Crew B is shared between Stringing and Lowering-in, which can be represented as in Figure 7-6. The priority of Activity Stringing to Crew B is "1" (omitted), and the priority of Lowering-in is "2". So Lowering-in cannot go on in parallel with Activity Stringing and has to wait until Stringing has been totally completed.

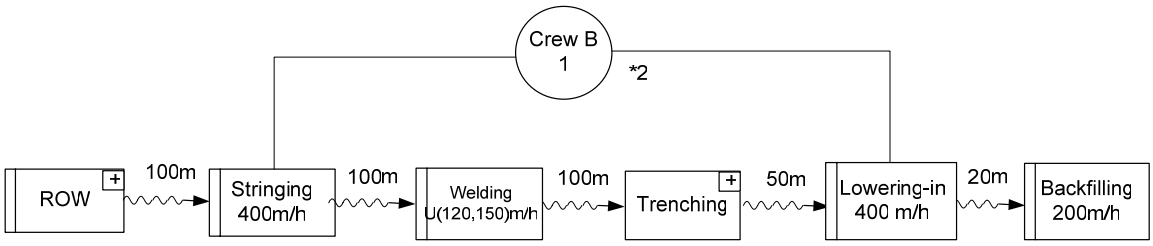


Figure 7-6. Representing resource sharing

7.2.4 Multi-level Modeling

This model can be viewed on two different levels: the expanded model shows all activities at the detailed level, the collapsed model emphasizes the big picture and the overall arrangement.

The multi-level modeling ability of the proposed method enables both the top-down and bottom- up approach. At the initial planning stage, the project manager may create an activity Trenching as an atomic continuous activity and specify its external relationships with other activities; and later on as more information become available, break it down into smaller, more controllable activities and focus on the internal relationships within the activity. Or in the other way, the project manager may first have the foreman of trenching work out the details of this

activity: how it is going to be divided into sections, which crews are going to be assigned to each section, etc; then he can couple these activities into one compound activity and look at it from a higher point of view to coordinate it with Activity Welding and Lowering-in.

7.2.5 Multi-dimensional Layouts

Following this single utility line project, we can further examine the site development project discussed in Section 3.2.3, which consists of two intersecting utility lines. This example can demonstrate the ability of the proposed method for modeling linear projects with multi-dimensional layouts.

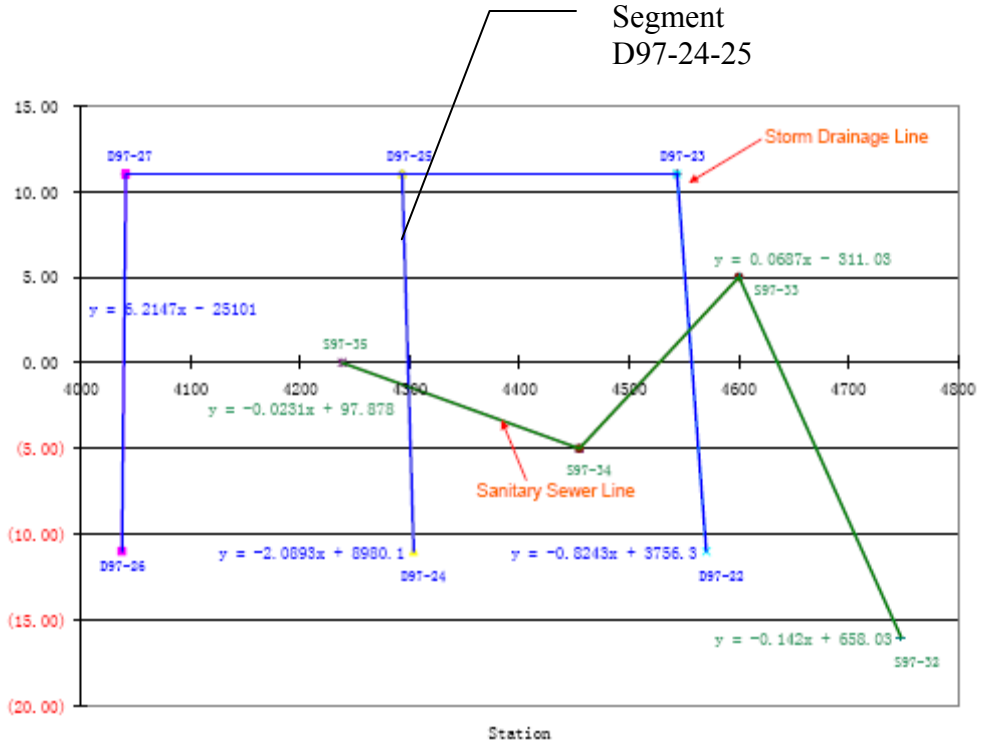


Figure 7-7. Site layout of storm drainage Segment D97-24-25

For convenience, the site plan of this project is again presented in Figure 7-7. Suppose that there are 200 segments in the storm line and 220 segments in the sewer line, the two activities can be represented as in Figure 7-8. As the two activities have different layouts, the paths of their

segments need to be defined in a two-dimensional coordination system. Figure 7-9 shows the dialogue window defining the path of Segment D97-24-25 in the storm line.

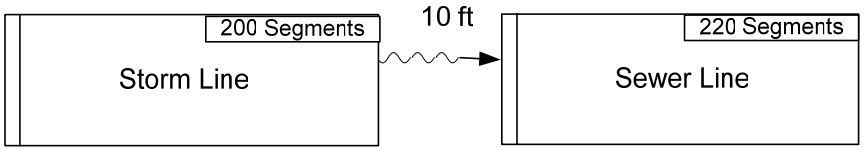


Figure 7-8. Modeling a project with multi-dimensional layout

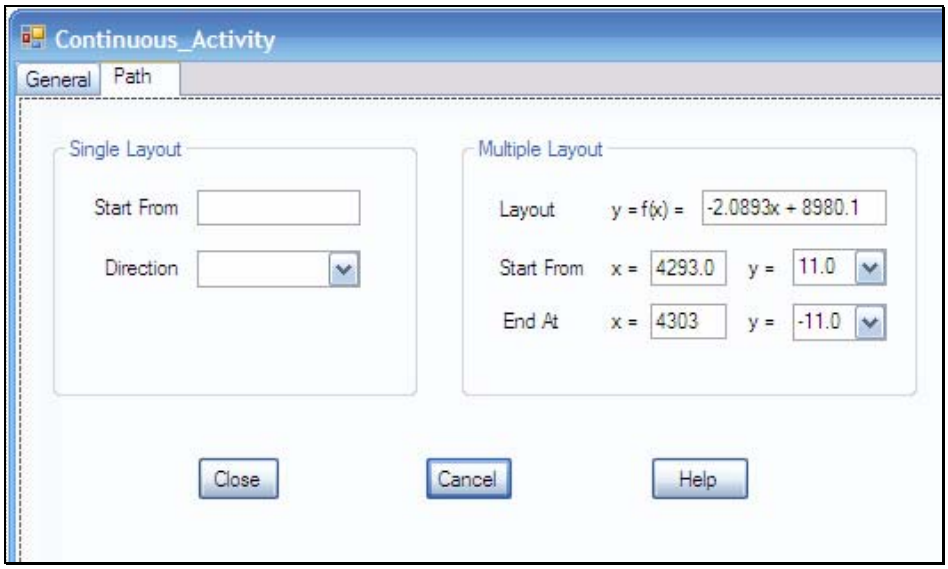


Figure 7-9. Defining the path of an activity in a project with multi-dimensional layouts

With the defined paths, at any time *t*, the current positions of the crews working on the segments can be determined from the activity's progress value. When any segment in Activity Sewer Line (denoted as $Sewer_i$) attempts to start or advance, it will be restrained by the 10ft external buffer constraint. Suppose that there are two crews working on the Activity Storm Line and currently two segments are under construction (denoted as $Storm_p$ and $Storm_q$), the following constraints will be checked for each $Sewer_i$:

$$Distance\ between\ Sewer_i\ and\ Storm_p \leq 10\ ft,$$

and

$$Distance\ between\ Sewer_i\ and\ Storm_q \leq 10\ ft;$$

in which the distance is measured between the current positions derived from the progress values.

7.3 Case Three: A Typical Repetitive Project

The third example has been described in Section 3.3. The CPM model, LSM model and STROBOSCOPE model for this example are presented in Figure 3-15, Figure 3-16 and Figure 3-19 respectively. Figure 7-10 shows the model developed with the proposed method.

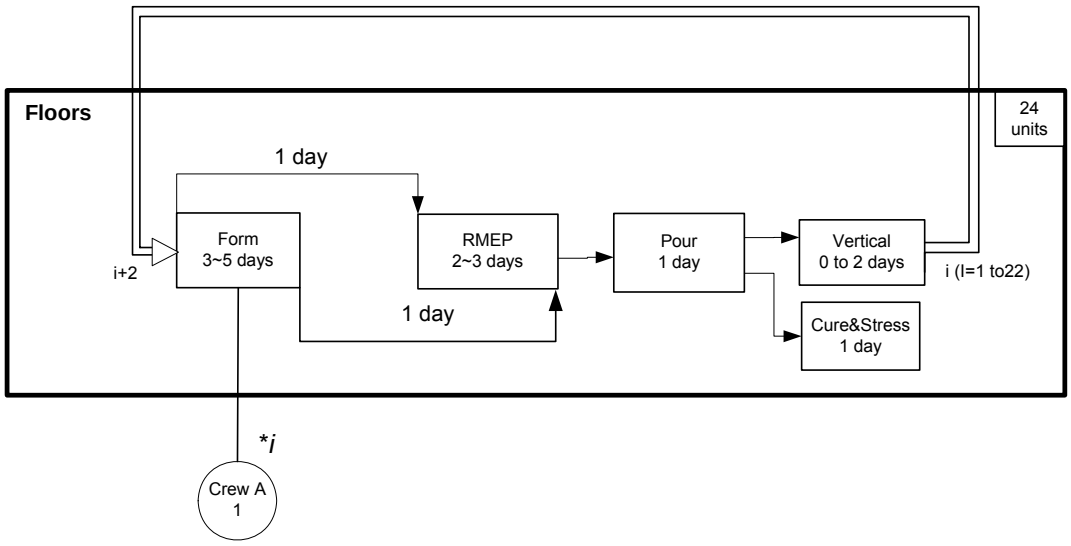


Figure 7-10. The proposed model for Case Three: one crew

The whole project is represented as a repetitive activity containing 24 repeating units, each unit for one pour. This model has the following characteristics compared with the other models:

7.3.1 Efficiency

Compared to the CPM model and the LSM model which occupies more than two pages, this model is obviously more efficient in representation. It is also much simpler than the STROBOSCOPE model — it contains only about half as many nodes as the STROBOSCOPE model does. Certainly, the expanded model contains many more nodes, but it is seldom

necessary to expand it except when we need to find out detailed information in a specific unit for Activity Form or Activity RMEP, which have variations among different units.

7.3.2 Resource-imposed Constraints

Among the existing models, only the STROBOSCOPE model is able to represent the resource-imposed constraints. In the proposed method, the resource-imposed constraints can be represented easily and flexibly. Figure 7-10 shows that one Crew A is going to work on Activity Form, and the units will be processed from Unit 1 to Unit 24 (the priority is defined on the unit identification variable *i*).

If there are two Crew As that can work on this activity, we can simply change the available number of the resource to 2. If the two crews are of different types, and Crew A is preferred over Crew B, the model can be quickly changed to Figure 7-11, in which "*2" is noted on the resource link connected with Crew B to indicate its priority.

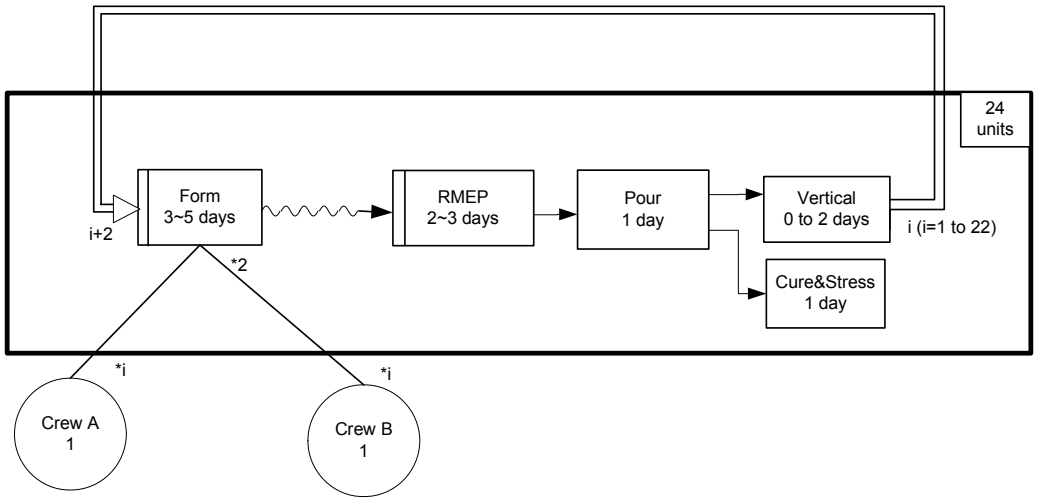


Figure 7-11. The proposed model for Case Three: two crews

Suppose that one Crew A is going to work on the odd-numbered units, and one Crew B is going to work on the even-numbered units, the situation can be represented as shown in Figure 7-12.

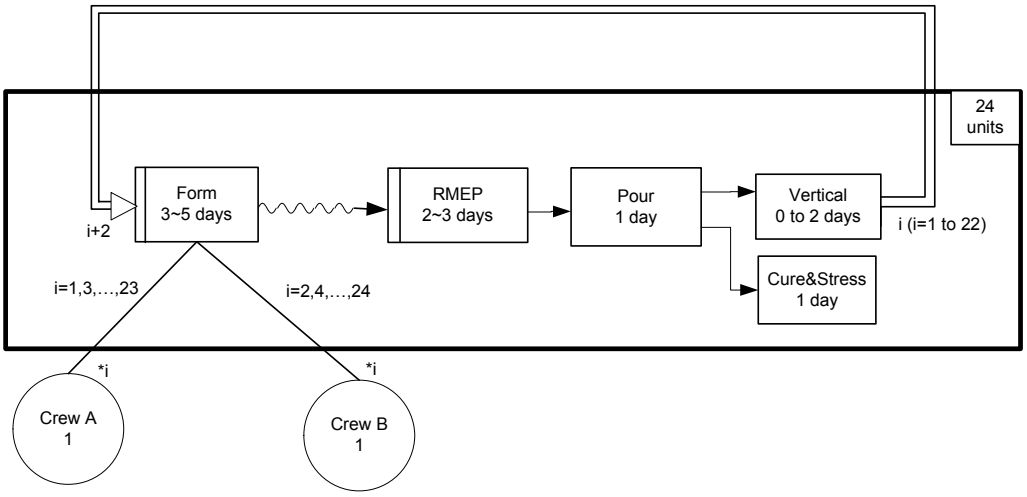


Figure 7-12. The proposed model for Case Three: two crews in alternating units

7.3.3 Hetero-relationships

As discussed in Section 7.3, the representation of the hetero-relationships is one of the biggest challenges in the modeling of repetitive activities for the LSM method and the STROBOSCOPE method. In the proposed method, hetero-relationships can be easily modeled by repeating and identifying the links.

For example, as shown in Figure 7-10 to 7-12, a repetitive link is drawn between Activity Form and Activity Vertical. It shows that Vertical 1 is connected to Form 3(=1+2), Vertical 2 is connected to Form 4 (=2+2), …, and Vertical 22 is connected to Form 24(=22+2), with the simple notations on the repetitive link — Vertical (*i*) (i=1 to 22) connected to Form (*i*+2). No coding is required as in the STROBOSCOPE method.

7.4 Case Four: A Project with Mixed Features

The project examined in Case Four includes both the foundation and the upper-structural parts of the 14-level Condo, as described in Section 3.4. The complete model developed with the proposed method is shown in Figure 7-13. It demonstrates the ability of the proposed method in the integration of different features, including:

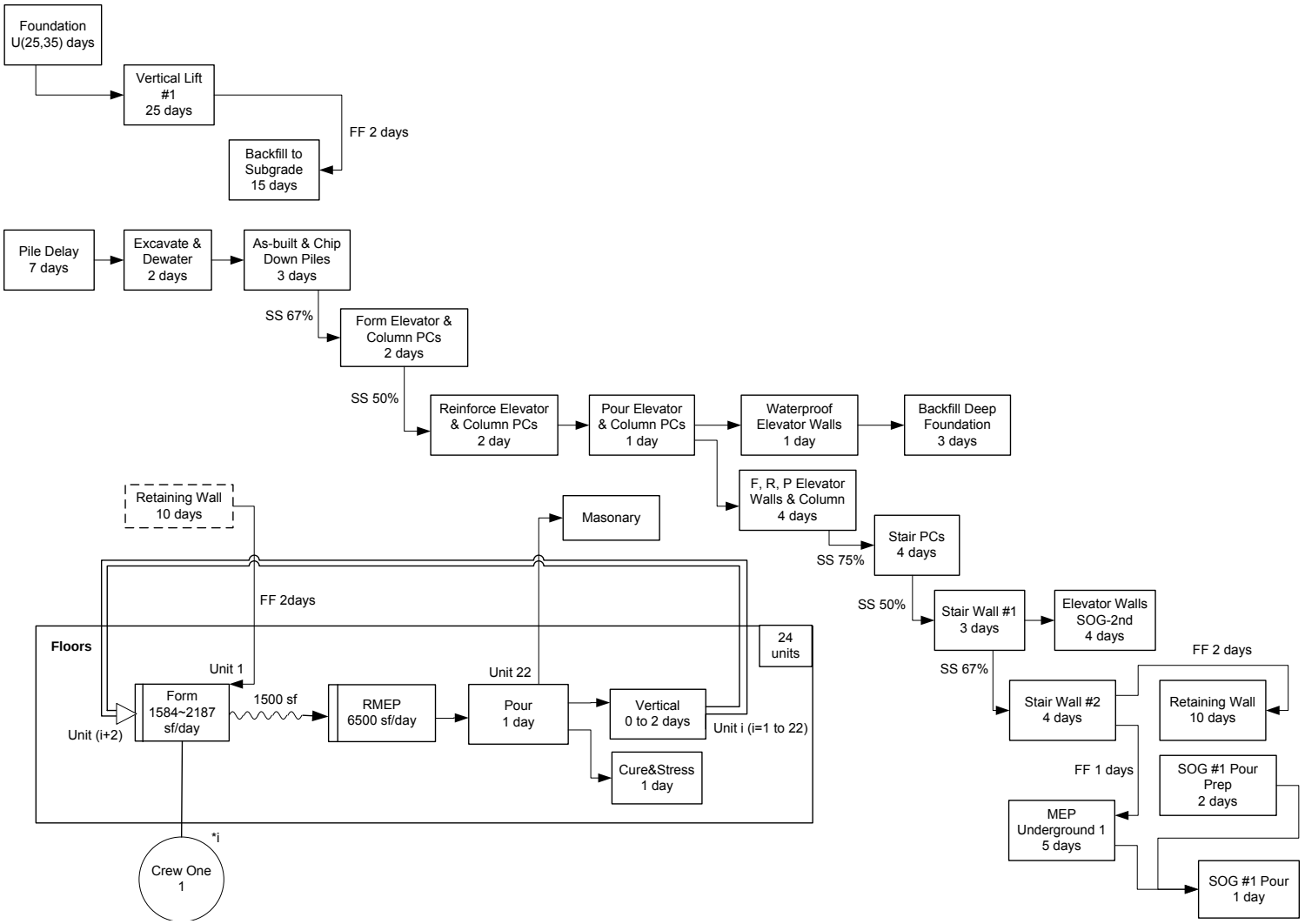


Figure 7-13. The proposed model for Case Four

7.4.1 Integration of Discrete and Continuous Activities

In Figure 7-13, Activity Form and Activity RMEP are represented as continuous activities, for it is required to ensure the 1500 SF buffer between them. These two continuous activities are integrated seamlessly in the project where the majority of the activities are discrete activities: they can be connected with the discrete activities with discrete types of links, and can be coupled with the discrete activities to form compound activities and repetitive activities.

7.4.2 Integration of Repetitive and Non-repetitive Activities

As shown in Figure 7-14, the enlarged upper structural part of the model, Form in Unit 1 in the repetitive activity Floor cannot be finished until 2 days after the non-repetitive activity Retaining Wall (the border of the box is dotted as this activity is duplicated in this model for easy of display) is finished, and Activity Masonry (a non-repetitive activity) cannot start until Activity Pour in Unit 22 in the repetitive activity is finished.

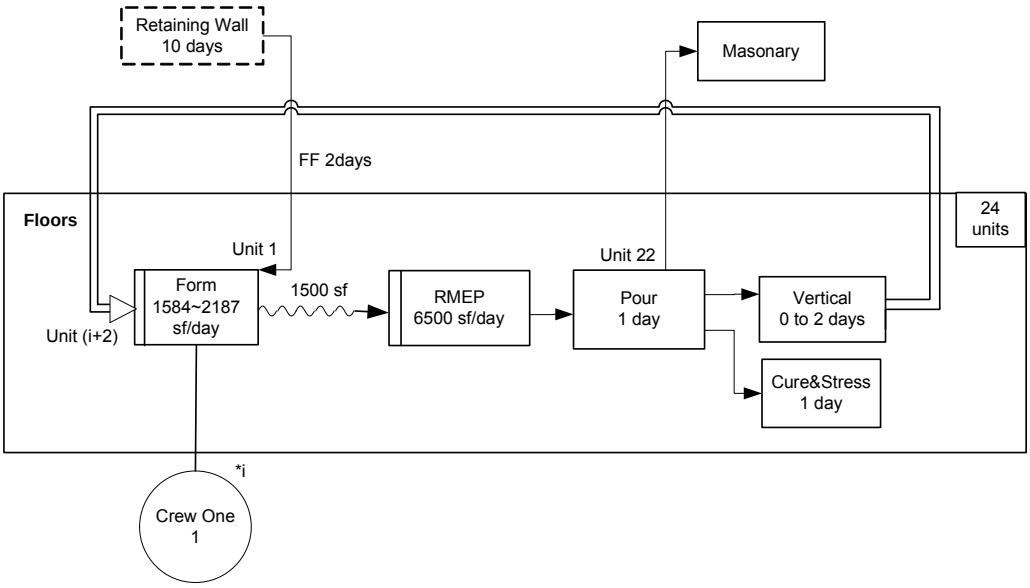


Figure 7-14. The enlarged partial model for Case Four

Unlike the STROBOSCOPE method where the repetitive and non-repetitive activities are modeled with different concepts and have to be synchronized with extensive coding work, the proposed method connects the repetitive and non-repetitive activities directly with links. Moreover, by collapsing the repetitive activities and identifying the links with the variable i, the relationships between the two types of activities are emphasized and clear, not obscured by a large number of links like in the CPM model.

CHAPTER 8
CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

Existing construction planning and scheduling methods all have a specific scope of application and some fundamental limitations that restrict their ability in the modeling of complex, realistic construction projects. In order to develop a method that is universally applicable, powerful and simple-to-use, a new theory needs to be established.

The method proposed in this research is based on the system-theory and it is formalized with the DEVS specification, which enables it to integrate any modeling formalisms and model a system at different levels. A set of graphical modeling elements have been designed so the user can easily utilize the full capacity of this method without any specialized knowledge in the system theory or computer languages.

According to the requirements for planning and scheduling construction projects as summarized in Chapter 4, the performance of the proposed method can be examined from the following four perspectives: scope of application, accuracy of modeling, form of representation and level of modeling.

Scope of application. The proposed method was applied to four case studies in Chapter 7 including a typical regular project, a typical linear project, a typical repetitive project and a project of mixed features. It was demonstrated that the proposed method is able to represent both the discrete and continuous, repetitive and non-repetitive elements and integrate them all together.

Table 8-1. Comparison of the modeling accuracy of the proposed method and major existing methods

			CPM	LSM	STROBOSCOPE	The proposed method
Discrete	Activities	Duration	Only real numbers or probabilistic distributions.		Both deterministic and probabilistic. The values are dynamically determined. Can be resource driven.	Both deterministic and probabilistic. The values are dynamically determined. Can be resource driven.
		Progress curve	No		No	Yes
		Interruption	No		No	Yes
		Adjustment	No		No	Yes
	Dependency Relationships	FS,SS,FF, SF relationships	Yes		Only FS relationships can be directly represented.	Yes
		Non-time-based dependencies	No		No	Can be based on progress or work quantity
		Compound logic constraints	Only "AND" logic.		"AND" logic can be graphically represented. "OR" logic requires code writing.	"AND", "OR" and multi-layered logic all can be graphically represented. Does not need code writing.
		Branching	No		Support both probabilistic and decision-based branching.	Support both probabilistic and decision-based branching.
Continuous	Activities	Productivity		Deterministic		Deterministic and probabilistic. Accounts for impacts of dynamically changing variables.
		Slow-down		Yes		Yes
		Break		Yes		Yes
		Layout		Only one-dimensional		Both one-dimensional and multi-dimensional.
	Dependency Relationships	Buffers		Yes		Yes
		Types		Distance-based and time-based		Progress-based

Table 8-1. Continued

		CPM	LSM	STROBOSCOPE	The Proposed Method
Resources	Representation	No	Not directly represented, but the work paths of the resources can be visualized.	Include generic, characterized and compound resources.	Resources can be defined with type, minimum/maximum quantity, and other attributes.
	Activity selecting resources	No	No	Yes, but activities can only select resources from the same queue.	Priorities of the resources can be defined on any variables.
	Resource selecting activities	No	No	Yes, but one activity can only be assigned one priority value for all the shared resources.	One activity can have different priority values to different resources.
Environmental factors		No	No	Yes, but these factors need to be modeled with separate sub-networks.	Modeled with a single node. One node can include multiple variables.
Repetitive	Representation	Not efficient	Not efficient when certain assumptions cannot be fulfilled. Difficult to show parallel activities.	Efficient	Efficient.
	Resource-imposed dependency relationships	Not directly modeled. Not flexible.	Show resource flows, but does not model the resources directly.	Yes	Flexible in modeling multiple crew strategies.
	Hetero-relationships	Must be represented one-by-one.	Difficult.	Difficult. Require extensive code writing.	Does not require code writing.
Integration	Discrete and continuous	No	Yes	No	Yes
	Non-repetitive and repetitive	Yes	No	Yes, but difficult.	Yes

Accuracy of modeling. Table 4-1 summarizes the major properties and behaviors that a universally applicable planning and scheduling method needs to incorporate in order to accurately represent the projects described in the case studies. Table 8-1 compares the proposed method and the major existing methods on their abilities to represent these factors based on the cased studies conducted in Chapter 3 and Chapter 7. It was demonstrated that the proposed method has overcome many of the limitations of the existing methods.

Moreover, this method is highly expandable. Theoretically, there is no limit on what properties and behaviors it is able to represent. Properties such as cost, material consumption and physical dimensions can be added as additional parameters to the activity models, and their changes can be linked to the progress of the activity and simulated dynamically. The resource model can also have additional parameters such as calendars, maintenance schedules and required working conditions to make it more accurate.

Form of representation. The proposed method is built with a black-box approach — the users can fully utilize the functions of the modeling elements without knowing their inside structures and the underlying theory. As a result, the representation of the proposed method is *easy to learn and understand*. It uses a representation similar to the CPM AON diagram. As shown in Case One, for typical regular projects, the proposed model could be just the same as the CPM model. In other words, if the user just wants the same level of accuracy in the modeling of regular projects, there is no difference between the models built with these two methods. But the proposed method is able to extend the same style of simple representation to other types of projects and much more complex situations. It can be just as effective as or even more powerful than the existing simulation methods including STROBOSOCOPE and SLAM II for the

modeling of construction projects, without the need of using the Combis, Queues and any simulation languages.

It is also very *efficient* in modeling and editing. Table 8-2 compares the number of nodes, links and lines of codes used by different methods for the four case studies. As it shows, the proposed method uses the least number of nodes and links for all of the four cases, and does not require the user to write computer codes as the STROBOSOCOPE and SLAM II simulation methods often do. What is more, unlike the LSM models, which require a whole model to be redrawn every time an activity or dependency relationship needs to be added, deleted or changed, the editing of the proposed models are quick and easy. Compared to the CPM, STROBOSOPE and SLAM models, the advantage of the proposed models on the efficiency of editing is also obvious if the compound activities, repetitive activities and resource-imposed dependency relationships could be fully utilized.

Table 8-2. Comparison of the representation efficiency of the proposed method and the major existing methods

	Case One (regular)			Case Two (linear)			Case Three (repetitive)			Case Four (mixed features)		
	N*	L*	C*	N	L	C	N	L	C	N	L	C
CPM	20	18	0	–	–	–	152	190	0	177	212	0
STROBOSCOPE	31	29	0	–	–	–	13	14	5	55	53	9
SLAM II	–	–	–	65	75	27	–	–	–	–	–	–
Proposed Method	20	18	0	17	11	0	7	7	0	28	26	0

*N: number of nodes
*L: number of links
*C: lines of codes

Moreover, it is *natural and intuitive*. Existing simulation methods for construction planning and scheduling all rely on the ACD diagrams, which are shaped by the underlying

three-phase simulation strategies. They require all activity nodes to be linked by the traveling of the resources. Many abstract nodes, such as the Generate node, Combi node, Queue node, have to be used to manipulate the flow of the resources. Logical dependency relationships often cannot be directly represented and have to be implemented by manipulating the virtual tokens through many tricks.

The proposed models have smaller gaps with realistic construction projects. All of the modeling elements maps directly to real-world objects or concepts and own the same attributes and behaviors as their real-world counterparts. Both the logical constraints and the resource-imposed constraints can be directly represented. So the user can think and communicate with terms used in construction, rather than terms used in simulation or any program-specific languages.

Multi-level modeling. One of the biggest advantages of the proposed method is that it establishes a hierarchical construction for the activities and thereby enables modeling at different levels. This feature can significantly improve efficiency in model building and editing, provide insight for complex projects, and support both the bottom-up and top-down scheduling approach.

8.2 Limitations and Recommendations for Future Research

This study is just the first step in introducing the system theory into construction planning and scheduling. Although the system theory and the DEVS Formalism contain no fundamental limitations for modeling complex dynamic construction projects, the proposed method has several major aspects that need to be improved:

- Currently, the method only measures the time performance of the project. To measure the project performance in other aspect, such as cost and material consumption, additional parameters and transitional functions need to be added to the activity models.
- A hierarchical structure needs to be established for the resources. In a construction project, crews are formed with one or more types of labors and equipment. Individual resources often need to be assembled and dissembled into different crews. A hierarchical

construction will be able to represent the assembling of resources, and simplify model constructions under many situations.

- For repetitive activities, an option needs to be provided to ensure resource continuity. Maintaining resource continuity can reduce the idle time and maximize the learning curve effect for the repetitive activities. To maintain resource continuity, the starting time of the first unit in the repetitive activity needs to be strategically delayed. An algorithm needs to be developed to search for the starting point to enable this option.

Besides enhancing the above areas, the proposed method also need to be:

- Implemented on computer. Chapter 5 has described the DEVS specifications for the major models in the proposed method. These models can be implemented with the DEVSJAVA or the DEVS/C++ packages provided by the Arizona Center for Integrative Modeling & Simulation(available on http://www.acims.arizona.edu/SOFTWARE/software.shtml, last accessed on March 28, 2008). Also, an interface needs to be developed to compile the graphical models to DEVS languages.

- Field tested. In this study, the proposed method is verified with case studies. A future research should be conducted to evaluate its effectiveness and efficiency with empirical experiments. The major data need to be collected include: the average learning time a user needs before he or she can use this method to represent a fairly complex construction project, the average time a user spends on modeling a certain project with this method compared with other methods, the number of errors in the models developed with this methods compared with that in other models, etc. The users can also be surveyed on their opinions regarding the ease-of-use, modeling ability and limitations of this proposed method.

APPENDIX A
FOUNDAMENTALS OF SIMULATION TECHNOLOGY

Overview of Simulation

A *(dynamic) simulation* is an imitation of the operation of a real-world process or system over time (Banks et al. 1999). Compared to analytical methods, simulation has the advantages of being able to address randomness and dynamics in real-world systems (Zhang et al. 2002). Simulation modeling assumes that "a system can be characterized by a set of variables, with each combination of variable values representing a unique state or condition of the system" and thereby "manipulation of the variable values simulates movement of the system from state to state(Prisker and O'Reiley 1999)". In a simulation experiment, the system's status, i.e., the variable values, evolve dynamically according to operation rules that have been pre-designed into the model. The performance of the system can be evaluated by analyzing the statistics of the variables, or by directly observing the animated system components, as supported by some simulation software.

Discrete vs. Continuous Simulation

In (dynamic) simulations, time is the most important independent variable. Other variables are functions of time and are the dependent variables.

According to the behaviors of the dependent variables, simulation models can be classified into two basic categories. In *discrete simulation models*, the dependent variables changes only at discrete points in simulated time referred to as *event times*, as shown in Figure A-1. In *continuous simulation models*, the dependent variables may change continuously over simulated time (Prisker and O'Reiley 1999), as shown in Figure A-2.

Note that "continuous" and "discrete" are modifiers to the simulation model, not to the actual system that the model aims to represents. A real-world system often can be modeled either

as a discrete model or a continuous model, depending on the objectives of the study (Pidd 1998; Prisker and O'Reiley 1999) . As described by Pidd (1998) in an example where an underground railway is being considered, if the analyst only concerns about the time it takes for a passenger to travel between stations, then the system should be simulated with a discrete model; but if the speed of the train as it travels between stations is of interest, then the system has to be viewed from a continuous prospective.

It has also been realized that discretely changing and continuously changing variables can co-exist in one system model. But it was until recently that a few simulation software systems allow users to program such combined discrete-continuous models, including SlamII (Prisker and O'Reiley 1999) and Extend (Krahl 1996).

Continuous Simulation Technology

In a continuous simulation model, the state of the system is represented by dependent variables which change continuously over time. These variables are referred to as state variables to be distinguished from discretely changing variables. To construct a continuous simulation model, the modeler needs to define the derivative $\left(\frac{ds}{dt}\right)$ for every state variable s, over time t, together with *s(0)* , the initial value of *s* at time 0. Theoretically, this allows the values of the state variables to be computed at any point of time by integrating Equation A-1:

$$S(t_2) = S(t_1) + \int_{t_1}^{t_2} \left(\frac{ds}{dt}\right) dt \qquad \text{(Equation A-1)}$$

However, analytical solutions to differential equations are not always attainable. Hence, integration of the differential equations is usually performed using numerical methods which divide time into small slices referred to as steps, and calculate state variables by employing an

approximation to the derivatives of the variables over the time step. The accuracy of these methods depends on the order of the selected approximation algorithm and the size of the step. An efficient algorithm can adjust the size of the step by increasing it when the change is smooth, and decrease it when the change is abrupt.

For some continuous systems, the derivatives $\left(\frac{ds}{dt}\right)$ do not change very fast and can be regarded as constant during short time periods. These continuous systems can be modeled using Equation A-2:

$$S_{k+1} = S_k + r \times \Delta t \,, \qquad \text{(A-2)}$$

where S_{k+1}, the value of a state variable S at step $k+1$, is derived from its previous value S_k at time k, by adding the increment $(r \times \Delta t)$ between the two steps. Time has been disretized into fixed steps of length Δt, and the rate of change r is assumed constant for any Δt. Figure A-3 shows approximation of a state variable in a continuous simulation model using fixed time steps

Discrete Simulation Technology

In management science, discrete simulation is much more commonly used than continuous simulation. The technology of discrete simulation can be discussed from four perspectives: time advancement techniques, simulation strategies, graphical modeling representations, and simulation languages.

Time Advancement Techniques

Time advancement techniques deal with how to advance the simulated time in the simulated system. The *fixed time-step* method can be used in the time handling of discrete simulation models as well as in the continuous simulation models as discussed above. Actually, it is the simplest way that mimics the natural flow of time. However, for discrete simulation

models, dependent variables change only at discrete event times, in other words, they change much more sparsely than the state variables do in continuous simulation models. It would be unnecessary to examine and update the models at each time step; rather, time can be advanced from one event to the next without any lost of information. This timing mechanism is referred to as the *next-event* technique, and simulations employed this technique are called *discrete-event simulations*. The *next-event* time advancement technique is the corner stone of the discrete simulation technology.

Simulation Strategies

Simulation strategies generally guild how the real-world system is going to be decomposed into entities and mimicked. Different simulation strategies employ different control mechanisms to select the next-event and manage the simulation time during a experimentation. There are three most common discrete-event simulation strategies: event scheduling (ES), activity scanning (AS) and process interaction (PI).

The relationship between the concepts of events, activities and processes are illustrated in Figure A-4. An activity consists of one start event and one end event, and a process consists of a sequence of events and/or a number of activities.

With ES, the modeler needs to identify the events that can change the state of the systems and then determine all possible consequences associated with each event type. A simulation of the system is produced by executing the events in a time-ordered sequence(Banks et al. 1999; Pidd 1998). ES is at the lowest level of simulation programming (modeling elements in AS and PI based models will eventually be translated into events and get simulated, but this process is hided from the user). It has complete flexibility and computation is fast. The problem with this strategy is that, it is very difficult to ensure that all possible consequences are accounted for

within the developed event routine, especially for complicated systems. Therefore, ES is seldom used alone, but often in combination with AS and PI to enhance their flexibility.

With AS, the modeler decompose the system to constituting activities and describe the conditions which cause an activity to start. During the simulation, every time the simulation clock advances, the entire set of activities will be scanned. If the conditions for starting an activity are satisfied, the appropriate action is taken (Pidd 1998). These actions typically include acquiring the requested resources, determining how long the activity will last, holding the acquired resources for the determined duration (when the activity starts), and release the resources (when the activity ends). Because all activities need to be scanned every time the simulation clock advances, the simulation runs much slower than its ES-oriented counterparts.

Three-phase AS is a modified approach that improves AS's computing efficiency. With this strategy, the activities are separated into Bs (activities that are bound to start at a predictable time), and Cs (activities that are not dependent on the simulation clock but must wait until predefined conditions can be satisfied or until requested resources are available). As the simulated time advances, only Cs will be scanned and tested, while Bs will be immediately executed once the simulation clock reaches the scheduled time.

AS is particularly well suited to situations where activities have very complex startup or ending conditions, or where resources with distinct properties must collaborate according to highly dynamic rules (Hooper 1986). Most AS-based simulation languages were developed in the 1960s, including GSP, CSL, and HOCUS.

The third type of discrete-event simulation strategy is PI. With this strategy, the modeler needs to identify the entities (or transactions in some cases) that flow through the system and describes the life cycles for each class of them. Typically, the life cycles of the entities involves

entering the system, undergoing some processing where they capture and release scarce resources, and then exit (Martinez and Ioannou 1999). These describe life cycle processes will be translated into a sequence of events by the simulation executive. During runtime, entities will be created and pushed though the life cycle, triggering the events on the path in sequence. This strategy is particularly suited to systems where the moving entities are differentiated by many attributes while the machines or resources that serve these entities have few attributes, and do not interact too much (Hooper 1986). As most manufacturing and service systems are of this type, a lot of commercial simulation systems are based on PI, including GPSS, SIMAN, SLAM, ModSim, etc.

Graphical Modeling Representation.

The simulation strategies provide a guide on how to abstract a real-world system into a conceptual model. The conceptual model can be represented, to some extent, by graphical diagrams, which can then be translated to simulation languages that the computer can recognize.

A major graphical modeling tool used for discrete-event simulation is the activity cycle diagram (ACD). ACDs evolved from wheel charts which was developed for the GSP simulation language(Tocher 1964), and was later popularized by Hill in his HOCUS simulation system (Hill 1971). An activity cycle diagram is a map that shows the life cycle of each class of "flowing" entities and displays graphically their interactions. There are only two types of symbols in the original activity cycle diagram: circles called queues that representing the dead state of the entities, and squares called activities that representing the active state. The path of a moving entity usually consists of alternating circles and rectangles connected together with links. Paths of different types of entities cross where collaborations take place. A common approach to draw an activity cycle diagram is to draw the activity cycles of each type of entities first and then

combine those cycles into one network by joining the activities that they have in common (Senior and Halpin 1998).

ACDs are a natural means for representing AS or three-phase simulation models, and have been proved usable for ES and PI models (Mathewson 1974). They are often used as the blueprint at the conceptual level — too much detail will obscure the diagram. As commented by Pidd (1998), though activity cycle diagrams are seldom able to include the full complexity of the system being simulated, they do provide a clear skeleton that can be enriched and enhanced later.

Simulation Languages

A simulation model can be built with general-purpose languages, special-purpose simulation languages or simulators (Kelton et al. 2004). Simulation with general-purpose languages like FORTURN, PASCAL and C++ is very customizable and flexible, but time-consuming and error prone, even with the help of support packages and pre-written libraries. Special purpose simulation languages like GPSS, Simscript, SLAM and SIMAN, provide a much better framework that meet the needs of most types of simulations. Nevertheless, users still need to invest considerable time to learn their syntax and how to use them effectively. By contrast, high-level simulators are very easy to use. They provide graphical interfaces, menus and dialogue boxes so users can construct a model without any programming. However, high-level simulators usually are limited to a certain application domain or even a certain kind of problem. As a rule, as ease-of-use increases, flexibility compromises.

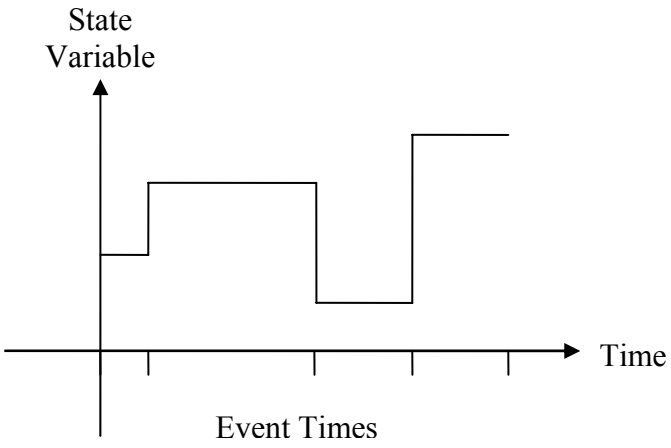


Figure A-1. A dependent variable in a discrete simulation model

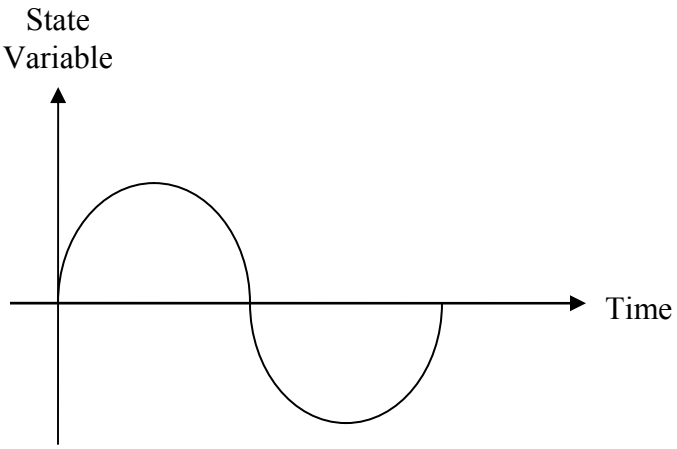


Figure A-2. A state variable in a continuous simulation model

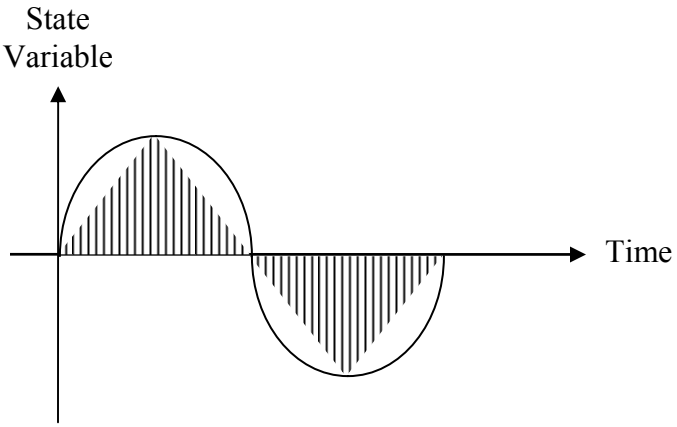


Figure A-3. Approximation of a state variable in a continuous simulation model using fixed time steps

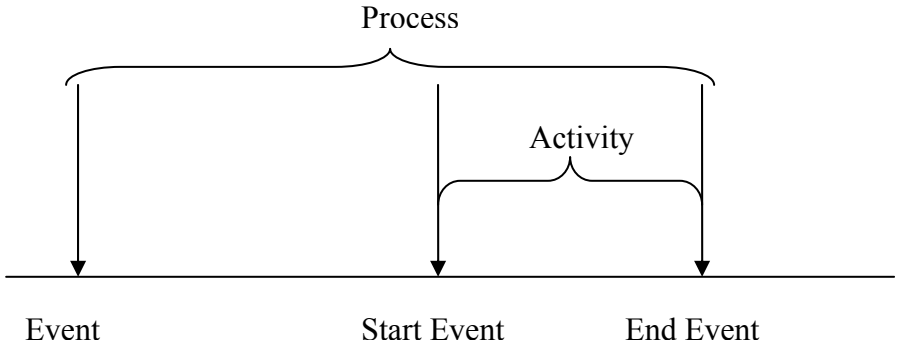
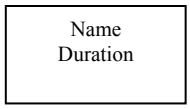
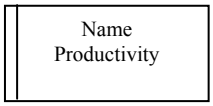
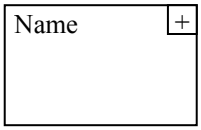
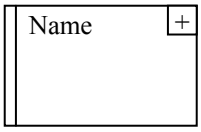
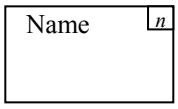
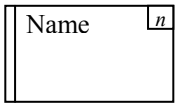
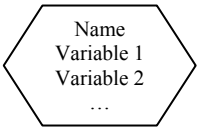
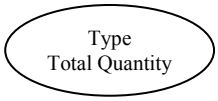
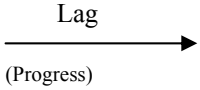
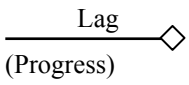
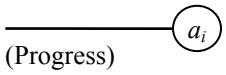
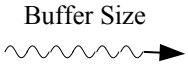
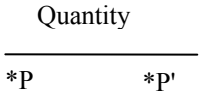
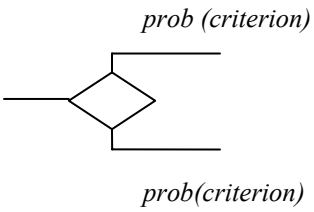


Figure A-4. Relationships among events, activities and processes. [Adapted after Prisker, A. A. B., and O'Reiley, J. J. (1999). *Simulation with Visual SLAM and AweSim*, John Wiley & Sons, Inc, New York. (Page 395, Figure 12-2)]

APPENDIX B
MODELING ELEMENTS IN THE PROPOSED METHOD

Symbol	Name	Notes	Examples
	Discrete Activity		Figure 6-2
	Continuous Activity		Figure 6-3 to 6-6
	Compound Discrete Activity		Figure 6-32 to 6-35
	Compound Continuous Activity		Figure 6-36, 6-37
	Repetitive Discrete Activity	*n*: number of repetitive units	Figure 6-38 to 6-43
	Repetitive Continuous Activity	*n*: number of repetitive segments	Figure 7-8
	Environmental Factor		Figure 6-7
	Resource		Figure 6-41 to 6-43
	Start/Finish Link		Figure 6-8 to 6-13

Symbol	Name	Notes	Examples
	Resume Link		Figure 6-13, 6-17
	Adjust Link	a_i: adjustment factor	Figure 6-18 to 6-20
	Buffer Link		Figure 6-23 to 6-26
	Resource Link	*p: priority of alternative resources *p': priority of activities for shared resources	Figure 6-31, 6-41
	"OR" Node		Figure 6-28, 6-29
	Branching Node		Figure 6-30
Doubling the link line	Repetitive Link		Figure 6-45 to 6-46
I	Unit Identification Variable		Figure 6-41 to 6-46

LIST OF REFERENCES

AbouRizk, S. M., and Hajjar, D. (1998). "A framework for applying simulation in construction." *Canadian Journal of Civil Engineering*, 25(3), 604-617.

Ammar, M. A., and Elbeltagi, E. (2001). "Algorithm for determining controlling path considering resource continuity." *Journal of Computing in Civil Engineering*, 15(4).

Ang, A. H., Abdelnoor, J., and Chaker, A. A. (1975). "Analysis of activity networks under uncertainty." ASCE Journal of the engineering Mechanics Division, 101(4), 373-386.

Ashley, D. B. (1980). "Simulation of repetitive-unit Construction." *Journal of Construction Division*, 106(2), 185-194.

Banks, J., Carson, J. S., Nelson, L. N., and Nicol, D. M. (2000). Discrete-Event System Simulation, Prentice Hall.

Carr, R. I. (1979). "Simulation of construction project duration." *Journal of Construction Division*, ASCE, 105(CO2), 117-128.

Carr, R. I., and Meyer, W. L. (1974). "Planning construction of repetitive building units." *Journal of Construction Division*, ASCE, 100(3), 403-412.

Ceric, V. "Hierarchical abilities of diagrammatic representations of discrete event simulation models." *Proc. of the 1994 Winter Simulation Conference*, 589-584.

Chang, D. Y. (1986). "RESCUE: A Resource Based Simulation System for Construction Process Planning," Ph.D. Dissertation, University of Michigan, Ann Arbor, MI.

Cheng, B. (2005). "Limitations of Existing Scheduling Tools in Planning Utility Line Construction Projects," University of Florida, Gainesville.

Chrzanowski, E. N., and Johnston, D. W. (1986). "Application of linear scheduling." *Journal of Construction Engineering and Management*, 112(4), 476-491.

Clough, R. H., and Sears, G. A. (1979). Construction Project Management, John Wiley and Sons, Inc., New York, N.Y.

David, E. D. "Pert and simulation." Proceedings of the 10th conference on Winter simulation, Miami, FL.

El-Rayes, K., and Moselhi, O. (2001). "Optimizing resource utilization for repetitive construction projects." *Journal of Construction Engineering and Management*, 117(3), 18-27.

El-sayegh, S. M. (1998). "Linear Construction Planning Model (LCPM): A New Model for Planning and Scheduling Linear Construction Projects," Texas A & M, College Station, Texas.

Fan, S. L., and Tserng, H. P. (2006). "Object-oriented scheduling for repetitive projects with soft logic." *Journal of Construction Engineering and Management*, 132(1), 35-58.

Fan, S. L., Tserng, H. P., and Wang, M. T. (2003). "Development of an object-oriented scheduling model for construction projects." *Automation in Construction*, 12(3), 283-302.

Fischer, M. A., and Aalami, F. (1996). "Scheduling with computer-interpretable construction method models." *Journal of Construction Engineering and Management-Asce*, 122(4), 337-347.

Flood, I., Issa, R., and Liu, W. (2006)."Rethinking the critical path method for construction project planning." CIB W107 Construction in Developing Economies International Symposium, Santiago, Chile.

Gantt, H. L. (1910). "Work, Wages and Profit." *The Engineering Magazine*, NY.

Goldhaber, S., Jha, C. K., and Macebo, M. C. (1977). Construction Management, John Wiley and Sons, Inc., New York, N.Y.

Gorman, J. E. (1972). "How to get visual impact on planning diagrams." Roads and Streets, Vol. 115(No. 8), 74-75.

Gould, F. E. (2005). Managing the Construction Process, Pearson Pretice Hall, Upper Saddle River, New Jersey.

Handa, V. K., and Barcia, R. M. (1986). "Linear scheduling using optimal control theory." *Journal of Construction Engineering and Management*, 112(3), 387-393.

Harmelink, D. J., and Rowings, J. E. (1998). "Development of controlling activity path." *Journal of Construction Engineering and Management*, ASCE, 124(4), 263-268.

Harris, R. B., and Ioannou, P. G. (1998). "Scheduling projects with repeating activities." *Journal of Construction Engineering and Management*, ASCE, 124(4), 269-278.

Hill, P. R. (1971). *HOCUS*, P.E. Group, Egham, Surrey.

Hinze, J. W. (2008). Construction Planning and Scheduling, Pearson Pretice Hall, Upper Saddle River, New Jersey.

Hooper, J. W. (1986). "Strategy related characteristics of discrte-event languages and models." *Simulation*, 46(4), 153-159.

Huber, P., K., J., and M., S. R. "Hierarchies of coloured petri nets." *Proc. of 10th Interational Conference on Application and Theory of Petri Nets (LNCS 483)*, Springer-Verlag, 313-341.

Hughes, B. D. (2005). "A case study on the Gantt chart scheduling format for high-rise structures in central Florida," University of Florida, Gainesville.

Johnston, D. W. (1981). "Linear scheduling method for highway construction." *Journal of the Construction Division*, ASCE, Vol. 107(No. CO2), 247-259.

Kelton, W. D., Sadowski, R. P., and Sturrock, D. T. (2004). *Simulation with Arena*, McGraw-Hill Companies, Inc., Boston.

Klir, G. J. (1991). *Facets of Systems Science* Springer, New York, New York.

Sawhney, A., and AbouRizk, S. M. (1995). "HSM-Simulation based planning method for construction projects." *Journal of Construction Engineering and Management*, 121(3).

Krahl, D. "Modeling with Extend ." *The 1996 Winter Simulation Conference*, Coranada, CA.

Law, A. M., and Kelton, W. D. (1991). *Simulation Modeling and Analysis*, McGraw-Hill Inc., New York.

Liu, L., Burns, S. A., and Feng, C. (1995). "Construction time-cost trade-off analysis using LP/IP hybrid method." *Journal of Construction Engineering and Management*, 121(4), 446-454.

Liu, L. Y. (1991). "COOPS: Construction Object-Oriented Simulation System," Ph.D. Dissertation, University of Michigan, Ann Arbor, MI.

Lumsden, P. (1968). The Line-of-Balance Method, Pergamon Press, London.

Lutz, J. D., and Halpin, D. W. (1992). "Analyzing linear construction operation using simulation and line-of-balance." *Transportation Research Record*, 1351, 48-56.

Mathewson, S. C. (1974). "Simulation program generator." *Simulation*, 23(6), 181-189.

Martinez, J. C., and Ioannou, P. G. (1999). "General-purpose systems for effective construction simulation." *Journal of Construction Engineering and Management,* ASCE, 125(4), 265-276.

Martinez, J. C. (1996). "State and Resource Based Simulation of Construction Processes," Ph.D. dissertation, University of Michigan.

Martinez, J. C., and Ioannou, P. G. (1995). "Advantages of the activity Scanning Approach in the modeling of complex construction processes." *Proceedings of the 1995 Winter Simulation Conference*, Arlinton, VA.

Martinez, J. C., Ioannou, P. G., and Carr, R. I. "State and resource based construction process simulation." *Proceedings of the First Congress on Computing in Civil Engineering*, ASCE, Washington, DC.

Morua-Padilla, E. (1986). "Resource strategies for dynamic construction management," University of Michigan, Ann Arbor, MI.

O'Brien, W. J., and Fischer, M. A. (2000). "Importance of capacity constraints to construction cost and schedule." *Journal of Construction Engineering and Management*, ASCE, 126(5), 366-373.

Peer, S. (1974). "Network analysis and construction planning." *Journal of Construction Division*, ASCE, Vol. 100(No. CO3), 203-210.

Peer, S., and Selinger, S. "Parameters affecting construction time in housing projects." *Proceedings of the Second International Symposium on Lower-Cost Housing Problems*, University of Missouri, Rolla, 217-220.

Pidd, M. (1998). *Computer Simulation in Management Science*, John Wiley & Sons Ltd., Chichester, England.

Prisker, A. A. B., and O'Reiley, J. J. (1999). *Simulation with Visual SLAM and AweSim*, John Wiley & Sons, Inc, New York.

Prisker, A. A. B. (1977). *Modeling and Analysis using Q-GERT Networks*, Halsted Press, John Wiley & Sons, New York, New York.

Prisker, A. A. B., and O'Reiley, J. J. (1999). Simulation with Visual SLAM and AweSim, John Wiley & Sons, Inc, New York.

Reda, R. M. (1990). "RPM: repetitive project modeling." *Journal of Construction Engineering and Management*, 116(2), 316-330.

Selinger, S. (1980). "Construction planning for linear projects." *Journal of Construction Division*, ASCE, Vol. 106 (No. CO2), 195-205.

Senior, B. A., and Halpin, D. W. (1998). "Simplified simulation system for construction projects." *Journal of Construction Engineering and Management*, ASCE, 124(1), 72-81.

Senouci, A. B., and Eldin, N. N. (1996). "Dynamic programming approach to scheduling of nonserial linear project." *Journal of Computing in Construction*, 10(2), 106-114.

Shi, J. (1999). "Activity-based construction modeling and simulation method." *Journal of Construction Engineering and Management*, ASCE, 125(9), 72-81.

Shi, J., and Abourizk, S. (1998). "Continuous and combined event-process models for simulation pipeline construction." *Construction Management and Economics*, 16, 489-498.

Stradal, O., and Cacha, J. (1982). "Time space scheduling method." *Journal of Construction Division,* ASCE, Vol. 108(No. CO3), 445-457.

Suhail, S. A., and Neale, R. H. (1994). "CPM/LOB: new methodology to integrate CPM and Line of Balance." *Journal of Construction Engineering and Management*, 120(3), 667-684.

Tocher, K. D. "Some Techniques of Model Building." *Proceedings of IBM Scientific Computing Symposium on Simulation Models and Gaming*, New York, 119-155.

Tommelein, I. D., Carr, R. I., and Odeh, A. M. (1994). "Assembly of Simulation Networks Using Designs, Plans, and Methods." *Journal of Construction Engineering and Management,* ASCE, 120(4), 796-815.

Van Slyke, R. M. (1963). "Monte-Carlo Methods and the PERT Problem." *Operational Research*, 11(5), 839-860.

Vorster, M. C., Beliveau, Y. J., and Bafna, T. (1992). "Linear scheduling and visualization." *Transportation Research Record*, 1351, 32-39.

Wang, C., and Huang, Y. (1998). "Controlling activity interval times in LOB scheduling." *Construction Management and Economics*, 16(1), 5-16.

Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of Modeling and Simulation*, Harcourt India Private Limited, New Delhi, India.

Zeigler, B. P. (1984). *Multifaceted Modeling and Discrete Event Simulation*, Academic Press, London.

Zeigler, B. P. (1976). *Theory of Modeling and Simulation*, Wiley Inter-science.

Zhang, H., Shi, J., and Tam, C. M. (2002). "Iconic Animation for Activity-based Construction Simulation." *Journal of Computing in Civil Engineering*, 16(3), 157-164.

BIOGRAPHICAL SKETCH

Wen Liu earned her bachelor's and master's degrees in construction project management at the Tianjin University in China. She attended the M. E. Rinker, Sr. School of Building Construction at the University of Florida to obtain her Doctor of Philosophy in the field of building construction, which will be awarded in May 2008. She also holds a master degree in decision and information sciences from the Warrington College of Business Administration at the University of Florida.