Neural Network Computing Model for Highway Construction Project Scheduling and Management

by

Hojjat Adeli
Professor
The Ohio State University

Sponsored by
Ohio Department of Transportation
and
Federal Highway Administration

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA/OH-99/010		

4. Title and Subtitle	5. Report Date
Neural Network Computing Model for Highway Construction Project Scheduling and Management	October, 1999
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Hojjat Adeli	
	10. Work Unit No. (TRAIS)
9. Performing Organization Name and Address	11. Contract or Grant No.
The Ohio State University Department of Civil Engineering Columbus, OH 43215	State Job No. 14634(0)
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Final Report
Ohio Department of Transportation 1600 West Broad Street Columbus, OH 43223	14. Sponsoring Agency Code

15. Supplementary Notes

Prepared in cooperation with the U.S. Department of Transportation, Federal Highway Administration

16. Abstract

A general mathematical formulation has been developed for scheduling of construction projects and applied to the problem of highway construction scheduling. Repetitive and non-repetitive tasks, work continuity considerations, multiple-crew strategies, and the effects of varying job conditions on the performance of a crew can be modeled. An optimization formulation is presented for the construction project scheduling problem with the goal of minimizing the direct construction cost. The nonlinear optimization problem is then solved by the recently patented neural dynamics model of Adeli and Park (U.S. patent No. 5,815,394 issued on September 29, 1998). For any given construction duration, the model yields the optimum construction schedule for minimum construction cost automatically. By varying the construction duration, one can solve the cost-duration trade-off problem and obtain the global optimum schedule and the corresponding minimum construction cost. The new construction scheduling model provides the capabilities of both the CPM and LSM approaches. In addition, it provides features desirable for repetitive projects such as highway construction and allows schedulers greater flexibility. It is particularly suitable for studying the effects of change order on the construction cost. An object-oriented (OO) information model has been developed for construction scheduling, cost optimization, and change order management based on the new construction scheduling model. The OO model has been implemented in a prototype software system for management of construction projects, called CONSCOM, in Visual C++. CONSCOM is particularly suitable for highway construction change management. It can be used by the owner as an intelligent decision support system in schedule reviews, progress monitoring, and cost-time trade-off analysis for change order approval.

17. Key Words	18. Distribution Statement
change order management, construction scheduling, cost estimation, highway construction, neural dynamics model of Adeli and Park, neural network computing, optimization	No Restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Neural Network Computing Model for Highway Construction Project Scheduling and Management

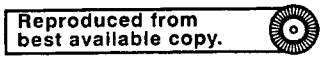
Principal Investigator: Hojjat Adeli, Professor, The Ohio State University

Executive Summary

A general mathematical formulation has been developed for scheduling of construction projects and applied to the problem of highway construction scheduling. Repetitive and non-repetitive tasks, work continuity considerations, multiple-crew strategies, and the effects of varying job conditions on the performance of a crew can be modeled. An optimization formulation is presented for the construction project scheduling problem with the goal of minimizing the direct construction cost. The nonlinear optimization problem is then solved by the recently patented neural dynamics model of Adeli and Park (United States patent number 5,815,394 issued on September 29, 1998). For any given construction duration, the model yields the optimum construction schedule for minimum construction cost automatically. By varying the construction duration, one can solve the cost-duration trade-off problem and obtain the global optimum schedule and the corresponding minimum construction cost. The new construction scheduling model provides the capabilities of both the CPM and LSM approaches. In addition, it provides features desirable for repetitive projects such as highway construction and allows schedulers greater flexibility. It is particularly suitable for studying the effects of change order on the construction cost. This research provides the mathematical foundation for development of a new generation of more general, flexible, and accurate construction scheduling systems.

Estimating the cost of a construction project is an important task in the management of construction projects. Quality of the construction management depends on the accurate estimation of the construction cost. Highway construction costs are very noisy and the noise is the result of many unpredictable factors. A regularization neural network is formulated and a neural network architecture is presented for estimating the cost of construction projects. The model is applied to estimate the cost of reinforced concrete pavements as an example. The new computational model is based on a solid mathematical foundation making the cost estimation consistently more reliable and predictable. Moreover, the problem of noise in the data is taken into account in a rational manner.

An object-oriented (OO) information model has been developed for construction scheduling, cost optimization, and change order management based on the new construction scheduling model developed in this work. The goal is to lay the foundation for a new generation of flexible, powerful, maintainable, and reusable software system for the construction scheduling problem. The model is presented as a domain-specific development *framework* using the Microsoft Foundation Class (MFC) library and utilizing the software reuse feature of the *framework*. The OO model has been implemented in a prototype software system for management of construction projects, called CONSCOM, in Visual C++. CONSCOM is particularly suitable for highway construction change management. It can be used by the owner as an intelligent decision support system in schedule reviews, progress monitoring, and cost-time trade-off analysis for change order approval. CONSCOM includes an integrated management environment (IME) as its user interface for effective control and management of construction projects.



Neural Network Computing Model for Highway Construction Project Scheduling and Management

Principal Investigator

Hojjat Adeli
Professor
The Ohio State University

October 1999

Sponsored by
Ohio Department of Transportation
and
Federal Highway Administration

Prepared in Cooperation with the Ohio Department of Transportation and the U.S. Department of Transportation, Federal Highway Administration

Summary and Organization of the Report

This report consists of five parts presented as five different manuscripts. In the first manuscript a general mathematical formulation is presented for scheduling of construction projects and applied to the problem of highway construction scheduling. Repetitive and non-repetitive tasks, work continuity considerations, multiple-crew strategies, and the effects of varying job conditions on the performance of a crew can be modeled. An optimization formulation is presented for the construction project scheduling problem with the goal of minimizing the direct construction cost. The nonlinear optimization problem is then solved by the neural dynamics model recently patented by Adeli and Park (United States patent number 5,815,394 issued on September 29, 1998). For any given construction duration, the model yields the optimum construction schedule for minimum construction cost automatically. By varying the construction duration, one can solve the cost-duration trade-off problem and obtain the global optimum schedule and the corresponding minimum construction cost. The new construction scheduling model provides the capabilities of both the CPM and LSM approaches. In addition, it provides features desirable for repetitive projects such as highway construction and allows schedulers greater flexibility. It is particularly suitable for studying the effects of change order on the construction cost. This research provides the mathematical foundation for development of a new generation of more general, flexible, and accurate construction scheduling systems.

Estimating the cost of a construction project is an important task in the management of construction projects. Quality of the construction management depends on the accurate estimation of the construction cost. Highway construction costs are very noisy and the noise is the result of many unpredictable factors. In the second manuscript, a regularization neural network is formulated and a neural network architecture is presented for estimating the cost of construction projects. The model is applied to estimate the cost of reinforced concrete pavements as an example. The new computational model is based on a solid mathematical foundation making the cost estimation consistently more reliable and predictable. Moreover, the problem of noise in the data is taken into account in a rational manner.

In the third manuscript an object-oriented (OO) information model is presented for construction scheduling, cost optimization, and change order management based on the new construction scheduling model described in the first manuscript. The goal is to lay the foundation for a new generation of flexible, powerful, maintainable, and reusable software system for the construction scheduling problem. The model is presented as a domain-specific development *framework* using the Microsoft Foundation Class (MFC) library and utilizing the software reuse feature of the *framework*. The *framework* reuse architecture is more flexible and powerful than other software reuse techniques such as *components* and *patterns*.

In the fourth manuscript the OO model has been implemented in a prototype software system for management of construction projects, called CONSCOM, in Visual C++. CONSCOM is particularly suitable for highway construction change management. It can be used by the owner as an intelligent decision support system in schedule reviews, progress monitoring, and cost-time trade-off analysis for change order approval. Finally, the last manuscript presents the recent and innovative capabilities and features of CONSCOM including an integrated management environment (IME) as its user interface for effective control and management of construction projects. A highway construction project is presented to demonstrate the unique modeling capabilities of CONSCOM that cannot be modeled by CPM or CPM-like networks.

SCHEDULING/COST OPTIMIZATION AND NEURAL DYNAMICS MODEL FOR CONSTRUCTION PROJECTS

Hojjat Adeli[1] and Asim Karim[2]

Abstract: A general mathematical formulation is presented for scheduling of construction projects and applied to the problem of highway construction scheduling. Repetitive and non-repetitive tasks, work continuity considerations, multiple-crew strategies, and the effects of varying job conditions on the performance of a crew can be modeled. An optimization formulation is presented for the construction project scheduling problem with the goal of minimizing the direct construction cost. The nonlinear optimization is then solved by the neural dynamics model developed recently by Adeli and Park. For any given construction duration, the model yields the optimum construction schedule for minimum construction cost automatically. By varying the construction duration, one can solve the cost-duration trade-off problem and obtain the global optimum schedule and the corresponding minimum construction cost. The new construction scheduling model provides the capabilities of both the CPM and LSM approaches. In addition, it provides features desirable for repetitive projects such as highway construction and allows schedulers greater flexibility. It is particularly suitable for studying the effects of change order on the construction cost. This research provides the mathematical foundation for development of a new generation of more general, flexible, and accurate construction scheduling systems.

[1]Professor, Dept. of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 470 Hitchcock Hall, 2070 Neil Avenue, Columbus, Ohio 43210
[2]Graduate Research Associate, Dept. of Civil and Environmental Engineering and Geodetic Science, The Ohio State University

INTRODUCTION

Most construction projects involve a combination of repetitive and non-repetitive tasks. A typical example is highway construction in which tasks such as clearing and grubbing are performed repeatedly over the length of the highway, and tasks such as site office construction are carried out only once. Presently, traditional network scheduling methods such as CPM and PERT are used for the scheduling and monitoring of such projects. Despite their extensive use these methods have a number of shortcomings:

- Network methods do not guarantee continuity of work in time which may result in crews being idle.
- Multiple-crew strategies are difficult to implement in the network methods.
- The network diagram is not suitable for monitoring the progress of a project.
- Network methods do not provide an efficient structure for the representation of repetitive tasks. All tasks are represented similarly and there is no consideration of the location of work in the scheduling.

To overcome these shortcomings new approaches have been proposed in the literature particularly for repetitive projects. Figure 1 presents a linear planning chart which is a graph of location (distance) versus time for the work to be carried out. Such a planning chart represents the progress of a task and can be used to monitor a project. The linear planning chart (also called LSM diagram) motivated the development of linear scheduling method (LSM). Selinger (1980) presented equations for the lines in a linear planning chart assuming non-interference of crews and continuity of work. Johnston (1981) showed the LSM is flexible and can be used to model most situations encountered in highway

construction projects. Using optimal control theory, Handa and Barcia (1986) formulated the problem as an optimization one minimizing the project duration. These early LSM models had limitations such as constant rate of production for each task, binding continuity constraints, and no provisions for the use of multiple crews.

Russell and Caselton (1988) presented a dynamic programming formulation to minimize the project duration. Their formulation can accommodate variable production rates for each task and non-binding work continuity constraints. Russell and Wong (1993) described and showed the use of a general scheduling model developed by incorporating the capabilities of CPM and LSM. In their model, each task is defined by a set of attributes which are then linked together using general precedence conditions to form a schedule.

Highway construction projects are large projects in terms of capital requirement. Minimizing cost is therefore a primary goal in the planning and scheduling of such projects. Cost, however, is closely related to time. In general, direct project cost increases with a decrease in project duration and this trade-off problem is complicated by the number of variables involved. A computer model to automate the process of project direct cost minimization is therefore highly desirable.

In the recent literature, direct cost optimization systems have been presented for LSM and CPM. Reda's (1990) LSM model assumes constant production rate for each task and binding constraints on work continuity. The cost-duration relationship for each task is assumed to be linear. Liang et al. (1995) present a hybrid linear/integer programming approach for handling a combination of discrete and linearly continuous cost-duration relationship for tasks.

In this article, a general mathematical formulation is presented for scheduling of construction projects. Various scheduling constraints are expressed mathematically. The construction scheduling is posed as an optimization problem where project direct cost is minimized for a given project duration assuming any combination of linear and nonlinear task cost-duration relationships. The robust neural dynamics model developed recently by Adeli and Park (1995a) is adapted for optimization.

COST-DURATION RELATIONSHIP OF A PROJECT

The major cost of a project consists of direct and indirect costs. The resources allocated to each task of a project determine the direct cost. Indirect costs are overhead costs. Additional costs may be incurred by the contractor in the form of damages if the project is not completed on time. The duration of a project is obtained by sequencing individual tasks whose durations are estimated from a knowledge of the resources allocated to each task and the job conditions. Thus, cost and duration are intricately related. Both of these parameters are of great importance to the contractor who strives to minimize cost while at the same time satisfying the contractual requirements, the most important of which is the completion deadline. Figure 2 shows the typical variation of direct, indirect and total costs of a construction project with the project duration. Assuming the sequencing constraints are not changed, the direct cost, in general, has an inverse relationship with the duration of a construction project. The indirect cost increases with an increase in the duration of the project. The total cost is the sum of these two and

can increase or decrease with duration. By solving the direct cost optimization problem for various durations the global optimum solution can be obtained from Figure 2.

There is also an inverse relationship between direct cost and the duration of an individual task. A scheduler estimates the time required to complete a task from the resources allocated to it. This time is based on assumed labor and equipment productivity rates ignoring the effects of varying job conditions. Depending on the options and the availability of resources the scheduler has for each task, a cost-duration curve can be constructed. This curve can be continuous or discrete. For efficient mathematical formulation the discrete relationship is approximated by a continuous linear (Figure 3a) or nonlinear (Figure 3b) curve. In this way a continuous variable optimization technique can be used to solve the construction time-cost trade-off problem.

In highway construction, it is convenient to represent cost and duration of a task in unit quantities of work. If d_i is the time required to complete a unit quantity of work of task i and W_{ij} is the total quantity of work required in segment j of task i, then the actual duration, D_{ij}, can be expressed as:

$$D_{ij} = \mu_{ij} d_i W_{ij} \tag{1}$$

where μ_{ij} is the job condition factor reflecting the effects of variable conditions such as weather, soil conditions, terrain, site congestion, learning effects, etc.

FORMULATION OF THE SCHEDULING OPTIMIZATION PROBLEM

A general mathematical formulation of the scheduling problem is presented in this section. The advantage of such a general formulation is that it can be specialized and

reduced for the solution of specific and perhaps less complicated scheduling problems. Further, it can be effectively integrated with the general neural dynamics model for solution of optimization problems developed by Adeli and Park (1995a).

Both non-repetitive and repetitive tasks are considered in the formulation. The non-repetitive tasks correspond to the activities of the traditional network methods such as CPM. A non-repetitive task involves no internal logic as it is performed only once. A repetitive task, on the other hand, may have an elaborate internal logic that connects the segments assigned to various crews. By specifying appropriate constraints, work continuity considerations and multiple-crew strategies can be modeled. A crew rarely performs at ideal productivity throughout; its performance is affected by the varying job conditions. This is included in our scheduling model by means of a factor, μ_{ij}, which modifies the ideal productivity of a crew to reflect the effect of the job conditions. The external logic of each task is specified by means of a full set of precedence relationships and/or stage (distance) and time buffers.

Development of the general scheduling formulation for a construction project such as highway construction involves the following steps divided into three main categories (headings) as follows:

Breakdown the work into tasks, crews, and segments

Step 1

Break down the project into N_T tasks. Identify non-repetitive and repetitive tasks. Let N_{NT} and N_{RT} be the number of non-repetitive and repetitive tasks, respectively. If $N_{RT} = 0$, skip steps 2 to 5 and go to step 6.

Step 2

For each repetitive task i, choose the number of crews to be used (N_{Ci}). Non-repetitive tasks have only one crew that performs over one segment only.

Step 3

Assign N_{Si}^{k} segments of the highway to crew k of repetitive task i. The segments are chosen considering the job conditions and quantity of work required, factors affecting the production rate. In addition, predetermined breaks in the work of a crew may influence the choice of segments. The segments are not required to have equal lengths or constructed in sequence. Each segment is identified by Z_{ij}^{k} and $Z_{ij}^{k'}$, the beginning and ending distances at which repetitive task i is performed by crew k over segment j. Note that each crew of a task is assigned a unique set of segments; two crews cannot perform the same task over the same portion of the highway.

Specify the internal logic of repetitive tasks

For each crew of a repetitive task, do the following:

Step 4

Specify the work continuity relationship between segments j and $j+1$, in the following form:

$$T_{ij}^{k} + D_{ij}^{k} + S_{ij}^{k} \leq T_{i(j+1)}^{k} \tag{2}$$

where T_{ij}^{k} is the time at which crew k of task i starts work on segment j, D_{ij}^{k} is the duration of work for crew k of task i on segment j, and S_{ij}^{k} is the idle or slack time of crew k of task i between segments j and $j+1$. For continuity of work, S_{ij}^{k} must be equal to zero. If a task has only one crew skip step 5 and go to step 6.

Step 5

Define the start of a crew with respect to previous crew(s). The following precedence relationships of start-to-start, finish-to-finish, and start-to-finish are used:

Start-to-start (SS):

$$T_{i1}^{k} + L_{SSi}^{kl} \leq T_{i1}^{l} \tag{3}$$

Finish-to-finish (FF):

$$T_{iN_{Si}^{k}}^{k} + D_{iN_{Si}^{k}}^{k} + L_{FFi}^{kl} \leq T_{iN_{Si}^{l}}^{l} + D_{iN_{Si}^{l}}^{l} \tag{4}$$

Start-to-finish (SF):

$$T_{i1}^{k} + L_{SFi}^{kl} \leq T_{iN_{Si}^{l}}^{l} + D_{iN_{Si}^{l}}^{l} \tag{5}$$

where the superscripts l and k refer to the current and the previous crews, respectively; L_{SSi}^{kl}, L_{FFi}^{kl}, L_{SFi}^{kl} are the start-to-start, finish-to-finish, and start-to-finish time lags between crews k and l, respectively. These time lags may be given as a function of quantity of work and/or time. If more than one relationship is specified for a particular crew, only one will govern in the final minimum cost schedule obtained from the optimization algorithm. This particular relationship

usually is not known in advance and all possible relationships have to be specified in the optimization model.

Specify the external logic of repetitive and non-repetitive tasks

Step 6

Describe the sequencing of the tasks in the project. Each task can be linked with any number of previous tasks by specifying one or more of the following precedence relationships:

Start-to-start (SS):

$$T_{i1}^{1} + L_{SSij} \le T_{j1}^{1} \tag{6}$$

Finish-to-start (FS):

$$T_{iN_{Si}^{k}}^{k} + D_{iN_{Si}^{k}}^{k} + L_{FSij} \le T_{j1}^{1} \qquad k = 1,\ldots,N_{Ci} \tag{7}$$

Start-to-finish (SF) (is specified when the task has only one crew):

$$T_{i}^{1} + L_{SFij} \le T_{jN_{Sj}^{l}}^{l} + D_{jN_{Sj}^{l}}^{l} \qquad l = 1 \tag{8}$$

Finish-to-finish (FF) (is specified when both tasks have one crew only)

$$T_{iN_{Si}^{k}}^{k} + D_{iN_{Si}^{k}}^{k} + L_{FFij} \le T_{jN_{Sj}^{l}}^{l} + D_{jN_{Sj}^{l}}^{l}; \qquad k = l = 1 \tag{9}$$

The quantities L_{SSij}, L_{FSij}, L_{SFij} and L_{FFij} are the respective time lags between task j and a previous task i. The FS relationship can be used to ensure continuity from one task to another by specifying $L_{FSij} = 0$. The relationships represented by Eqs. (6)-(9) can also be written for any given crew or segment of a task rather than the whole task. For example, consider the case where crew B of task Y is the same as crew A of a previous task X. Crew B can start work only

after crew *A* has finished. Therefore, an FS relationship has to be specified between crew *A* of task *X* and crew *B* of task *Y*.

Step 7

Define the space and/or time buffer between tasks. These constraints are essential if interference of crews on different tasks is to be prevented. If task *i* precedes task *j* by a distance buffer B_{Sij}, the following constraints have to be satisfied:

$$Z_j(T_{in}^k) + B_{Sij} \le Z_{in}^k \qquad k = 1,\ldots,N_{Ci}, n = 1,\ldots,N_{Si}^k \tag{10}$$

$$Z_j(T_{in}^k + D_{in}^k) + B_{Sij} \le Z_{in}^{k\prime} \qquad k = 1,\ldots,N_{Ci}, n = 1,\ldots,N_{Si}^k \tag{11}$$

$$Z_{jn}^k + B_{Sij} \le Z_i(T_{jn}^k) \qquad k = 1,\ldots,N_{Cj}, n = 1,\ldots,N_{Sj}^k \tag{12}$$

$$Z_{jn}^k + B_{Sij} \le Z_i(T_{jn}^k + D_{jn}^k)' \qquad k = 1,\ldots,N_{Cj}, n = 1,\ldots,N_{Sj}^k \tag{13}$$

The term $Z_i(T_{jn}^k)$ denotes the location of task *i* at the time T_{jn}^k. For tasks with a constant production rate during a segment of work, $Z_i(T_{jn}^k)$ is found by a linear interpolation between the values at the start (Z_{im}^l) and the finish ($Z_{im}^{l\prime}$) of segment *m* performed by crew *l* of task *i*.

$$Z_i(T_{jn}^k) = Z_{im}^l + \frac{(T_{jn}^k - T_{im}^l)(Z_{im}^{l\prime} - Z_{im}^l)}{(T_{im}^l + D_{im}^l) - T_{im}^l} \tag{14}$$

Similarly, if task *i* precedes task *j* by the time buffer B_{Tij}, then we have the following constraints:

$$T_{in}^k + B_{Tij} \le T_j(Z_{in}^k) \qquad k = 1,\ldots,N_{Ci}, n = 1,\ldots,N_{Si}^k \tag{15}$$

$$(T_{in}^k + D_{in}^k) + B_{Tij} \le T_j(Z_{in}^{k\prime}) \qquad k = 1,\ldots,N_{Ci}, n = 1,\ldots,N_{Si}^k \tag{16}$$

$$T_i(Z_{jn}^k) + B_{Tij} \le T_{jn}^k \qquad k = 1,\ldots,N_{Cj}, n = 1,\ldots,N_{Sj}^k \tag{17}$$

$$T_i(Z_{jn}^{k\prime}) + B_{Tij} \le (T_{jn}^k + D_{jn}^k) \qquad k = 1,\ldots,N_{Cj}, n = 1,\ldots,N_{Sj}^k \tag{18}$$

Likewise, $T_i(Z_{jn}^k)$ is found by a linear interpolation between the starting time (T_{im}^l) and the stopping time ($T_{im}^l + D_{im}^l$) for segment m performed by crew l of task i.

$$T_i(Z_{jn}^k) = T_{im}^l + \frac{(Z_{jn}^k - Z_{im}^l)\{(T_{im}^l + D_{im}^l) - T_{im}^l\}}{(Z_{im}^{l\prime} - Z_{im}^l)} \tag{19}$$

The optimization problem can now be formulated as the minimization of direct cost

$$C_D = \sum_{i=1}^{N_{NT}} W_i C_i(d_i) + \sum_{i=1}^{N_{RT}} \sum_{k-1}^{N_{Ci}} \sum_{j=1}^{N_{Si}^k} W_{ij}^k C_i(d_i^k), \tag{20}$$

subject to the scheduling constraints (Eqs. (2)-(13) and (15)-(18)), plus initial constraint

$$T_{11}^1 = \text{const}, \tag{21}$$

project duration constraints

$$T_{ij}^k + D_{ij}^k \le D^{\max} \qquad i = 1,\ldots,N_T, k = 1,\ldots,N_{Ci}, j = 1,\ldots,N_{Si}^k, \tag{22}$$

task duration constraints

$$(d_i^k)^{\min} \le d_i^k \le (d_i^k)^{\max} \qquad i = 1,\ldots,N_T, k = 1,\ldots,N_{Ci}, \tag{23}$$

and, non-negativity constraints

$$T_{ij}^k, d_i^k \ge 0 \qquad i = 1,\ldots,N_T, k = 1,\ldots,N_{Ci}, j = 1,\ldots,N_{Si}^k, \tag{24}$$

where C_i is the direct cost per unit quantity of work for task i; d_i^k is the time required by crew k of task i to complete a unit quantity of work based on resource allocation only; $(d_i^k)^{\min}$, $(d_i^k)^{\max}$ are the minimum and maximum possible values of d_i^k, respectively; and $D^{\max}$ is the

maximum acceptable project duration. Note that in this formulation, Eq. (1) can be written for each crew k of task i as:

$$D_{ij}^{k} = \mu_{ij}^{k} d_i^{k} W_{ij}^{k} \tag{25}$$

ARTIFICIAL NEURAL NETWORKS AND SCHEDULING

Artificial neural networks (ANN) are a functional abstraction of the biological neural structures of the central nervous system. Their computing abilities have been proven in the fields of prediction and estimation, pattern recognition, and optimization (Adeli and Yeh, 1989; Adeli and Zhang, 1993; Adeli and Hung, 1995; Adeli and Park, 1995a, b, c; Adeli and Park, 1996a). The use of ANN for solving scheduling problems has been reported in the recent literature; however, practically all work is in the area of job-shop scheduling (Gulati et al., 1987; Watanabe et al., 1993; Willems and Rooda, 1994; Pellerin and Herault, 1994; Foo et al., 1995). Job-shop scheduling is a resource allocation problem in which n jobs have to be scheduled on m machines (the resources) given their operation pattern. The performance criterion is usually the minimization of work completion time. The aforementioned papers pose the problem as an optimization problem and use the Hopfield network (Hopfield and Tank, 1985), or its variations, to solve the problem. The job-shop scheduling problem has been formulated as linear programming (Chang and Nam, 1993), integer programming (Willems and Rooda, 1994) and mixed integer programming (Foo et al., 1995). The ANN models presented in these papers are specific to the particular problem considered. Also, it should be mentioned that the job-shop scheduling problem is an NP-complete problem which requires exhaustive

enumeration for solution. Construction scheduling problems, on the other hand, should be formulated as a constrained nonlinear mathematical programming problem.

Recently, Adeli and Park (1995a) developed a nonlinear neural dynamics model as a new optimization technique for solution of complex optimization problems by integrating penalty function method. Lyapunov stability theorem, Kuhn-Tucker conditions, and the neural dynamics concept. The Lyapunov stability theorem guarantees that solutions of the corresponding dynamic system (trajectories) for arbitrarily given starting points approach an equilibrium point without increasing the value of the objective function. This guarantees global convergence and robustness. But, it does not guarantee the equilibrium point is a local minimum. The Kuhn-Tucker conditions are used to verify that the equilibrium point satisfies the necessary conditions for a local minimum. The robustness of the model was demonstrated by application to both linear (Park and Adeli, 1995) and nonlinear (Adeli and Park, 1995b) structural optimization problems. Most recently, the model was applied to optimization of very large structures including a 144-story super-high-rise building structure with over 20,000 members subjected to actual design specifications (Adeli and Park, 1996b).

An ANN model for the complete scheduling of construction projects has not been presented in the literature. Alsugair and Chang (1994) used a backpropagation learning network to capture human knowledge of allocating construction resources. The ANN determines the size and number of equipment units required for earthmoving processes. Mohammed et al. (1995) formulated the problem of optimally allocating available yearly budget to bridge rehabilitation and replacement projects among a number of alternatives as an optimization problem using the Hopfield network.

NEURAL DYNAMICS COST OPTIMIZATION MODEL FOR CONSTRUCTION PROJECTS

Formulation

Defining $\mathbf{X} = \{T_{ij}^k, d_i^k | i = 1, N_T, k = 1, N_{Ci}, j = 1, N_{Si}^k\}$ as the vector of decision variables, the optimization problem can be written as:

Minimize

$$C_D = f(\mathbf{X}) \tag{26}$$

subject to inequality constraints

$$g_j(\mathbf{X}) \le 0 \qquad j = 1,\ldots,J \tag{27}$$

and equality constraints

$$h_k(\mathbf{X}) = 0 \qquad k = 1,\ldots,K \tag{28}$$

where $g_j(\mathbf{X})$ is the jth inequality constraint function, $h_k(\mathbf{X})$ is the kth equality constraint function, J is the total number of inequality constraints, and K is the total number of equality constraints. Using the exterior penalty function method, a pseudo-objective function is defined as:

$$P(\mathbf{X}, r_n) = f(\mathbf{X}) + \frac{r_n}{2}\left\{\sum_{j=1}^{J}\left[g_j^+(\mathbf{X})\right]^2 + \sum_{k=1}^{K}\left[h_k(\mathbf{X})\right]^2\right\} \tag{29}$$

where $g_j^+(\mathbf{X}) = \max\{0, g_j(\mathbf{X})\}$ and r_n is a penalty parameter magnifying constraint violations.

A dynamic system is defined as:

$$\frac{d\mathbf{X}}{dt} = \dot{\mathbf{X}} = F(\mathbf{X}) \tag{30}$$

where $\mathbf{X} = \{X_1(t), X_2(t), \ldots, X_N(t)\}^T$ is the state vector tracing a trajectory in N-dimensional space where the superscript T indicates the transpose of a vector and $N = \sum_{i=1}^{N_T} \sum_{k=1}^{N_{Ci}} N_{Si}^k + \sum_{i=1}^{N_T} N_{Ci}$. The dynamic system evolves until it reaches an equilibrium point. The stability of such an equilibrium point is ensured by satisfying the Lyapunov stability theorem which states that a solution $\hat{\mathbf{X}}$ to the system of differential equations $\dot{\mathbf{X}} = 0$ is stable if

$$\frac{dV}{dt} \leq 0 \qquad \text{for all non-zero } \mathbf{X} \tag{31}$$

where $V(\mathbf{X})$ is the Lyapunov functional defined as an analytic function of the state variables such that $V(\mathbf{0}) = 0$ and $V(\mathbf{X}) > 0$ for all $|\mathbf{X}| > 0$ (Kolk and Lerman, 1992). The objective (direct cost) function and the constraint functions in our construction cost optimization model individually satisfy the conditions for a Lyapunov functional. Therefore, the pseudo-objective function P defined by Eq. (29) is also a valid Lyapunov functional, V.

Following Adeli and Park (1995a), by defining

$$\frac{d\mathbf{X}}{dt} = \dot{\mathbf{X}} = -\nabla f(\mathbf{X}) - r_n \left\{ \sum_{j=1}^{J} g_j^+ \nabla g_j(\mathbf{X}) + \sum_{k=1}^{K} h_k \nabla h_k(\mathbf{X}) \right\} \tag{32}$$

where $\nabla f(\mathbf{X})$, $\nabla g_j(\mathbf{X})$, and $\nabla h_k(\mathbf{X})$ are the gradients of the objective function, the jth inequality constraint, and kth equality constraint, respectively, the Lyapunov stability theorem for the dynamic system is satisfied.

$$\frac{dV}{dt} = \left(\frac{dV}{d\mathbf{X}}\right)\left(\frac{d\mathbf{X}}{dt}\right) = -\left[\nabla f(\mathbf{X}) + r_n\left\{\sum_{j=1}^{J} g_j^+ \nabla g_j(\mathbf{X}) + \sum_{k=1}^{K} h_k \nabla h_k(\mathbf{X})\right\}\right]^2 \le 0 \qquad (33)$$

This also shows that the dynamic system evolves such that the value of the pseudo-objective function always decreases. Equation (32) is in fact the learning rule of the neural dynamics model.

For an equilibrium point $\mathbf{X}$ to be a local optimum solution, we also need to satisfy the Kuhn-Tucker optimality conditions:

$$\frac{\partial L}{\partial X_i} = \frac{\partial f(\mathbf{X})}{\partial X_i} + \sum_{j=1}^{J} u_j \frac{\partial g_j(\mathbf{X})}{\partial X_i} + \sum_{k=1}^{K} v_k \frac{\partial h_k(\mathbf{X})}{\partial X_i} = 0;\ i = 1,\ldots,N \qquad (34)$$

$$g_j(\mathbf{X}) + s_j^2 = 0; \qquad j = 1,\ldots,J \qquad (35)$$

$$h_k(\mathbf{X}) = 0; \qquad k = 1,\ldots,K \qquad (36)$$

$$u_j s_j = 0; \qquad j = 1,\ldots,J \qquad (37)$$

$$u_j \ge 0; \qquad j = 1,\ldots,J \qquad (38)$$

$$v_k = \text{unrestricted in sign} \qquad (39)$$

where L is the Lagrangian function defined as a linear combination of the objective and constraint functions:

$$L(\mathbf{X},\mathbf{u},\mathbf{v},\mathbf{s}) = f(\mathbf{X}) + \sum_{j=1}^{J} u_j\left[g_j(\mathbf{X}) + s_j^2\right] + \sum_{k=1}^{K} v_k h_k(\mathbf{X}), \qquad (40)$$

in which s_j is the slack term for the jth inequality constraint, and u_j and v_k are the Lagrangian multipliers corresponding to the jth inequality and kth equality constraint, respectively.

Finally, the optimum solution to the direct cost optimization problem can be found by the integration:

$$\mathbf{X} = \int \dot{\mathbf{X}} dt . \tag{41}$$

This integration can be performed by the Euler or Runge-Kutta method.

Topological characteristics

The neural network topology for the neural dynamics construction cost optimization model is shown in Figure 4. The nodes in the network represent the variables and constraints of the problem. The variable layer has N nodes corresponding to the total number of decision variables. The constraint nodes are divided into N_{NT} layers corresponding to non-repetitive tasks, N_{RT} layers corresponding to repetitive tasks, and an initial constraint node. Nodes are grouped within each layer into the constraint categories described in a previous section. Variable and constraint nodes are fully interconnected (interlayer connections). In addition, recurrent and intra-layer connections are also used to be described shortly.

Associated with each connection is a weight whose magnitude and sign affect the impulse the connected node will receive. Both excitatory (positive connection weights) and inhibitory (negative connection weights) connections are used in our model. The coefficients of the constraint functions are assigned to the excitatory connections from the variable layer to the constraint nodes. The gradients of the constraint functions are assigned to the inhibitory connections from the constraint nodes to the variable layer. The gradients of the objective function are assigned to the recurrent inhibitory connections of

the variable layer. A weight of one is assigned to the intra-layer connections. This allows the outputs of nodes in competition to be compared. It should be noted that all the connection weights are fixed in magnitude and sign for a given problem.

The output of the variable layer is the current state vector $\mathbf{X}$. As the coefficients of the constraint functions are encoded in the excitatory connections from the variable layer to the constraint nodes, the input to a constraint node is the magnitude of the constraint at any given state, that is, $g_j(\mathbf{X})$ for an inequality constraint j, and $h_k(\mathbf{X})$ for an equality constraint k. The output of a constraint node will depend on the type of the constraint it represents. For an inequality constraint j, the output is:

$$O_{cj} = \begin{cases} 0 & \text{when } g_j(\mathbf{X}) \le 0 \\ r_n g_j(\mathbf{X}) & \text{when } g_j(\mathbf{X}) > 0 \end{cases} \tag{42}$$

and for an equality constraint k the output is:

$$O_{ck} = \begin{cases} 0 & \text{when } h_k(\mathbf{X}) = 0 \\ r_n h_k(\mathbf{X}) & \text{when } h_k(\mathbf{X}) \neq 0 \end{cases} \tag{43}$$

Equations (42) and (43) represent the activation functions. They are chosen such that the output of a constraint node is the penalized constraint violation. When more than one equation is specified for a particular category of constraint, such as external logic constraint, a competition is created between the outputs of the nodes in that group. For a group of n nodes with outputs $O_{c1}, O_{c2}, \ldots, O_{cj}, \ldots, O_{cn}$ such that

$$O_{cj} = \max\{O_{c1}, O_{c2}, \ldots, O_{cj}, \ldots, O_{cn}\} \tag{44}$$

The outputs after competition are as follows:

$$O_{cj} = O_{cj} \text{ and} \tag{45}$$

$$O_{c1}, O_{c2}, \ldots, O_{cn} = 0 \tag{46}$$

Let w_{ji} and w_{ki} be the connection weight from the jth and kth inequality and equality constraint node, respectively, to the ith variable node and Y_i be the weight of the recurrent connection to a node i in the variable layer. Then, the input to the ith variable node is given by:

$$I_{vi} = Y_i + \sum_{j=1}^{J} w_{ji} O_{cj} + \sum_{k=1}^{K} w_{ki} O_{ck} \tag{47}$$

The new value of the ith decision variable is obtained by the integration:

$$X_i^{new} = \int I_{vi} dt \tag{48}$$

This integration is done by the Euler or the Runge-Kutta methods. In the construction cost optimization problem we found the simple Euler method to yield accurate results.

The network operates until no change in the decision variables occur within a given tolerance, that is, when $\dot{\mathbf{X}} = 0$. $\mathbf{X}$ is the solution to the minimum direct cost construction scheduling problem.

ILLUSTRATIVE EXAMPLE

A 5 km-long two-lane highway construction project is used to illustrate the capabilities of the computational model presented in this article. The work required is divided into 14 repetitive and non-repetitive tasks summarized in Table 1. Tasks 1 to 5 represent the establishment of a temporary site office at the beginning of the 5 km long stretch. The erection of an asphalt concrete plant at a distance of 2.5 km from the beginning of the

roadway (at the center of the project) is represented by tasks 6 and 7 (together with portion of task 9).

Cost-duration relationship

The relationship between direct cost and duration for unit quantity of work for each task is given in Table 2. The direct cost-duration relationship for task 2 (a linear relationship) and task 10 (a nonlinear relationship) are shown in Figures 5 and 6 as examples. An initial cost of $5000 and, thereafter, a daily cost of $500 is used as the indirect cost for this example.

Scheduling logic

The way in which each task is performed and the logic in which the tasks are carried out for a given project is not always well defined. Different schedulers may have different ideas for breaking down and sequencing each task. Often schedulers are constrained by the scheduling model available, forcing them to make simplifying assumptions. The flexible computational model presented in this article, however, allows schedulers a greater control over the progress of work and enables them to complete the job more efficiently.

Details of the breakdown of repetitive tasks into crews and segments, the start and finish distances, the quantities of work required, and the job condition factors for segments of work are given in Table 3. A constant number (1000 m) is used as the start distance of the project to avoid division by zero in the computation.

How the variation in the quantities of work and the job condition factors affect the breakdown of tasks can be explained by the clear and grub operations represented by tasks 1 and 9. Figure 7 shows the areas that have to be cleared and grubbed and the type of vegetation involved. Task 1 operates on the first 200 m of the roadway but also includes the area for the site office. Task 9 covers the remaining length of the highway including the site for the asphalt concrete plant. A new segment of work is required whenever there is a change in the quantity of work required per unit length of the highway and/or a change in the job condition factor. Each change will affect the production rate. To reflect such a change a separate segment of work is defined. Whenever there is no such changes, such as for task 14, there is no need to break down the work into smaller segments.

Base laying and paving operations (tasks 12 and 13) require material from the asphalt concrete plant. Therefore, as the operation moves away from the plant more time will be taken to do the same amount of work. In our example, we increased the time required for operations beyond 1250 m from the plant by 5 percent indicated by the job condition factor of 1.05. Instead of a step function, a continuous linear or nonlinear function may be used for the job condition factor that will reflect the impact of increasing haul distances on the rate of operation.

The internal logic of repetitive tasks is given in Table 4. Work continuity relationships between segments of work and multiple-crew strategies are specified. Usually no slack time ($S_{ij}^{k} = 0$) is allowed between segments of the work of a crew. However, the slack term can be any function of the decision variables. We use a nonlinear slack term to model the continuity of work constraint of task 8 in the form:

$$S = 1 - \beta \leq 1.0 \tag{49}$$

where $\beta < 1.0$ is the fractional portion of the finishing time (starting time plus duration) of the previous segment of work. For example, for a finishing time of 10 days and 2 hours (10.25 days assuming 8 hours per day) $\beta = 0.25$. This insures that the work on the next segment will start on the following day. As a result, adequate time is provided for the crew to move from one location to the next. Multiple-crew strategies become important when more than two crews are used.

Table 5 gives the external logic of tasks for the illustrative example. The external logic of the first 5 tasks, which are non-repetitive, can also be shown by an activity-on-node (AON) diagram (Figure 8). Standard precedence relationships, Eqs. (6) to (9), are used to link the tasks. The time lag term, however, may be any function of the decision variables.

The construction of a culvert cannot start unless the area has been cleared and grubbed. Therefore, the external logic of task 8 requires that work on any culvert be delayed until the crews of repetitive task 9 has worked through the corresponding location. A space buffer of 150 m is provided around earthmoving operations (task 10) to make sure adequate space is available for the equipment. Tasks 12 and 13 cannot start before the completion of the asphalt concrete plant. A minimum time buffer of 2 days is provided between tasks 11, 12, 13 and 14.

Solution of the problem

The direct cost optimization problem is solved for project durations of 60, 65, 70, 80, 90 and 100 days. The penalty parameter, r_n, is taken as (Adeli and Park, 1995a):

$$r_n = r_0 + \frac{n}{\alpha} \tag{50}$$

where r_0 is the initial penalty, n the iteration number and α is a positive number. Through this relationship the penalty is increased gradually in each iteration to avoid the possibility of numerical ill-conditioning. As stopping criteria, a change of less than \$1 in the original objective (direct cost) function and a maximum of 450 iterations are chosen. The convergence curves for the solutions are given in Figure 9. Table 6 and Figure 10 show the variation of direct, indirect, and total costs for different values of project duration. From Figure 10 and Table 6, a project duration of 70 days leads to the minimum total cost. The final global optimum schedule is shown as a linear planning chart in Figure 11.

CONCLUSION

A general formulation was presented for the scheduling of construction projects. Both repetitive and non-repetitive tasks are considered in the formulation. By specifying appropriate constraints, work continuity considerations and multiple-crew strategies can be modeled. The effects of varying job conditions on the performance of a crew is taken into account by introducing a job conditions factor which modifies the task duration computed on the basis of resource allocation only. This factor can be a constant, a linear, or a nonlinear function depending on the complexity of the situation. An optimization formulation is presented for the construction project scheduling problem with the goal of minimizing the direct construction cost. Any linear or non-linear function can be used for task direct cost-duration relationships. The nonlinear optimization problem is then solved by the neural dynamics model developed

recently by Adeli and Park (1995a). For any given construction duration, the model yields the optimum construction schedule for the minimum construction direct cost automatically. By varying this construction duration, one can solve the cost-duration trade-off problem and obtain the global optimum schedule and the corresponding minimum construction cost.

The new scheduling construction model provides the capabilities of both CPM and LSM approaches, In addition, it provides feature desirable for repetitive tasks such as highway construction projects and allows schedulers greater flexibility in modeling construction projects more accurately. In particular, it is suitable for studying the effects of change order on the construction cost. The new scheduling model can be specialized for the solution of specific and perhaps less complicated scheduling problems.

ACKNOWLEDGMENT

This manuscript is based on a research project sponsored by the Ohio Department of Transportation and Federal Highway Administration.

APPENDIX I. REFERENCES

Adeli, H. and Hung, S. L. (1995), *Machine Learning--Neural Networks, Genetic Algorithms, and Fuzzy Systems*, John Wiley & Sons, Inc., New York, NY.

Adeli, H. and Park, H. S. (1995a), "A Neural Dynamics Model for Structural Optimization--Theory," *Computers and Structures*, Vol.. 57, No. 1, pp. 383-390.

Adeli, H. and Park, H. S. (1995b), "Optimization of Space Structures by Neural Dynamics," *Neural Networks*, Vol. 8, No. 5, pp. 769-781.

Adeli, H. and Park, H. S. (1995c), "Counterpropagation Neural Networks in Structural Engineering," *Journal of Structural Engineering*, Vol. 121, No. 8, pp. 1205-1212.

Adeli, H. and Park, H. S. (1996a), "Hybrid CPN-Neural Dynamics Model for Discrete Optimization of Steel Structures," *Microcomputers in Civil Engineering*, Vol. 11, No. 5, pp. 355-366.

Adeli, H. and Park, H. S. (1996b), "Fully Automated Design of Super-High-Rise Buildings Structures by a Hybrid AI model on a Massively Parallel Machine," *AI Magazine*, Fall Issue, pp. 87-93.

Adeli, H. and Yeh, C. (1989), "Perceptron Learning in Engineering Design," *Microcomputers in Civil Engineering*, Vol. 4, No. 4, pp. 247-256.

Adeli, H. and Zhang, J. (1993), "An improved Perceptron Learning Algorithm," *Neural, Parallel, and Scientific Computations*, Vol. 1, No. 2, pp. 141-152.

Alsugair, A. M. and Chang, D. Y. (1994), "An Artificial Neural Network Approach to Allocating Construction Resources," *Proceedings of the First Congress on Computing in Civil Engineering*, Washington, DC, June 20-22, Vol. 1, ASCE, New York, NY, pp.950-957.

Chang, S. H. and Nam, B. H. (1993), "Linear Programming Neural Networks for Job-Shop Scheduling," *Proceedings of the International Joint Conference on Neural Networks*, Nagoya, Japan, Oct. 25-29, Vol. 2, IEEE Press, New York, NY, pp. 1557-1560.

Foo, S. Y., Takefuji, Y. and Szu, H. (1995), "Scaling Properties of Neural Networks for Job-Shop Scheduling," *Neurocomputing*, Vol. 8, No. 1, pp. 79-91.

Gulati, S., Iyengar, S. S., Toomarian, N., Protopopescu, V. and Barhen, J. (1987), "Nonlinear Neural Networks for Deterministic Scheduling," *IEEE First International Conference on Neural Networks*, San Diego, CA, June 21-24, Vol. IV, IEEE Press, New York, NY, pp.745-752.

Handa, M. and Barcia, R. M. (1986), "Linear Scheduling Using Optimal Control Theory," *Journal of Construction Engineering and Management*, Vol.. 112, No. 3, pp. 387-393.

Hopfield, J. J. and Tank, D. W. (1985), "'Neural' Computations of Decisions in Optimization Problems," *Biological Cybernetics*, Vol.. 52, No. 3, pp. 141-152.

Johnston, D. W. (1981), "Linear Scheduling Method for Highway Construction," *Journal of the Construction Division*, ASCE, Vol.. 107, No. CO2, pp. 247-261.

Kolk, W. R. and Lerman, R. A. (1992), *Nonlinear System Dynamics*, Van Nostrand Reinhold, New York, NY.

Liang, L., Burns, S. A. and Feng, C, -W. (1995), "Construction Time-Cost Trade-Off Analysis Using LP/IP Hybrid Method," *Journal of Construction Engineering and Management*, Vol. 121, No. 4, pp. 446-454.

Mohammed, H. A., Abd El Halim, A. O. and Razaqpur, A. G. (1995), "Use of Neural Networks in Bridge Management Systems," *Transportation Research Record*, No. 1490, pp. 1-8.

Park, H. S. and Adeli, H (1995), "A Neural Dynamics Model for Structural Optimization--Application to Plastic Design of Structures," *Computers and Structures*, Vol.. 57, No. 3, pp. 391-400.

Pellerin, D. and Herault, J. (1994), "Scheduling with Neural Networks: Applications to Timetable Construction," *Neurocomputing*, Vol.. 6, No. 4, pp. 419-442.

Reda, R. M. (1990), "RPM: Repetitive Project Modeling," *Journal of Construction Engineering and Management*, Vol.. 116, No. 2, pp. 316-330.

Russell, A. D. and Caselton, W. F. (1988), "Extensions to Linear Scheduling Optimization," *Journal of Construction Engineering and Management*, Vol.. 114, No. 1, pp. 36-52.

Russell, A. D. and Wong, W. C. M. (1993), "New Generation of Planning Structures," *Journal of Construction Engineering and Management*, Vol. 119, No. 2, pp. 196-214.

Selinger, S. (1980), "Construction Planning for Linear Projects," *Journal of Construction Division*, ASCE, Vol.. 106, No. CO2, pp. 195-205.

Watanabe, T., Tokumaru, H. and Hashimoto, Y. (1993), "Job-Shop Scheduling Using Neural Networks," *Control Engineering Practice*, Vol.. 1, No. 6, pp. 957-961.

Willems, T. M. and Rooda, J. E. (1994), "Neural Networks for Job-Shop Scheduling," *Control Engineering Practice*, Vol.. 2, No. 1, pp. 31-39.

APPENDIX II. NOTATIONS

The following symbols are used in this paper:

B_{Sij} = space (distance) buffer between tasks i and j;

B_{Tij} = time buffer between tasks i and j;

C_i = direct cost of executing unit quantity of work of task i;

C_D = total project direct cost;

d_i^k = duration per unit quantity of work of task i performed by crew k based on resource allocation only;

$(d_i^k)^{\max}$ = maximum possible value of d_i^k;

$(d_i^k)^{\min}$ = minimum possible value of d_i^k;

D_{ij}^k = the actual duration of executing segment j by crew k of task i;

$D^{\max}$ = maximum acceptable project duration;

$f(\mathbf{X})$ = objective function;

$g_j(\mathbf{X})$ = jth inequality constraint;

$h_k(\mathbf{X})$ = kth equality constraint;

J = number of inequality constraints;

K = number of equality constraints;

$L(\mathbf{X}, \mathbf{u}, \mathbf{v}, \mathbf{s})$ = Lagrangian function;

L_{FFij} = finish-to-finish time lag between task j and a preceding task i;

L_{FFi}^{kl} = finish-to-finish time lag between new crew l and a previous crew k of

task i;

L_{FSij} = finish-to-start time lag between task j and preceding task i;

L_{SFij} = start-to-finish time lag between task j and a preceding task i;

L_{SFi}^{kl} = start-to-finish time lag between new crew l and a previous crew k of task i;

L_{SSij} = start-to-start time lag between task j and a preceding task i;

L_{SSi}^{kl} = start-to-start time lag between new crew l and a previous crew k of task i;

N = number of decision variables;

N_{Ci} = number of crews used for task i;

N_{Si}^{k} = number of segments over which crew k of task i performs;

N_{NT} = number of non-repetitive tasks;

N_{RT} = number of repetitive tasks;

N_T = number of tasks in the project;

$P(\mathbf{X}, r_n)$ = pseudo-objective function;

r_n = penalty parameter;

r_o = initial value of the penalty parameter;

s_j = slack for the jth inequality constraint;

S = slack time;

S_{ij}^{k} = idle or slack time of crew k after segment j of task i;

T_{ij}^{k} = time at which crew k of task i starts work on segment j;

$T_i(Z_{jn}^k)$ = time of task i defined by location Z_{jn}^k;

u_j = Lagrangian multiplier for the jth inequality constraint;

v_k = Lagrangian multiplier for the kth equality constraint;

$V(\mathbf{X})$ = Lyapunov function;

W_{ij} = quantity of work required in segment j of task i;

$\mathbf{X}$ = vector of decision variables;

Z_{ij}^k = distance at which crew k of task i starts work on segment j;

$Z_{ij}^{k'}$ = distance at which crew k of task i finishes work on segment j;

$Z_i(T_{jn}^k)$ = location of task i defined by time T_{jn}^k;

β = parameter used in the slack term;

μ_{ij}^k = job condition factor for segment j performed by crew k of task.

Table 1. The description and type of tasks in the illustrative example

Task #	Description	Type
1	Clear and grub site for temporary offices plus right-of-way	Non-repetitive
2	Grade site for temporary offices	Non-repetitive
3	Erect temporary offices	Non-repetitive
4	Construct temporary roads	Non-repetitive
5	Move-in	Non-repetitive
6	Grade asphalt concrete plant site	Non-repetitive
7	Erect asphalt concrete plant	Non-repetitive
8	Construct culverts	Repetitive
9	Clear and grub right-of-way	Repetitive
10	Earthwork	Repetitive
11	Lay sub-base	Repetitive
12	Lay base	Repetitive
13	Pave	Repetitive
14	Finish shoulders	Repetitive

Table 2. The direct cost-duration relationship for each task

Task #	Direct cost-duration relationship	Range (days)
1	$C = -300d + 1050$	$1.0 \le d \le 1.5$
2	$C = -280d + 960$	$0.5 \le d \le 2.0$
3	$C = -200d + 250$	$0.25 \le d \le 0.50$
4	$C = -200d + 550$	$0.50 \le d \le 1.25$
5	$C = -150d + 550$	$1.0 \le d \le 2.0$
6	$C = -280d + 960$	$0.5 \le d \le 2.0$
7	$C = -400d + 5700$	$5 \le d \le 8$
8	$C = 1600 / d$	$2 \le d \le .3$
9	$C = -300d + 1050$	$1.0 \le d \le 1.5$
10	$C = (1600 + 500d) / d$	$1.0 \le d \le 2.0$
11	$C = -200d + 850$	$0.75 \le d \le 1.25$
12	$C = -200d + 950$	$0.75 \le d \le 1.25$
13	$C = -200d + 900$	$0.75 \le d \le 1.25$
14	$C = -100d + 800$	$2 \le d \le 4$

Table 3. Task details for the illustrative example

Task #	Crew #	Segment #	Distances (m)		Quantity of work	Unit	μ
			Start	Finish			
1	1	-	1000	1200	3	hectares	1.0
2	1	-	1000	1200	3	hectares	1.0
3	1	-	1000	1200	5	units	1.0
4	1	-	1000	1200	3	100 m	1.0
5	1	-	1000	1200	100	percent	1.0
6	1	-	3500	3650	1.5	hectares	1.0
7	1	-	3500	3650	100	percent	1.0
8	1	1	1300	1305	1	culvert	1.0
		2	2750	2755	1	culvert	1.0
		3	5500	5505	1	culvert	1.0
9	1	1	1200	3000	4.5	hectares	1.0
		2	3000	3500	1.25	hectares	1.1
		3	3500	3650	1.875	hectares	1.1
		4	3650	6000	5.875	hectares	1.15
10	1	1	1000	3000	10	1000 m^3	1.0
		2	3000	3500	6	1000 m^3	1.15
	2	1	3500	5000	8	1000 m^3	1.05
		2	5000	6000	5	1000 m^3	1.0
11	1	1	1000	2000	4.25	1000 m^3	1.0
		2	2000	4000	8.5	1000 m^3	1.05
		3	4000	6000	8.5	1000 m^3	1.0
12	1	1	3500	2250	2.6	1000 m^3	1.0
		2	2250	1000	2.6	1000 m^3	1.05
	2	1	3500	4750	2.6	1000 m^3	1.0
		2	4750	6000	2.6	1000 m^3	1.05
13	1	1	3500	2250	1.25	1000 m^3	1.0
		2	3250	1000	1.25	1000 m^3	1.05
	2	1	3500	4750	1.25	1000 m^3	1.0
		2	4750	6000	1.25	1000 m^3	1.05
14	1	1	1000	6000	3	hectares	1.0

Table 4. The internal logic of repetitive tasks

Task #	Crew #	Segment #	Continuity relationship	Multiple-crew strategy	
				Predecessor crew	Relationship
8	1	1	-	-	-
		2	$S=1-\beta$ [a]		
		3	$S=1-\beta$ [a]		
9	1	1	-	-	-
		2	$S=0$		
		3	$S=0$		
		4	$S=0$	-	-
10	1	1	-		
		2	$S=0$	1	SS, $L=0$
	2	1	-		
		2	$S=0$		
11	1	1	-	-	-
		2	$S=0$		
		3	$S=0$		
12	1	1	-	-	-
		2	$S=0$		
	2	1	-	1	FF, $L=0$
		2	$S=0$		
13	1	1	-	-	-
		2	$S=0$		
	2	1	-	1	SS, $L=0$
		2	$S=0$		

[a] β is the fractional portion of the finishing time of the previous segment of work

Table 5. The external logic of tasks

Task	Predecessor	Relationship
Task 1		
Task 2	Task 1	FS, $L = 0$
Task 3	Task 2	SS, $L = 0$
	Task 4	FS, $L = 0$
Task 4	Task 2	FS, $L = 0.25D$
Task 5	Task 3	FS, $L = 0$
Task 6	Segment 3, Crew 1, Task 9	FS, $L = 0$
Task 7	Task 6	FS, $L = 0$
Task 8: Crew 1, Segment 1	Task 9 at 1300 m	FS, $L = 0$
Crew 1, Segment 2	Task 9 at 2750 m	FS, $L = 0$
Crew 1, Segment 3	Task 9 at 5500 m	FS, $L = 0$
Task 9	Task 1	FS, $L = 0$
Task 10	Task 9	Space buffer, $B = 150$ m
Task 11	Task 10	Space buffer, $B = 150$ m
Task 12	Task 7	FS, $L = 0$
	Task 11	Time buffer, $B = 2$ days
Task 13	Task 12	Time buffer, $B = 2$ days
Task 14	Task 13	Time buffer, $B = 2$ days

Table 6. The direct, indirect, and total costs variation for the illustrative example

Duration (days)	Direct cost (dollars)	Indirect cost (dollars)	Total cost (dollars)
60	94118	30000	124118
65	91215	32500	123715
70	87314	35000	122414
80	85742	40000	125742
90	85438	45000	130438
100	84411	50000	134411

LIST OF CAPTIONS FOR FIGURES

1. A linear planning chart
2. Typical variation of direct, indirect and total costs of a construction project
3. Direct cost-duration curve for unit quantity of task *i*

   (a) Linear, (b) Nonlinear
4. The neural network topology for the neural dynamics cost optimization model
5. The linear direct cost-duration curve for unit quantity of task 2
6. The nonlinear direct cost-duration curve for unit quantity of task 10
7. The areas and types of vegetation that have to be cleared by tasks 1 and 9
8. Activity-on-node diagram for the first five tasks of the illustrative example
9. The direct cost convergence curves for the illustrative example
10. Time-cost trade-off curve for the illustrative example
11. The linear planning chart for the minimum total cost schedule

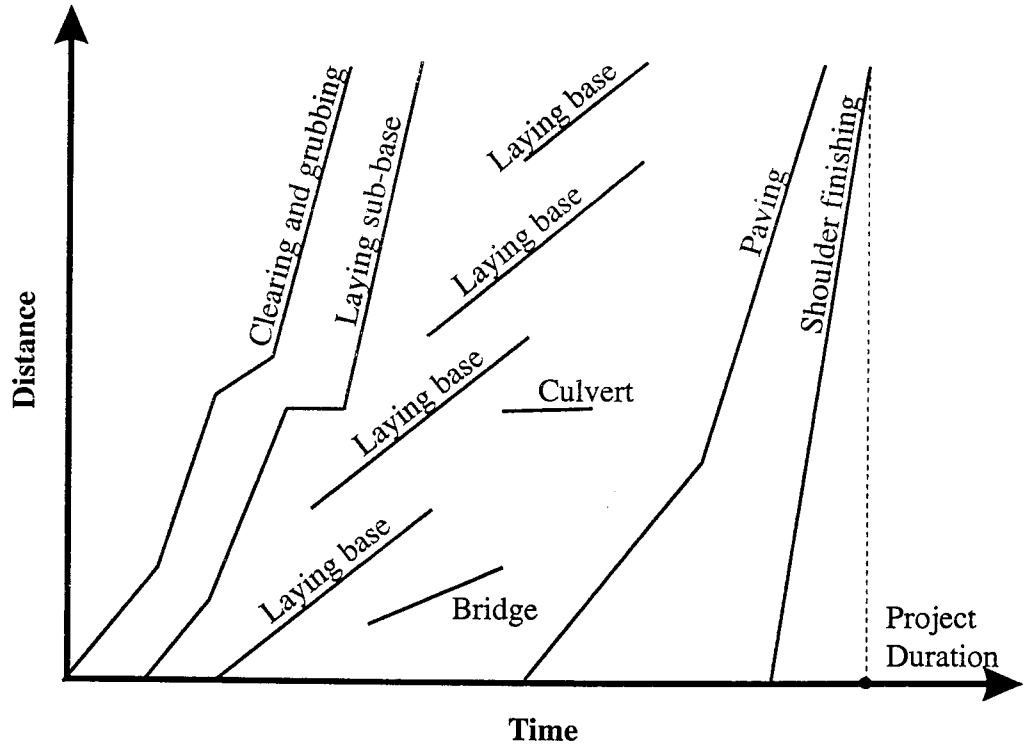

Distance
Clearing and grubbing
Laying sub-base
Laying base
Laying base
Laying base
Culvert
Laying base
Bridge
Paving
Shoulder finishing
Project
Duration
Time

Figure 1

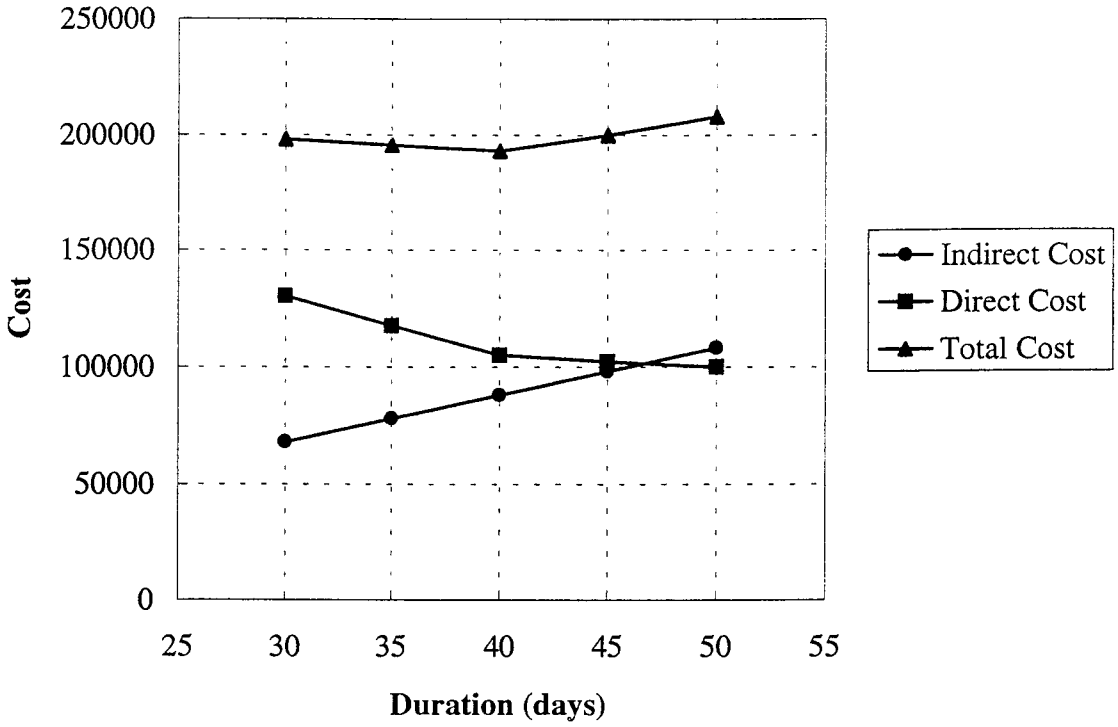

250000
200000
150000
100000
50000
0
Cost
25
30
35
40
45
50
55
Duration (days)
Indirect Cost
Direct Cost
Total Cost

Figure 2

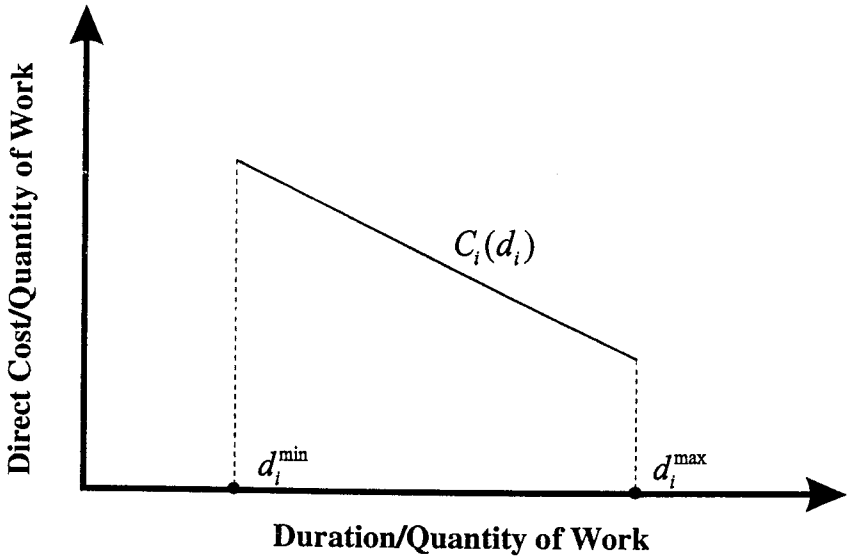

Direct Cost/Quantity of Work
$C_i(d_i)$
$d_i^{\min}$
$d_i^{\max}$
Duration/Quantity of Work

Figure 3a

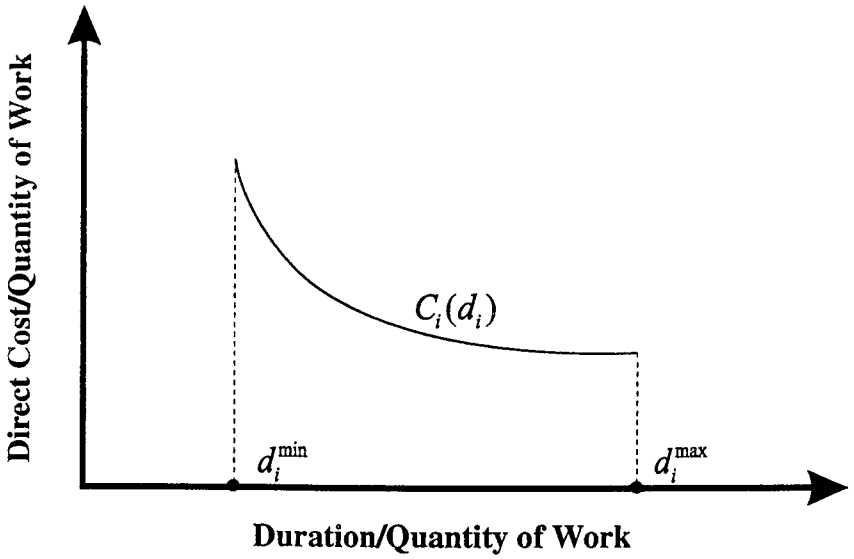

Direct Cost/Quantity of Work
$C_i(d_i)$
d_i^{min}
d_i^{max}
Duration/Quantity of Work

Figure 3b

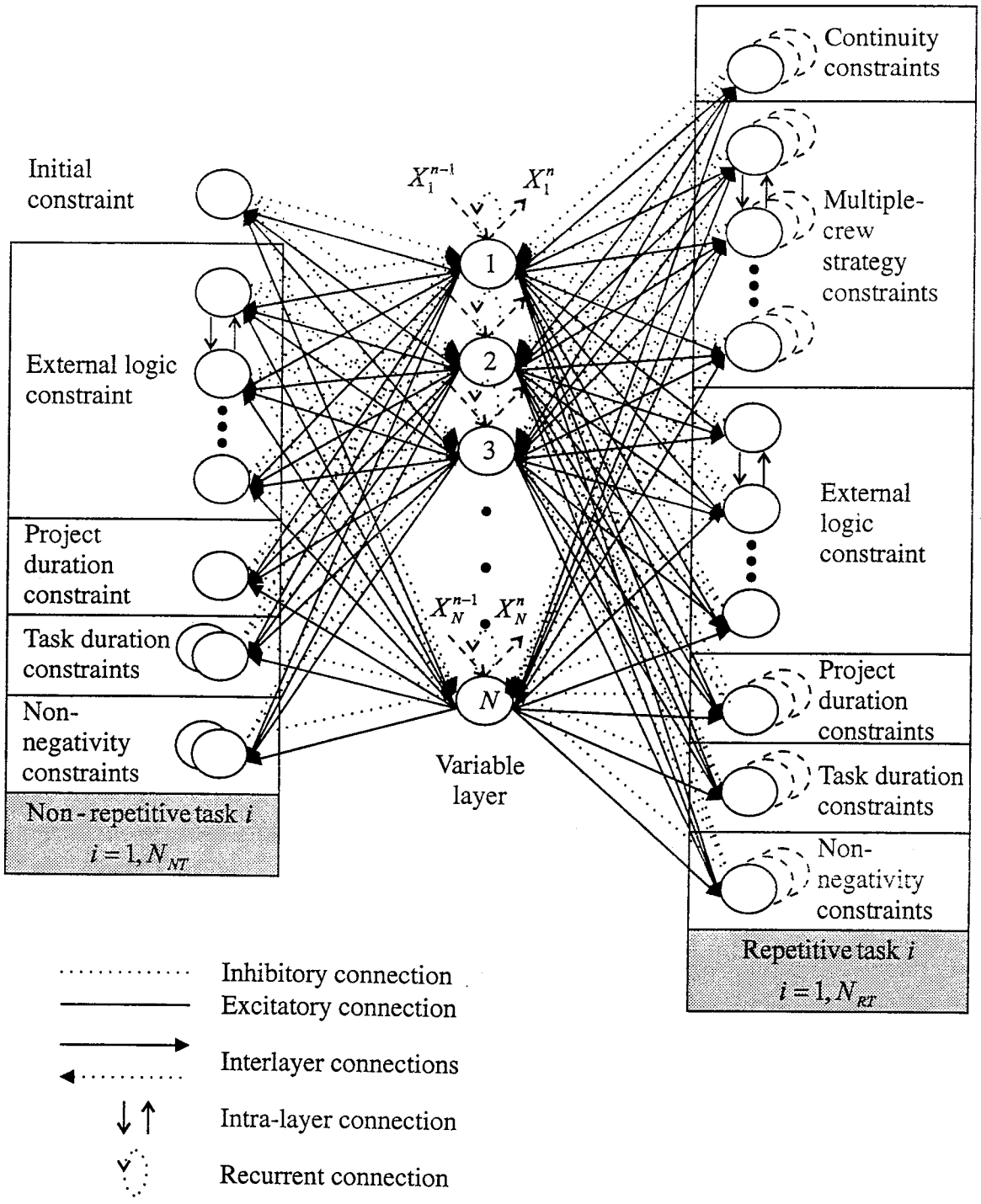

Initial constraint
External logic constraint
Project duration constraint
Task duration constraints
Non-negativity constraints
Non-repetitive task i
$i = 1, N_{NT}$
X_1^{n-1}
X_1^n
1
2
3
X_N^{n-1}
X_N^n
N
Variable layer
Continuity constraints
Multiple-crew strategy constraints
External logic constraint
Project duration constraints
Task duration constraints
Non-negativity constraints
Repetitive task i
$i = 1, N_{RT}$
Inhibitory connection
Excitatory connection
Interlayer connections
Intra-layer connection
Recurrent connection

Figure 4

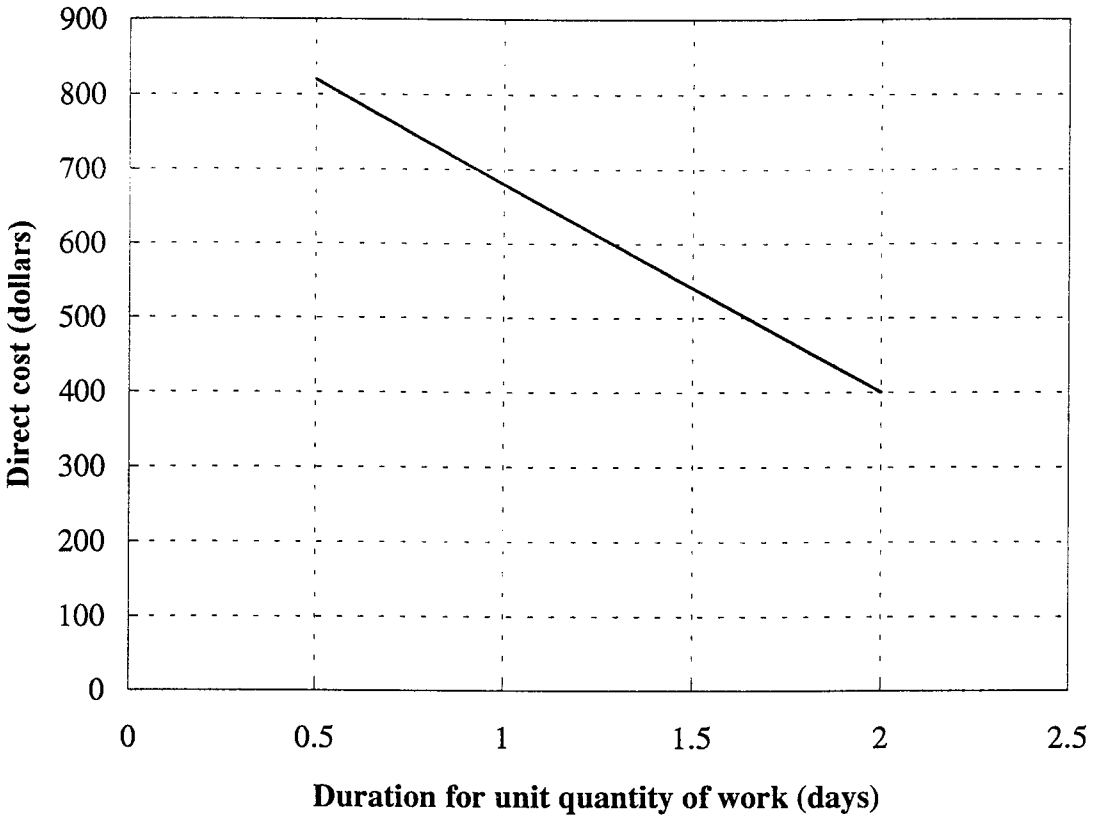

900
800
700
600
500
400
300
200
100
0
Direct cost (dollars)
0
0.5
1
1.5
2
2.5
Duration for unit quantity of work (days)

Figure 5

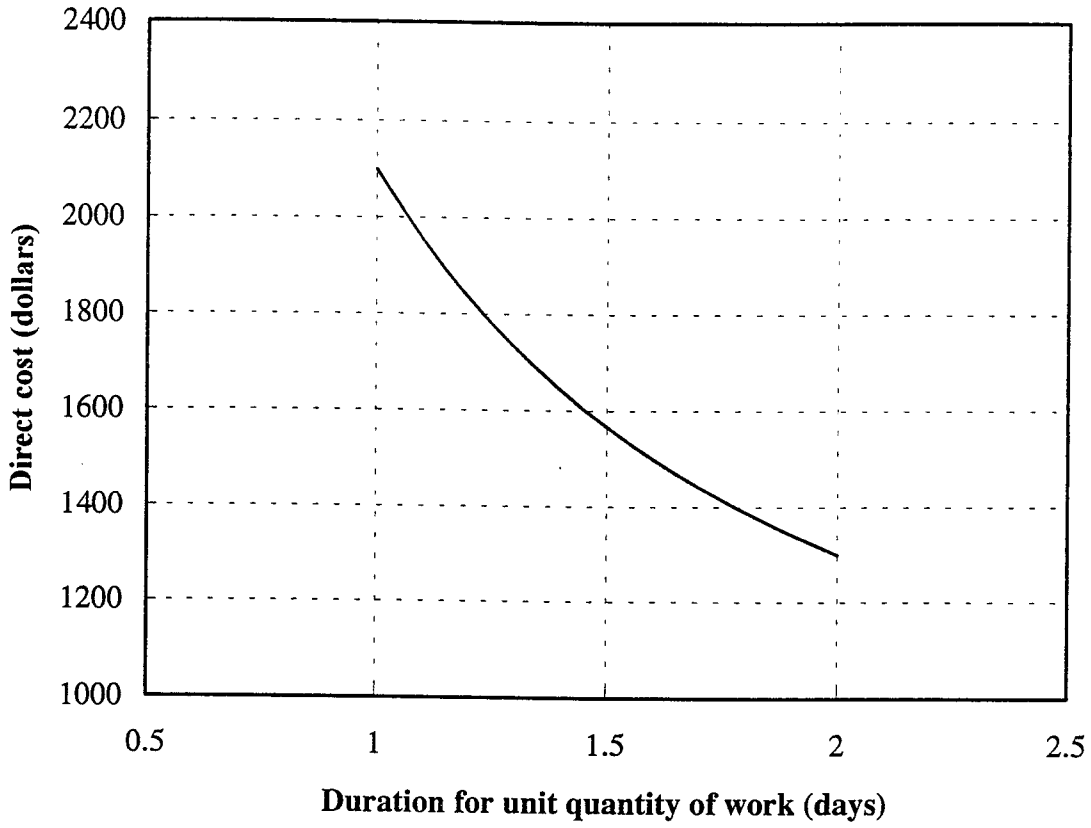

2400
2200
2000
1800
1600
1400
1200
1000
Direct cost (dollars)
0.5
1
1.5
2
2.5
Duration for unit quantity of work (days)

Figure 6

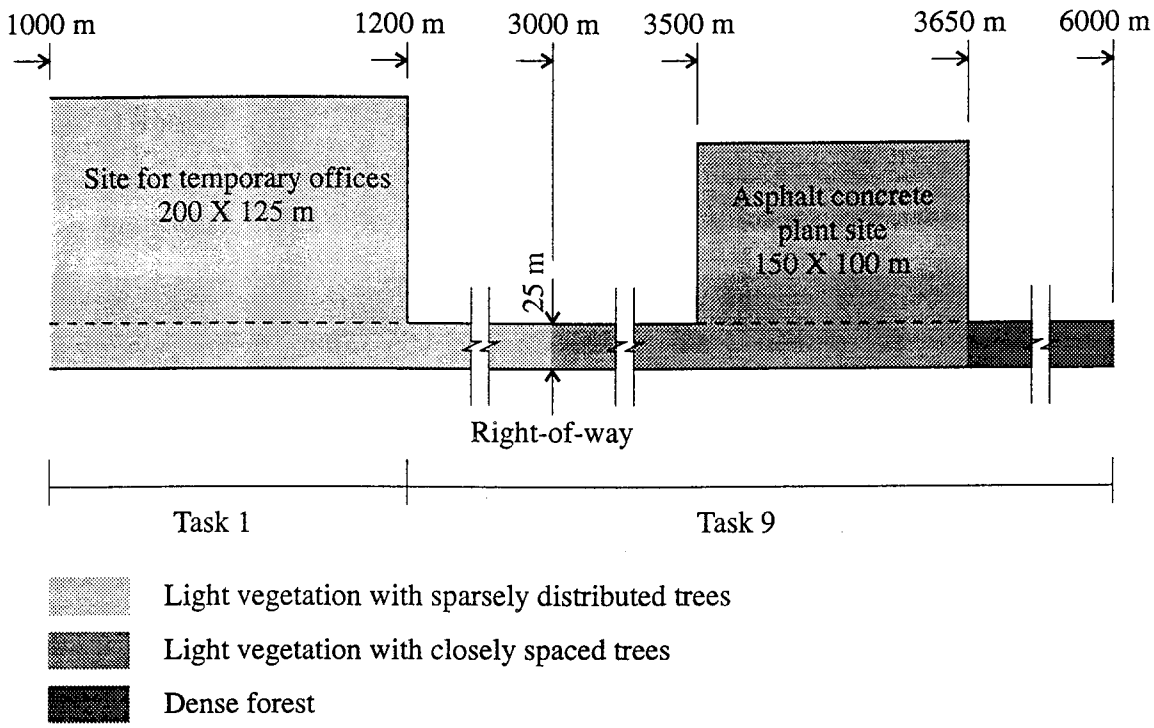
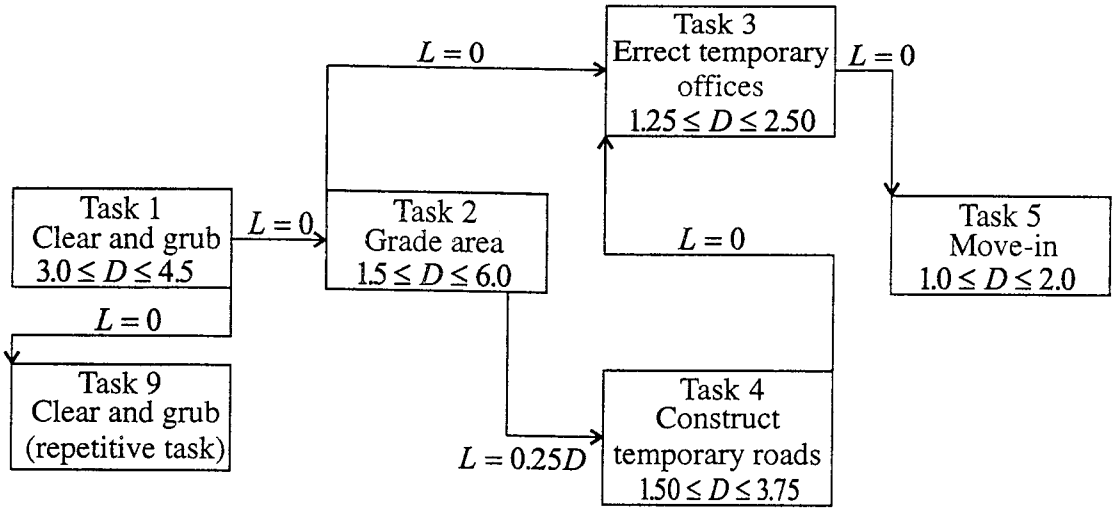

1000 m
1200 m
3000 m
3500 m
3650 m
6000 m
Site for temporary offices
200 X 125 m
Asphalt concrete
plant site
150 X 100 m
25 m
Right-of-way
Task 1
Task 9
Light vegetation with sparsely distributed trees
Light vegetation with closely spaced trees
Dense forest

Figure 7



D = Duration in days

L = Lag in days

Figure 8

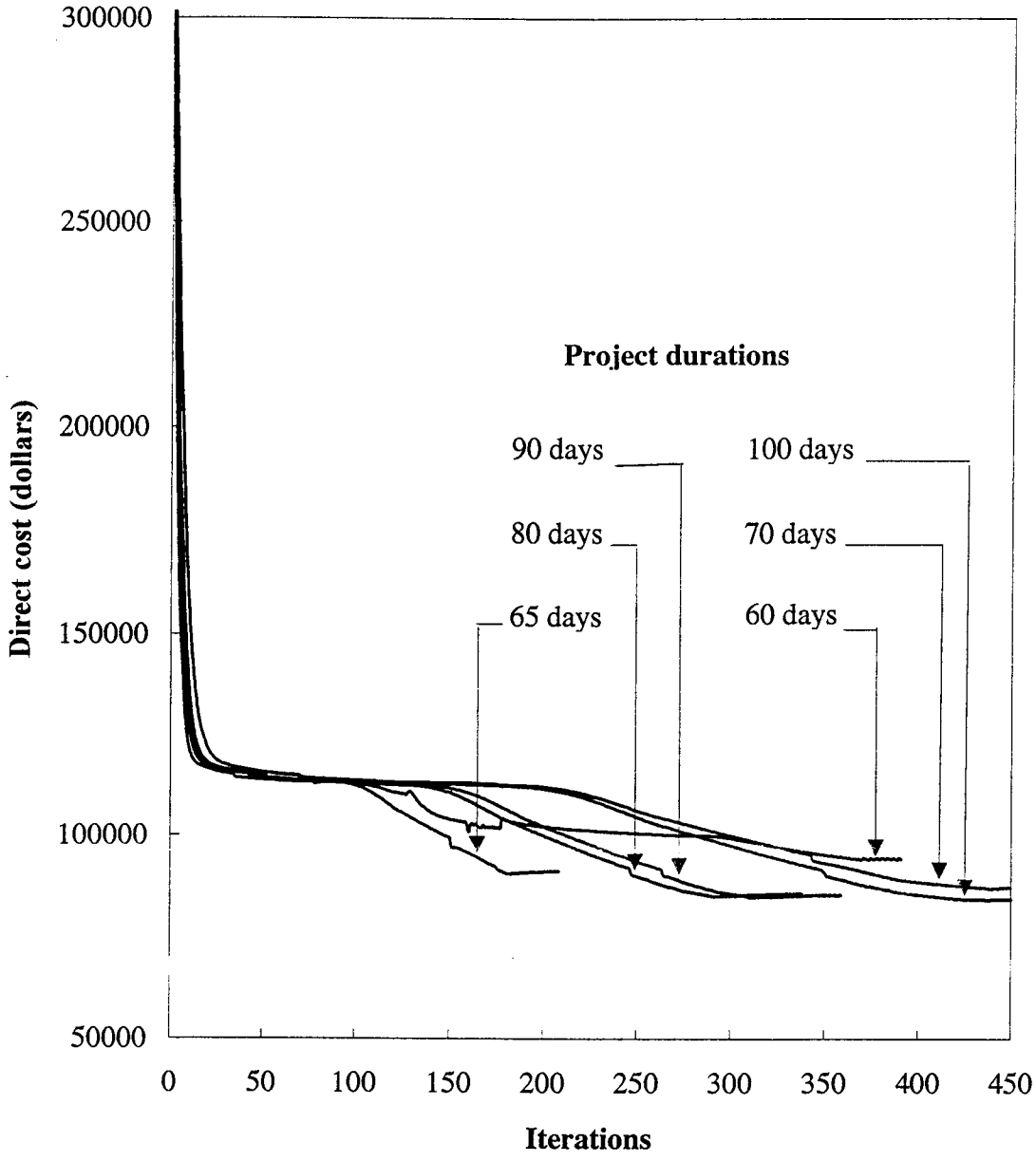

300000
250000
200000
150000
100000
50000
Direct cost (dollars)
Project durations
90 days
100 days
80 days
70 days
65 days
60 days
0
50
100
150
200
250
300
350
400
450
Iterations

Figure 9

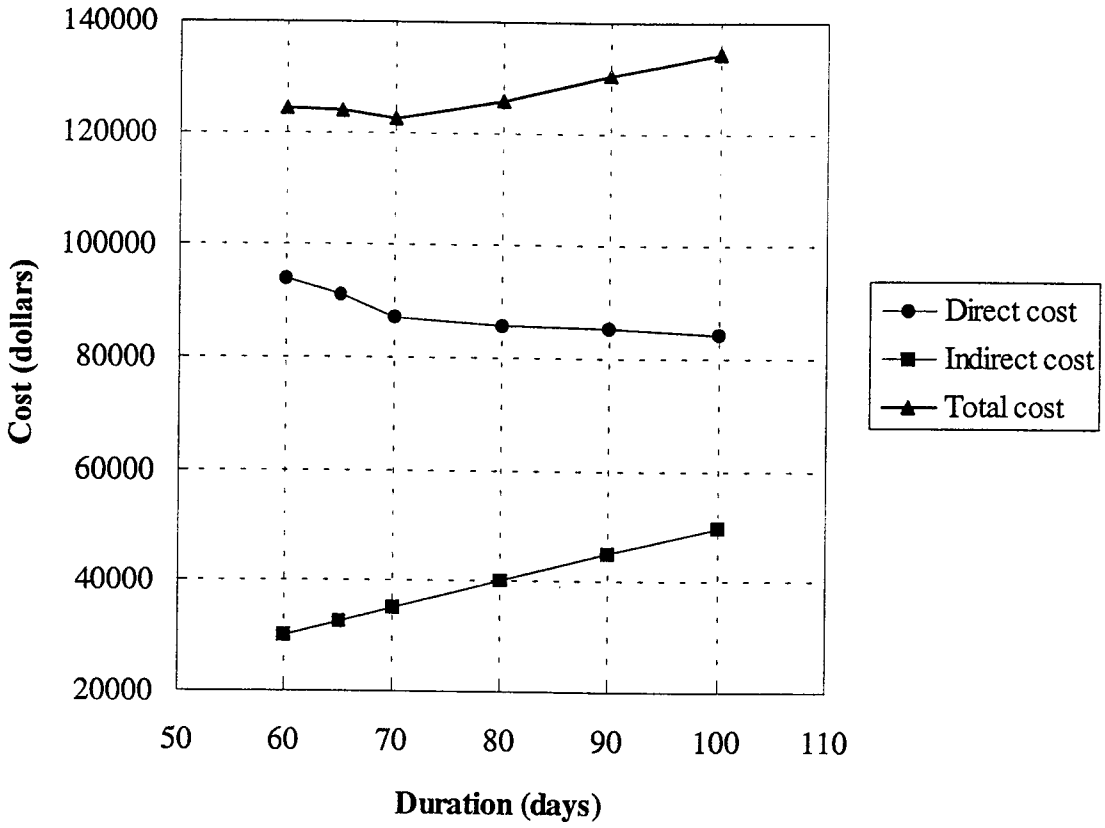

140000
120000
100000
80000
60000
40000
20000
Cost (dollars)
50
60
70
80
90
100
110
Duration (days)
Direct cost
Indirect cost
Total cost

Figure 10

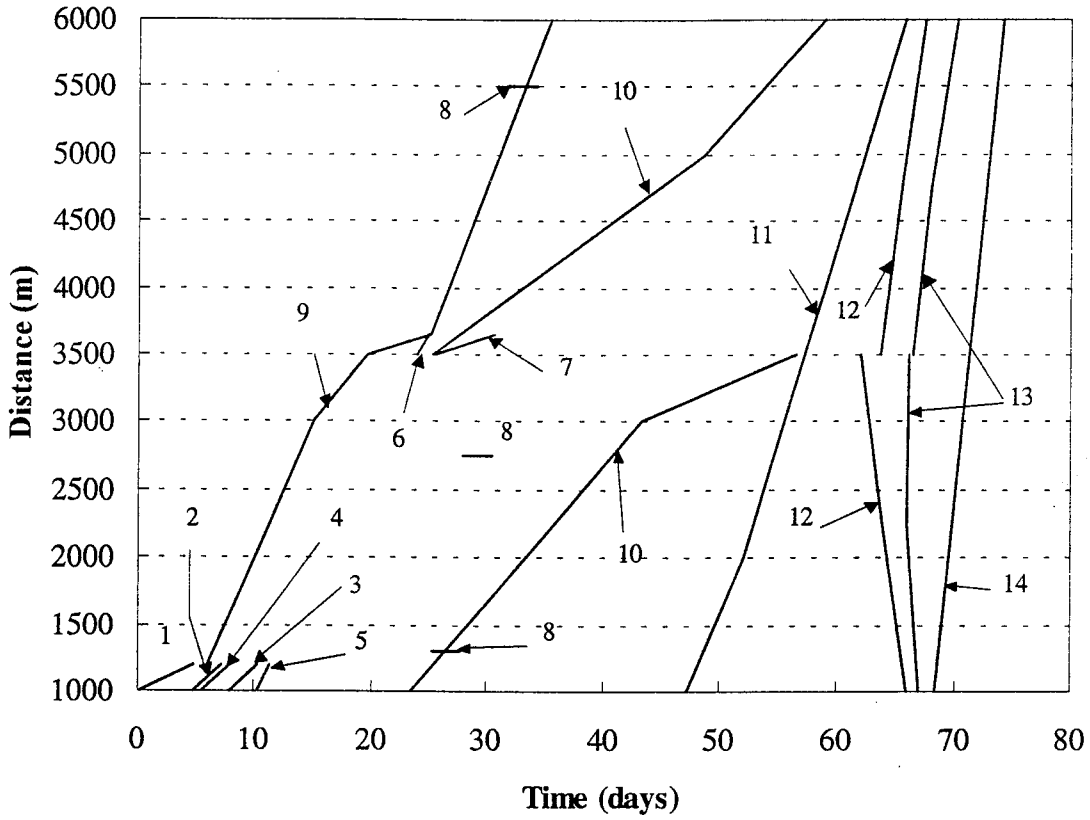

6000
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
Distance (m)
0
10
20
30
40
50
60
70
80
Time (days)
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Figure 11

REGULARIZATION NEURAL NETWORK MODEL FOR HIGHWAY CONSTRUCTION COST ESTIMATION

Hojjat Adeli* and Mingyang Wu**

ABSTRACT: Estimating the cost of a construction project is an important task in the management of construction projects. Quality of the construction management depends on the accurate estimation of the construction cost. Highway construction costs are very noisy and the noise is the result of many unpredictable factors. In this article, a regularization neural network is formulated and a neural network architecture is presented for estimating the cost of construction projects. The model is applied to estimate the cost of reinforced concrete pavements as an example. The new computational model is based on a solid mathematical foundation making the cost estimation consistently more reliable and predictable. Further, the result of estimation from the regularization neural network depends only on the training examples. It does not depend on the architecture of the neural network, the learning parameters, and the number of iterations required for training the system. Moreover, the problem of noise in the data is taken into account in a rational manner.

---

* Professor, Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 470 Hitchcock Hall, 2070 Neil Avenue, Columbus, Ohio, 43210.

** Graduate Research Associate, Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University.

1. INTRODUCTION

Estimating the cost of a construction project is an important task in the management of construction projects. Quality of the construction management depends on the accurate estimation of the construction cost. This task is currently performed by "*experienced*" construction cost estimators in a highly subjective manner. Such a subjective analysis is subject to human errors and varying results depending on who the construction cost estimator is and possible litigation consequences.

Automating the process of construction cost estimation based on objective data is highly desirable not only for improving the efficiency but also for removing the subjective questionable human factors as much as possible. The problem is not amenable to traditional problem solving approaches. The costs of construction materials, equipment, and labor depend on numerous factors with no explicit mathematical model or rule for price prediction. Recently, neural networks have been used for learning and prediction problems with no explicit model such as securities price prediction (Hutchinson et al., 1994), system identification (Narendra and Parthasarathy, 1990), engineering design (Hung and Adeli, 1991), and image recognition (Adeli and Hung, 1995).

Several authors have discussed potential applications of neural networks in construction in recent years. Moselhi et al. (1991) point out the potential applications of neural networks in the general area of construction. Moselhi et al. (1993) used the backpropagation neural networks (Hegazy et al., 1994) and the genetic algorithm (Adeli

and Hung, 1995) to develop a decision-support system to aid the contractors in preparing bids. Backpropagation algorithm has also been used for estimating construction productivity (Chao and Skibniewski, 1994), evaluation of new construction technology acceptability (Chao and Skibniewski, 1995), and for condition rating of roadway pavement sections (Eldin and Senouci, 1995). Kartam (1996) uses neural networks to determine optimal equipment combinations for earthmoving operations. Pompe and Feelders (1997) use neural networks to predict corporate bankruptcy.

Learning from previous data or examples, neural networks can make very reasonable estimations without using specific experts and rules. As an example the price of a concrete pavement is influenced by a number of factors including the quantity, the dimension (thickness of the pavement), the local economic factors, and the year of the construction. The problem to be investigated is whether using the values for these factors obtained from the past construction projects a neural network model can estimate the price of a future construction project accurately.

In this article, first the concepts of estimation, learning, and noisy curve fitting are described and formulated mathematically. Next, a special case of radial-based function neural networks, called regularization neural network, is formulated for estimating the cost of construction projects. Then, the model is applied to reinforced concrete pavement cost estimation as an example.

2. ESTIMATION, LEARNING, AND NOISY CURVE FITTING

The most fundamental problem that neural networks have been used to solve is *learning*. But, it is very difficult, if not impossible, to present a precise definition of learning. In order to model learning computationally, however, it has to be defined in a pragmatic manner rather than as an abstract concept. Learning can be defined as a self-organizing process, a mapping process, an optimization process, or a decision making process. The last definition is based on the observation that given a set of examples and a stimulus one makes a decision about the response to the stimulus.

Consider the special case of supervised learning where the system is first trained by a set of input-output examples. Then, given a new input the learner decides the output. This is an ill-posed problem because the answer can be any multitude of values. The selected answer depends on the generalization criteria or constraints chosen for the decision process. The advantage of viewing the learning process as a decision making process is its explicit representation of the generalization criteria.

An estimation problem can be formulated as a supervised learning problem. Consider a one-input-one-output noisy system with input x and output y. The system can be expressed mathematically as

$$y = f(x) + e \quad , \tag{1}$$

where f(x) is a function of the input variable x and e is an independent noise function with zero mean. A set of input-output examples x_i and y_i (i = 1, 2, ..., N) is given. The estimation problem is for any given input x find the best estimate of the output y. The best estimate can be defined as the estimate that minimizes the average error between the real and estimated outputs. Thus, such a supervised learning problem becomes equivalent to a mapping or curve fitting problem in a multi-dimensional hyperspace. It must be pointed

out that the traditional statistical approaches to curve fitting such as the regression analysis fail to represent problems with no explicit mathematical model accurately in a multi-dimensional space of variables. The neural network approach, on the other hand, can solve such problems more effectively.

Figure 1 shows a very simple example of curve fitting and learning. The dots represent the example data points that include noise. The dashed line represents the properly learned curve. The solid line represents the over-fitted learned curve. For this curve, the training error is very small (because the learned curve passes through all the training data points), but the estimation or generalization error is large. This is due to the fact that the influence of the noise has not been taken into account at all. This problem is referred to as *overfitting* which leads to less than satisfactory learning. Avoiding the overfitting problem is very important for accurate estimation and learning.

A mathematical definition of learning is now formulated as a mapping (generalization of curve-fitting) problem in a multi-dimensional hyperspace. A neural network is designed to perform a nonlinear mapping function s from a p-dimensional input space R^p to a one-dimensional output space R^1.

$$s: R^p \rightarrow R^1 \tag{2}$$

The set of N available input-output data can be described as

Input signal: $\mathbf{x}_i \in R^p, \quad i = 1, 2, \ldots, N$

Example output signal: $d_i \in R^1, \quad i = 1, 2, \ldots, N$

Where $\mathbf{x}_i = \left(x_1^i, x_2^i, \cdots, x_p^i\right)$ is the ith example with p input attributes, (x_n^i is the nth attribute of the ith example.) and d_i is the corresponding example output. The approximation mapping function is denoted by $F(\boldsymbol{x})$.

What is the *best fit*? This is an important question. Because of the existence of the noise in the data examples, a *perfect fit*, that is when $F(\mathbf{x}_i) = d_i$, usually is not the *best fit*. In this case, the approximation function is often very *curvy* with numerous steep peaks and valleys which leads to poor generalization. This is the overfitting problem mentioned earlier. Two other fitting situations can also be recognized: *underfitting* with over-smooth surfaces resulting in poor generalization and *proper fitting*. Only the last type of fitting can lead to accurate generalization and estimation and this is the research challenge. A method to achieve proper fitting will be discussed in the following sections.

Highway construction costs are affected by many factors, but only a few main factors are usually recorded and can be considered in the mathematical modeling of the cost estimation problem. As such, the highway construction data are very noisy and the noise is the result of many unpredictable factors such as human judgmental factors, random market fluctuations, and weather conditions. Consequently, finding a properly-fitted approximation is extremely important. Otherwise, the predicted cost will have a substantial error.

One approach to solve this problem is the multilayer feedforward backpropagation neural network (Haykin, 1994). The problem with this approach is that the generalization properties (underfitted, overfitted, or properly fitted) depend on many factors including the architecture (number of hidden layers and number of nodes in the hidden layers), initial

weights of the links connecting the nodes, and number of iterations for training the system. The performance of the algorithm depends highly on the selection of these parameters. The problem of arbitrary trial-and-error selection of the learning and momentum ratios encountered in the momentum backpropagation learning algorithm can be circumvented by the adaptive conjugate gradient learning algorithm developed by Adeli and Hung (1994). But, that algorithm does not address the issue of noise in the data. In the highway construction estimation problem, the data has substantial noise and the neural network algorithm must be able to address the issue of noise in the data properly. In this article we employ a neural network architecture called regularization network to obtain properly fitted approximation function and solve the construction estimation problem accurately.

3. REGULARIZATION NETWORKS

According to the regularization theory (Tikhonov and Arsenin, 1977; Haykin, 1994), the approximation mapping function F is determined by minimizing an error function, $E(F)$, consisting of two terms in the following form:

$$E(F) = E_s(F) + E_c(F) \tag{3}$$

The first term is the standard error term measuring the error between the sample example response d_i and the corresponding computed response o_i and is defined as follows:

$$E_s(F) = \frac{1}{2}\sum_{i=1}^{N}(d_i - o_i)^2 = \frac{1}{2}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]^2 \tag{4}$$

As discussed above, the perfect fit may not be the best answer due to the noise in the data. In order to overcome the overfitting problem a regularization term is added to the

standard error term whose function is to smoothen the approximation function. This term is defined as:

$$E_c(F) = \frac{1}{2}\|PF\|^2 \tag{5}$$

where the symbol $\|g\|$ denotes the norm of function $g(\mathbf{x})$ defined as:

$$\|g\|^2 = \int_{R^p} [g(\mathbf{x})]^2 dx_1 dx_2 \cdots dx_p \tag{6}$$

and P is a linear differential operator defined as (Poggio and Girosi, 1990; Al-Gwaiz, 1992):

$$\|PF\|^2 = \sum_{k=0}^{K} b_k \left\|D^k F(\mathbf{x})\right\|^2 \tag{7}$$

In Eq. (7), K is a positive integer, b_k's ($k=0,1,\ldots,K$) are positive real numbers, and the norm of the differential operator D^k is defined as

$$\left\|D^k F\right\|^2 = \sum_{|a|=k} \int_{R^p} \left[\partial^a F(\mathbf{x})\right]^2 dx_1 dx_2 \cdots dx_p \tag{8}$$

The *multi-index* $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_p)$ is a sequence of non-negative integers whose order is defined as $|\alpha| = \sum_{i=1}^{p} \alpha_i$. In Eq. (8), the partial differential term inside the bracket is defined as

$$\partial^\alpha F(\mathbf{x}) = \frac{\partial^{|\alpha|} F(\mathbf{x})}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_p^{\alpha_p}} \tag{9}$$

Therefore, the regulation term is:

$$E_c(F) = \frac{1}{2} \sum_{k=0}^{K} \sum_{|a|=k} \int_{R^p} b_k \left[\partial^a F(\mathbf{x})\right]^2 dx_1 dx_2 \cdots dx_p \tag{10}$$

This function is simply the summation of the integrations of the partial derivatives of the approximation function squared. As such, the regularization term is small when the function is smooth because the derivatives tend to be small and vice versa.

For $b_k = \frac{\beta^{2k}}{k!2^k}$ and K approaching infinity, where β is a positive real number, it can be proven that by minimizing the error function, Eq. (3), with respect to the approximation function, the solution of the problem can be written in the following form (Poggio and Girosi, 1990):

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \exp\left(-\frac{1}{2\beta^2}\|\mathbf{x} - \mathbf{x}_i\|^2\right) = \sum_{i=1}^{N} w_i \exp\left(-\sigma\|\mathbf{x} - \mathbf{x}_i\|^2\right) \tag{11}$$

where $\sigma = \frac{1}{2\beta^2}$ and w_i's are real numbers. Eq. (11) is a linear superposition of multivariate Gaussian functions with centers x_i's located at the data points.

The architecture of the regularization network is shown schematically in Figure 2. It consists of an input layer, a hidden layer, and an output layer. The number of nodes in the input layer is equal to p, the number of input attributes. The number of nodes in the hidden layer is equal to N, the number of training examples. The output node gives the estimated construction cost.

The network shown in Figure 2 is a feedforward network. The nodes in the hidden layer represent the nonlinear multivariate Gaussian activation functions $G_i(\mathbf{x}) = \exp\left[-\sigma\|\mathbf{x} - \mathbf{x}_i\|^2\right]$. In other words, the output of the ith node in the hidden layer is $G_i(\mathbf{x}) = \exp\left[-\sigma\|\mathbf{x} - \mathbf{x}_i\|^2\right]$. The input and hidden layers are fully connected. That means every node in the hidden layer receives inputs from all the nodes in the input layer. The

links connecting the hidden layer to the output layer represent the weights w_i's in the approximation function, Eq. (11).

The learning process of the regularization network consists of two steps. In the first step, the value of the parameter σ in Eq. (11) is found by a cross-validation procedure to be described in Section 5. The smoothness of the approximation function is primarily controlled by this parameter. The smaller the value of σ, the smoother the approximation function will be. We will call σ the smoothing parameter. In the second step, w_i's are found using the method described in next section.

4. DETERMINATION OF WEIGHTS OF THE REGULARIZATION NETWORK

The smoothing parameter σ and the weights w_i's depend on each other in a complicated way and consequently must be calculated iteratively. In this section, a method is presented for finding the weights w_i's. Defining the following matrices

$$\mathbf{d} = [d_1, d_2, \ldots, d_N]^T \tag{12}$$

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{x}_1;\mathbf{x}_1) & G(\mathbf{x}_1;\mathbf{x}_2) & \ldots & G(\mathbf{x}_1;\mathbf{x}_N) \\ G(\mathbf{x}_2;\mathbf{x}_1) & G(\mathbf{x}_2;\mathbf{x}_2) & \ldots & G(\mathbf{x}_2;\mathbf{x}_N) \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ G(\mathbf{x}_N;\mathbf{x}_1) & G(\mathbf{x}_N;\mathbf{x}_2) & \ldots & G(\mathbf{x}_N;\mathbf{x}_N) \end{bmatrix} \tag{13}$$

$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T \tag{14}$$

where $G(\mathbf{x}_i;\mathbf{x}_j)=\exp\left[-\sigma\|\mathbf{x}_i-\mathbf{x}_j\|^2\right]$, it can be shown that the solution of the regularization problem, i.e., the weights w_i's, satisfies the following equation (Haykin, 1994):

$$(\mathbf{G}+\mathbf{I})\mathbf{w}=\mathbf{d} \tag{15}$$

where $\mathbf{I}$ is the $N\times N$ identity matrix. If the matrix $(\mathbf{G}+\mathbf{I})$ is not ill-conditioned, the solution of the linear equations represented by Eq. (15) can be solved by linear equation solvers such as the Gauss-Jordan elimination or LU decomposition method. The $N\times N$ matrix $(\mathbf{G}+\mathbf{I})$, however, is large and usually suffers from numerical ill-conditioning. This leads to zero pivot in the Gauss-Jordan elimination method resulting in large errors in the solution. The aforementioned approach was in fact employed in this research but without any success. The elements of the optimum vector $\mathbf{w}$ were found to be very large due to numerical instability.

To overcome the numerical ill-conditioning problem, a *singular value decomposition* method is used to find the weights w_i's (Press et al., 1988). In this approach, the matrix $(\mathbf{G}+\mathbf{I})$ is first decomposed as

$$\mathbf{G}+\mathbf{I}=\mathbf{U}\mathbf{C}\mathbf{V}^{\mathrm{T}} \tag{16}$$

where $\mathbf{U}$ and $\mathbf{V}$ are $N\times N$ orthonormal matrices (i.e., $\mathbf{U}\mathbf{U}^{\mathrm{T}}=\mathbf{I}$ and $\mathbf{V}\mathbf{V}^{\mathrm{T}}=\mathbf{I}$) and $\mathbf{C}$ is an $N\times N$ diagonal matrix with diagonal entries c_i's, called singular values, where $|c_1|\geq|c_2|\geq\cdots\geq|c_N|$.

Having found the matrices, $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{C}$, the weights vector can be found from

$$\mathbf{w} = \sum_{i=1}^{J} \left(\frac{\mathbf{U}_{(i)} \cdot \mathbf{d}}{c_i} \right) \cdot \mathbf{V}_{(i)} \tag{17}$$

where $\mathbf{U}_{(i)}$, i = 1, ..., N, denotes the ith column of U and $\mathbf{V}_{(i)}$, i = 1, ..., N, denotes the ith column of V. In Eq. (17), the summation is performed over J terms and not the entire N terms in order to avoid numerical ill-conditioning due to division by very small numbers and the truncation error. The terms c_i's in the denominator of Eq. (17) can not take very small values. All of the singular values used in the Eq. (17) are greater than $\varepsilon = N\varepsilon_m |c_1|$, where ε_m is the machine precision, and c_1 is the largest singular value. By selecting a predetermined value for ε_m the small values of c_i's are excluded from the summation in Eq. (17), and the summation is done over J terms such that $|c_i| > \varepsilon$ for any $i \le J$ and $|c_i| \le \varepsilon$ for any $i > J$.

5. PROPER GENERALIZATION AND ESTIMATION BY CROSS-VALIDATION

For proper solution of the problem and accurate estimation, a trade-off is necessary between minimizing the standard error term, Eq. (4), and smoothness of the approximation function. As mentioned earlier, the smoothness of the approximation function and the generalization properties of the network are influenced by the parameter σ. In order to obtain a proper value for σ a method used in statistical pattern recognition called *cross-validation* is employed in this research (Fukunaga, 1990).

In the cross-validation method, the available set of examples is divided randomly into two sets, a training set $\left(\mathbf{x}_t^n, d_t^n\right)$, n = *1, 2, ..., N_t* and a validation set $\left(\mathbf{x}_v^n, d_v^n\right)$, n = *1,*

2, ..., N_v, where subscripts t and v refer to training and validation sets, respectively, and N_t and N_v are the numbers of training and validation examples, respectively. The network is trained with the training set $\left(\mathbf{x}_t^n, d_t^n\right)$, $n = 1, 2, ..., N_t$, using different values of σ within a given range. This range is problem-dependent and is determined by experience and numerical experimentation.

For each value of σ, the weights w_i's are found using Eqs. (16) and (17) and an average training error is calculated in the following form:

$$E_t = \sqrt{\sum_{n=1}^{N_t}\left[d_t^n - F\left(\mathbf{x}_t^n\right)\right]^2 \Big/ N_t} \quad . \tag{18}$$

Next, using the validation set $\left(\mathbf{x}_v^n, d_v^n\right)$, $n = 1, 2, ..., N_v$, an average validation error is calculated in the following form:

$$E_v = \sqrt{\sum_{n=1}^{N_v}\left[d_v^n - F\left(\mathbf{x}_v^n\right)\right]^2 \Big/ N_v} \quad . \tag{18}$$

Typical trend relationships between the average training and validation errors and the smoothing factor σ are shown conceptually in Figure 3. The average training error always decreases with an increase in the magnitude of σ for a numerically-stable algorithm. In contrast, the average validation error curve does not have a continuously decreasing trend. Rather, one can identify a minimum on this curve. As mentioned earlier in the article, broadly-speaking a large σ indicates overfitting and a small σ indicates underfitting. The σ corresponding to the global minimum point on the validation curve represents the properly-fitted estimation curve. The average validation error gives an estimate of the estimation/prediction error.

6. INPUT AND OUTPUT NORMALIZATION

Because the regularization network uses spherically symmetric multivariate Gaussian activation functions, to improve the performance, the input variables are normalized so that they span similar ranges in the input space. The purpose of the normalization is to ensure that all examples in the training set have similar influence in the learned approximation function. In other words, it is statistically desirable to have variables with zero mean and the same unit standard deviation. This can be achieved by using the following change of variables and normalization procedure (Fukunaga, 1990):

$$\tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{\sigma_i} \tag{19}$$

where $\tilde{x}_i^n$, $n = 1, 2, ..., N$, are the normalized input data, and

$$\bar{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_i^n \qquad \text{for } i = 1, ..., p \tag{20}$$

and

$$\sigma_i^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left(x_i^n - \bar{x}_i\right)^2 \qquad \text{for } i = 1, ..., p \tag{21}$$

are the means and standard deviations of the original set of variables.

The Gaussian activation function is maximum at its center (data point) and approaches zero at large distances from the center. In other words, statistically speaking, the use of the Gaussian activation function amounts to large output near the center (data point) and zero output at large distances from the center where there is no data point. But,

the lack of data point does not necessarily mean the output is zero at large distances from the available sample data points.

One may argue that it is not possible to make an accurate estimate at large distances from the example data points. This is the well-known *extrapolation* problem. While the regularization theory solves the interpolation problem accurately it is not concerned with the extrapolation problem. But, a practical estimation system should not fail at the boundaries of the available data domain abruptly. Consequently, to improve the estimation accuracy at large distances from the available data points, first a linear trend (hyperplane) is found through the example data points by performing a linear regression analysis. Next, the output data are normalized with respect to this hyperplane (the outputs are measured from this plane instead of a zero base hyperplane). Finally, regularization network is applied using the normalized data output. This process will bring the estimates at large distances from the available data points close to the linear trend hyperplane.

Mathematically, the function $\sum_{n=1}^{N}\left(d_i^n - \sum_{i=1}^{p} a_i \tilde{x}_i^n - a_0\right)^2$ is minimized with respect to linear parameters a_i's ($i = 0,1,\ldots, p$) in order to find the linear trend hyperplane. This hyperplane is represented by

$$y = \sum_{i=1}^{p} a_i \tilde{x}_i + a_0 \tag{22}$$

Then, the normalized output data are defined as

$$\tilde{d}_i^n = d_i^n - \sum_{i=1}^{p} a_i \tilde{x}_i^n - a_0 \qquad \text{for } n = 1, 2, \ldots, N. \tag{23}$$

7. APPLICATION

The computational models presented in this article have been implemented in the programming language MATLAB (MathWorks, 1992) and applied to the problem of estimating the cost of concrete pavements. The reason for selection of MATLAB is the availability of a large number of built-in numerical analysis functions such as singular value decomposition.

The data set was collected from the files of previous projects at the Ohio Department of Transportation. It includes 242 examples of construction costs for reinforced concrete pavements. The cost factors used in the examples are the quantity and the dimension (thickness) of the pavement.

Example 1

In this example only the quantity information is used. The variation of the unit cost versus the quantity is presented in Figure 4. Because the reinforced concrete pavement quantity has a large variation the quantity scale in Figure 4 is a logarithmic scale. Figure 4 shows clearly that the highway construction cost data are very noisy.

The training set of 121 examples and the validation set of 121 examples are selected randomly from the available 242 data examples. Variations of the average training and validation errors with respect to the smoothing parameter σ are shown in Figure 5. The trend for both curves is similar to trends discussed in Section 5 and Figure 3. The minimum point on the average validation error curve corresponds to $\sigma = 0.8$ which represents the value needed for the proper generalization. The corresponding average

training and validation errors for the unit cost of the concrete pavement are $\$6.45/m^3$ and $\$7.22/m^3$, respectively. For comparison, the average unit cost of the concrete pavement for the 242 example data is $\$39.2/m^3$. Figures 6 shows the learned curve along with the training and validation data sets.

Example 2

In this example, the quantity and the dimension (thickness of the pavement) are used as input attributes. The unit cost versus the concrete pavement thickness for the 242 example data are shown in Figure 7. Similar to example 1, the data set is divided into 121 training and 121 validation examples. Variations of the average training and validation errors with respect to the smoothing parameter σ are shown in Figure 8. The smoothing parameter corresponding to the minimum point on the average validation error curve is found to be σ=0.05. The corresponding average training and validation errors for the unit cost of the concrete pavement are $\$6.3/m^3$ and $\$6.7/m^3$, respectively. The learned approximation function is a surface in a three-dimensional space. The average validation error for this example is less than that of example 1. As the number of attributes is increased the average validation error decreases which means the construction cost is estimated more accurately.

8. CONCLUSION

In this article, a regularization neural network was presented for estimating the cost of construction projects. The problem is formulated in terms of an error function

consisting of a standard error term and a regularization term. The purpose of the latter term is to compensate for the overfitting problem and to improve the cost estimation outside of the available data points.

The traditional regression analysis methods can fit the data only in certain types of functions such as polynomial functions. Further, a major assumption is made that the data must fit one of these functions. In the regularization neural networks approach presented in this article, on the other hand, no assumption is made about the shape of the approximation function to be learned. The only assumptions made are the continuity and the general smoothness of the function.

The neural networks model presented in this article has the following major advantages over other neural networks algorithms such as the backpropagation (BP) neural networks:

- The regularization neural networks is based on a solid mathematical foundation. This makes the cost estimation model consistently reliable and predictable.
- The result of estimation from the regularization neural network depends only on the training examples. It does not depend on the architecture of the neural network (such as the number of nodes in the hidden layer), the learning parameters (such as the learning and momentum ratios in the BP algorithm), and the number of iterations required for training the system. As such, it can be said the regularization neural network presented in this article is an *objective* cost estimator.
- The problem of noise in the data which is important in the highway construction cost data is taken into account in a rational manner.

The generalization error of the regularization networks can be attributed to insufficient data examples which can be improved by increasing the database of examples from previous construction projects and intrinsic noise due to nonquantifiable and unpredictable factors which is impossible to avoid.

ACKNOWLEDGMENT

This article is based on the research sponsored by Ohio Department of Transportation and Federal Highway Administration.

APPENDIX I. REFERENCES

Adeli, H. and Hung, S.L. (1994), "An Adaptive Conjugate Gradient Learning Algorithm for Efficient Training of Neural Networks", *Applied Mathematics*, Vol. 62, No. 1, pp. 81-102.

Adeli, H. and Hung, S.L. (1995), *Machine Learning — Neural Networks, Genetic Algorithms, and Fuzzy Systems*, John Wiley and Sons, New York, New York.

Al-Gwaiz, M.A. (1992), *Theory of Distributions*, Marcel Dekker, New York.

Bishop, Christopher M. (1995), "Neural Networks for Pattern Recognition", *Clarendon Press, Oxford*, UK.

Chao, L.C. and Skibniewski, M.J. (1994), "Estimating Construction Productivity: Neural-Network-Based Approach", *Journal of Computing in Civil Engineering*, Vol. 8, No. 2, pp. 234-251.

Chao, L.C. and Skibniewski, M.J. (1995), "Neural Networks of Estimating Construction Technology Acceptability", *Journal of Construction Engineering and Management*, Vol. 121, No. 1, pp. 130-142.

Eldin, N.N. and Senouci, A.B. (1995), "A Pavement Condition Rating Model using Backpropagation Neural Networks", *Microcomputers in Civil Engineering*, Vol. 10, No. 6, pp. 433-441.

Fukunaga, K. (1990), *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, Boston, Massachusetts.

Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, Inc., New York.

Hegazy, T., Fazio, P., and Moselhi, O. (1994), "Developing Practical Neural Network Applications using Backpropagation", *Microcomputers in Civil Engineering*, Vol. 9, No. 2, pp. 145-159.

Hung, S.L. and Adeli, H. (1991), "A Model of Perception Learning with a Hidden Layer for Engineering Design", Neurocomputing, Vol. 3, No. 1, pp. 3-14.

Hutchinson, J.M., Lo, A. and Poggio, T., (1994), "A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks", MIT A.I. Memo No. 1471, Cambridge, Massachusetts.

Kartam, N. (1996), "Neural Network - Spreadsheet Integration for Earthmoving Operations, *Microcomputers in Civil Engineering*, Vol. 11, No. 4, pp. 283-288.

MathWorks, Inc. (1992), *MATLAB, high-performance numeric computation and visualization software : user's guide : for UNIX workstations*, MathWorks, Inc., Natick, Massachusetts.

Moselhi, O., Hegazy, T. and Fazio, P. (1991), "Neural Networks as Tools in Construction", *Journal of Construction Engineering and Management*, Vol. 117, No. 4, pp. 606-623.

Moselhi, O., Hegazy, T. and Fazio, P. (1993), "DBID: Analogy-Based DSS for bidding in Construction", *Journal of Construction Engineering and Management*, Vol. 119, No. 3, pp. 466-479.

Narendra, K.S. and Parthasarathy, K. (1990), "Identification and Control of Dynamical System Using Neural Networks", *IEEE Transactions on Neural Networks* 1, 4-27.

Poggio, T. and Girosi, F. (1990), "Network for Approximation and Learning", *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1481-1497.

Pompe, P.P.M. and Feelders, A.J. (1997) "Using machine learning and statistics to predict corporate bankruptcy", *Microcomputers in Civil Engineering*, Vol. 12, No. 4.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. (1988), *Numerical Recipes in C : The Art of Scientific Computing*, Cambridge University Press, New York.

Tikhonov, A.N. and Arsenin, V.Y. (1977), *Solution of Ill-posed Problems*, W.H. Winston, Washington, D.C..

APPENDIX II. NOTATIONS

a_i : parameters obtained from the linear least-squares regression algorithm for output normalization.

$b_k = \frac{\beta^{2k}}{k!2^k}$

C : diagonal matrix in singular value decomposition.

c_i : singular values.

c_1 : the largest singular value.

d : example output vector.

d_i : example output corresponding to input example $\mathbf{x}_i = \left(x_1^i, x_2^i, \cdots, x_p^i\right)$.

$\tilde{d}_i^n$: normalized output data.

e : independent noise function with zero mean.

$E(F)$: error function.

$E_s(F)$: standard error term.

$E_c(F)$: regularization term.

E_v : average validation error.

E_t : average training error.

$F(\boldsymbol{x})$: approximation mapping function.

$g(\mathbf{x})$: nonlinear mapping function mapping a p-dimensional input space R^p to a one-dimensional output space R^1.

G : Gaussian matrix.

$$G_i(\mathbf{x}) = \exp\left[-\sigma\|\mathbf{x} - \mathbf{x}_i\|^2\right]$$

$$G(\mathbf{x}_i;\mathbf{x}_j) = \exp\left[-\sigma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right]$$

$\mathbf{I}$: identity matrix.

N : number of data examples.

N_t : number of training examples.

N_v : number of validation examples.

o_i : computing response corresponding to $\mathbf{x}_i = \left(x_1^i, x_2^i, \cdots, x_p^i\right)$.

P : linear differential operator in the regularization term.

p : dimension o f the input space (number of input attributes).

$\mathbf{U}$: orthonormal matrixes in singular value decomposition.

$\mathbf{U}_{(i)}$: ith column of U.

$\mathbf{V}$: orthonormal matrices in singular value decomposition.

$\mathbf{V}_{(i)}$: ith columns of V.

$\mathbf{w}$: weight vector.

$\mathbf{x}_i = \left(x_1^i, x_2^i, \cdots, x_p^i\right)$: input data.

x_n^i : nth attribute of ith example input data.

$\tilde{x}_i^n$: normalized input data.

$\bar{x}_i$: mean for the ith input attribute.

$\alpha = \left(\alpha_1, \alpha_2, \cdots, \alpha_p\right)$: a sequence of non-negative integers.

$\varepsilon = N\varepsilon_m|c_1|$: threshold value for c_i's.

ε_m : machine precision.

σ : smoothing parameter.

σ_i : standard deviation for the ith input attribute.

CAPTIONS OF FIGURES

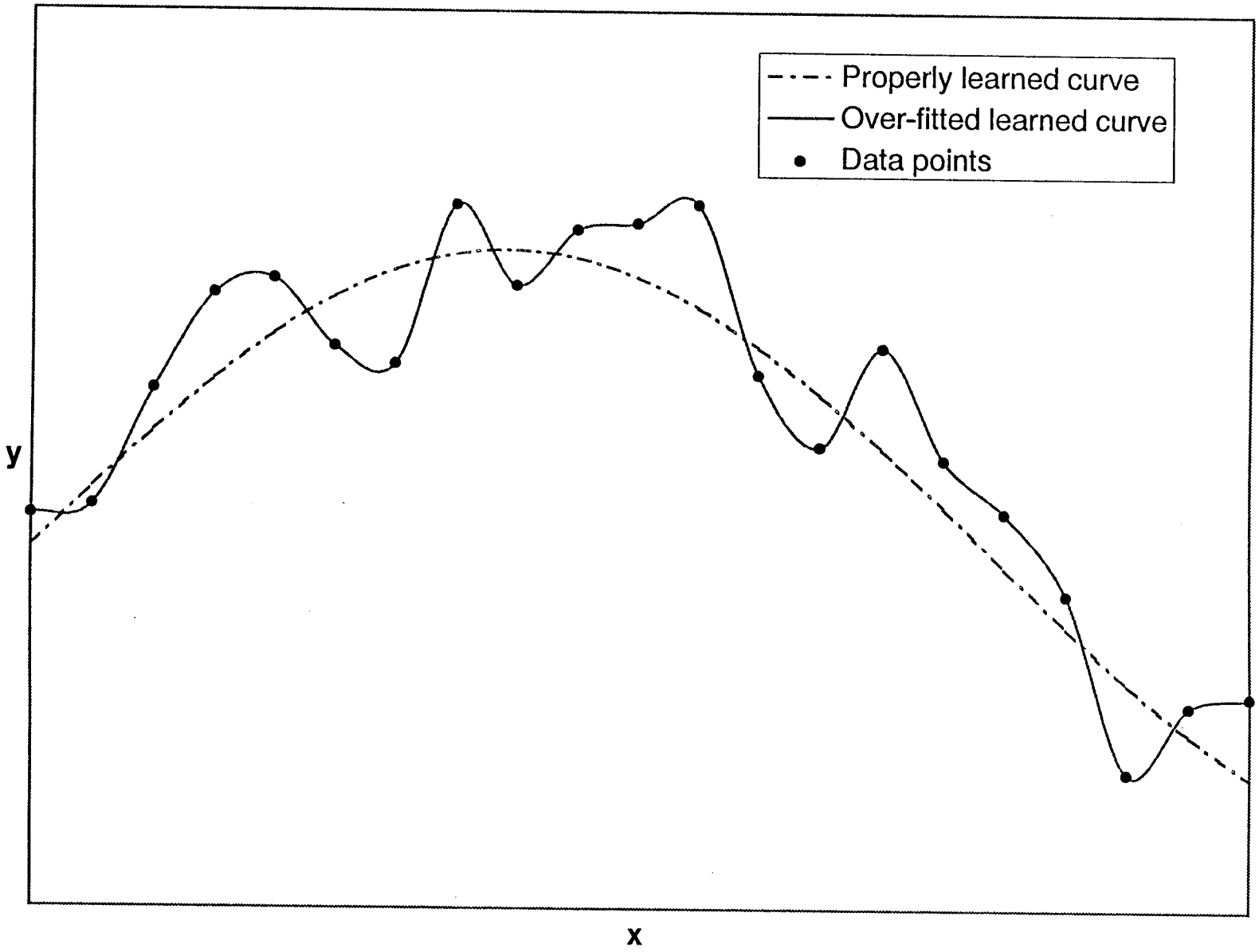

Properly learned curve
Over-fitted learned curve
Data points
y
x

Figure 1

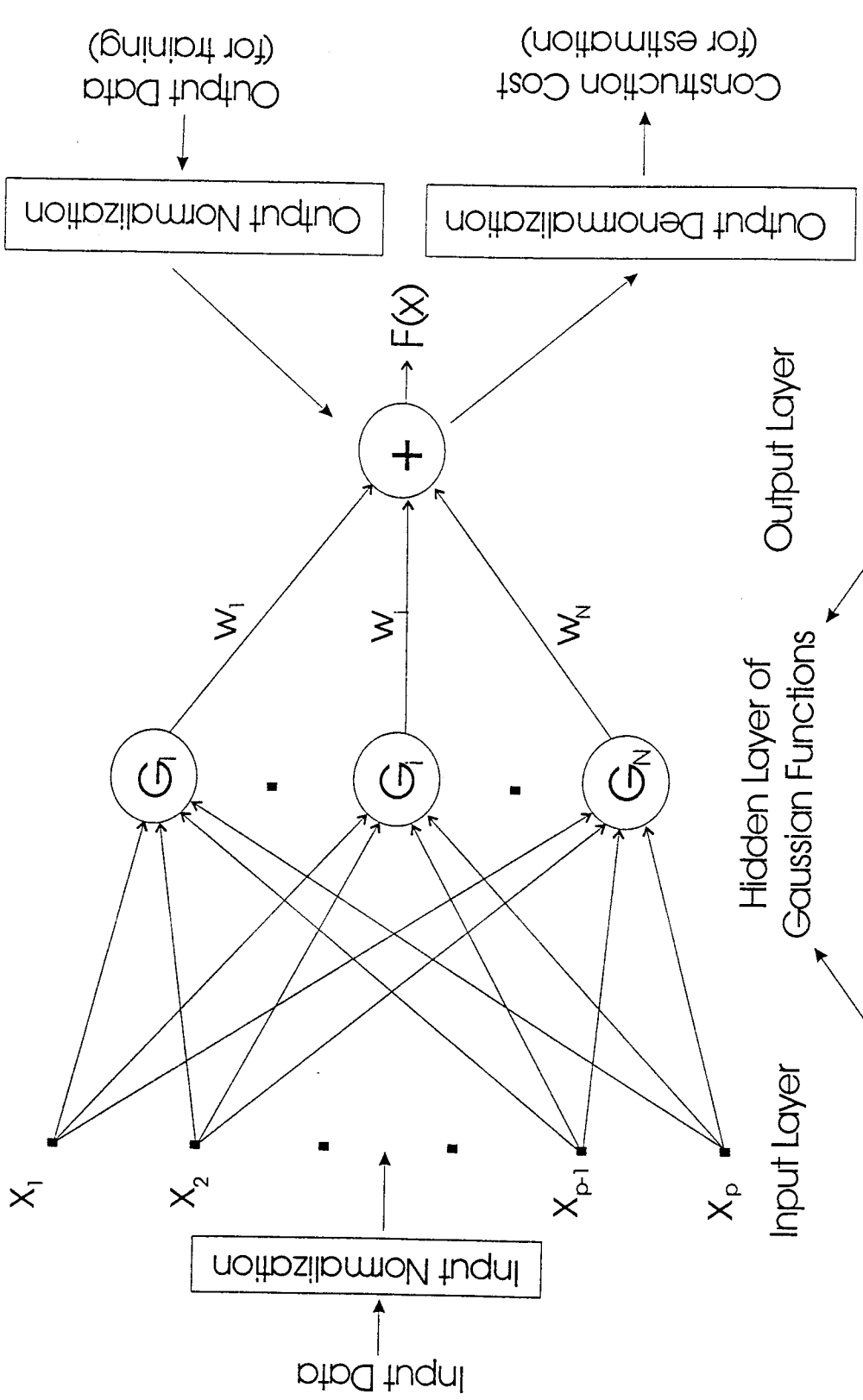

Input Data
Input Normalization
X_1
X_2
X_{p-1}
X_p
Input Layer
Hidden Layer of Gaussian Functions
Output Layer
G_1
G_i
G_N
W_1
W_i
W_N
+
F(X)
Output Denormalization
Output Normalization
Construction Cost (for estimation)
Output Data (for training)
σ
Cross-Validation
σ

Figure 2

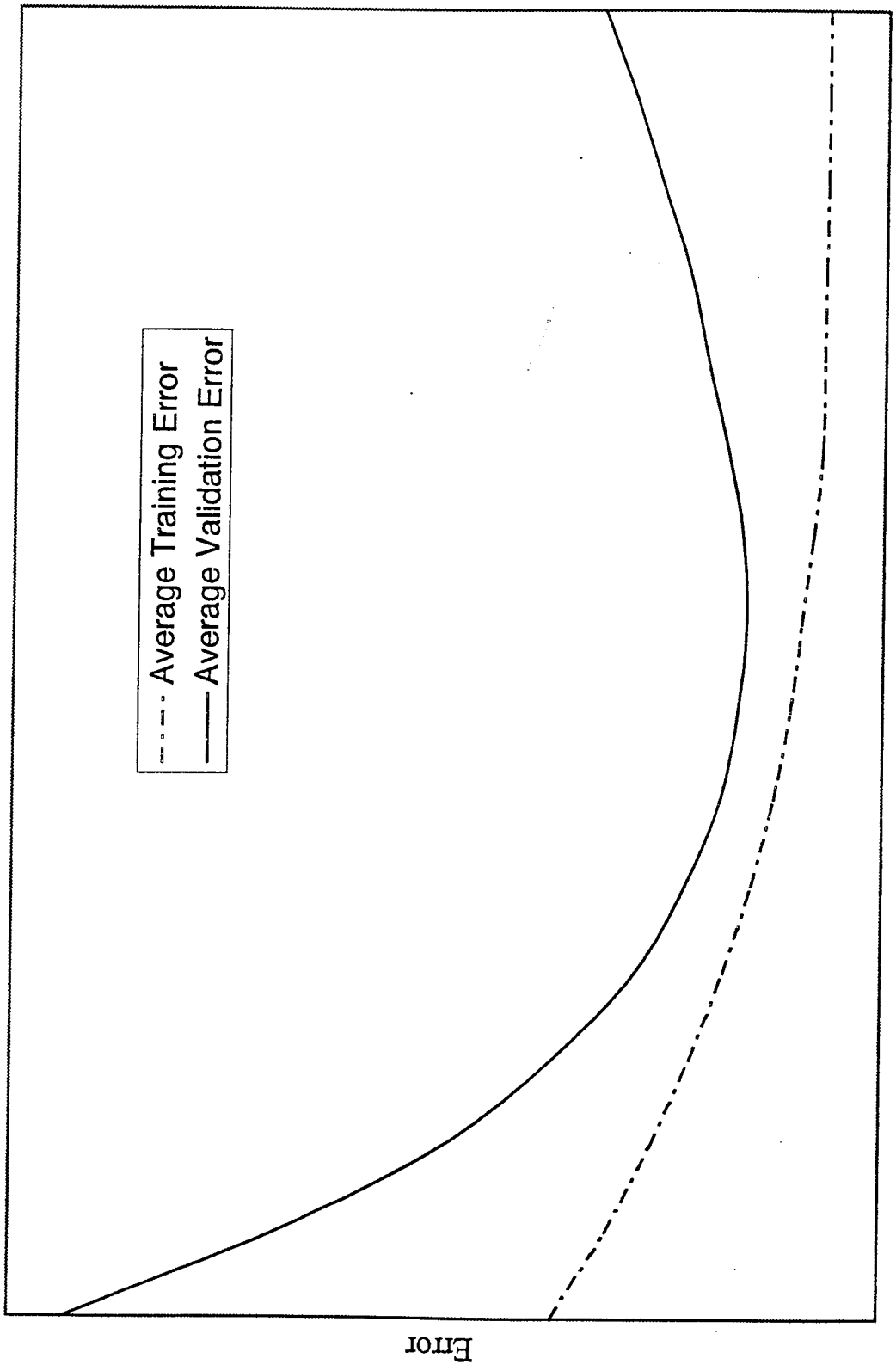


Figure 3

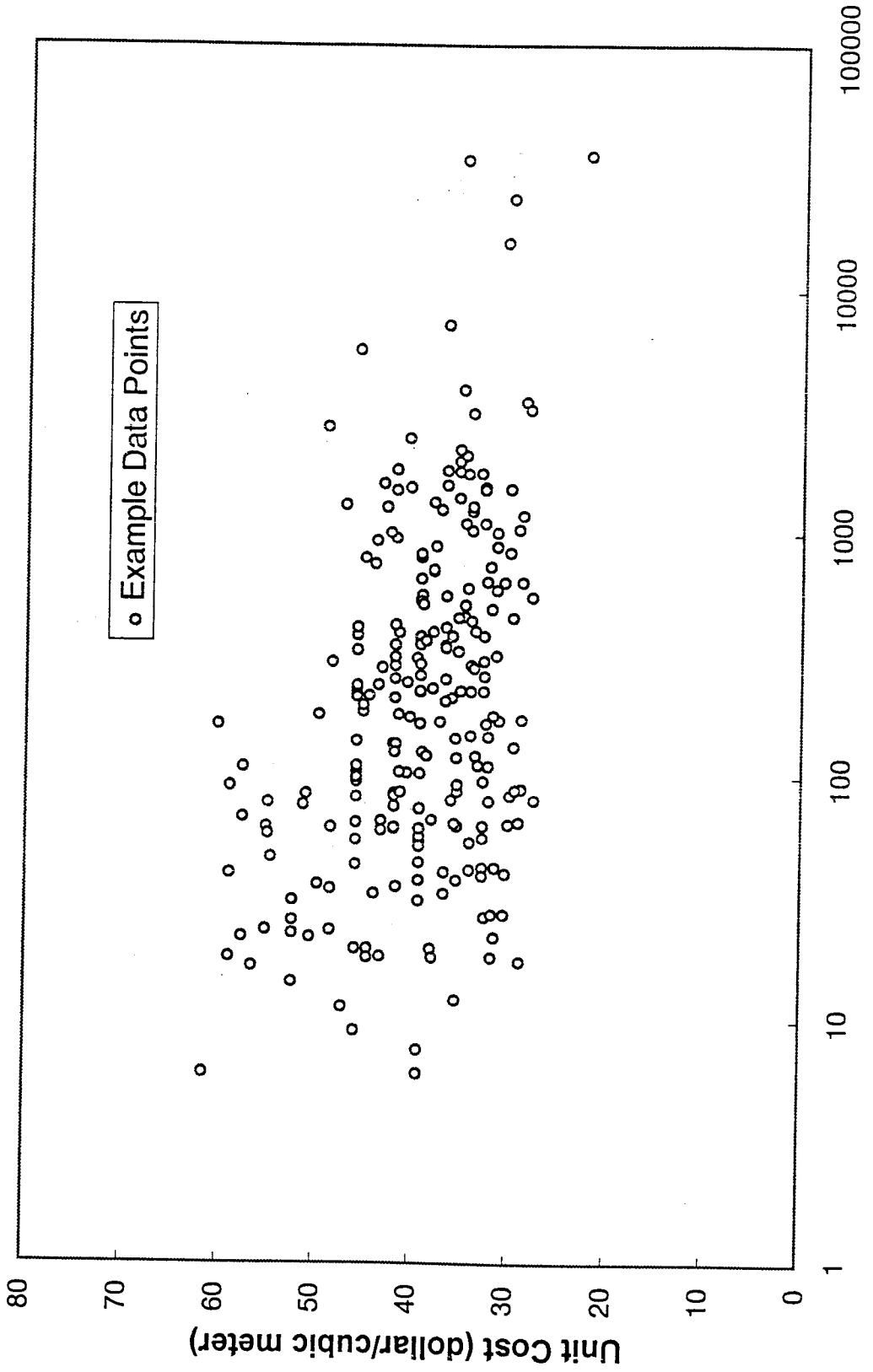

Example Data Points
Unit Cost (dollar/cubic meter)
Quantity (cubic meter)
0
10
20
30
40
50
60
70
80
1
10
100
1000
10000
100000

Figure 4

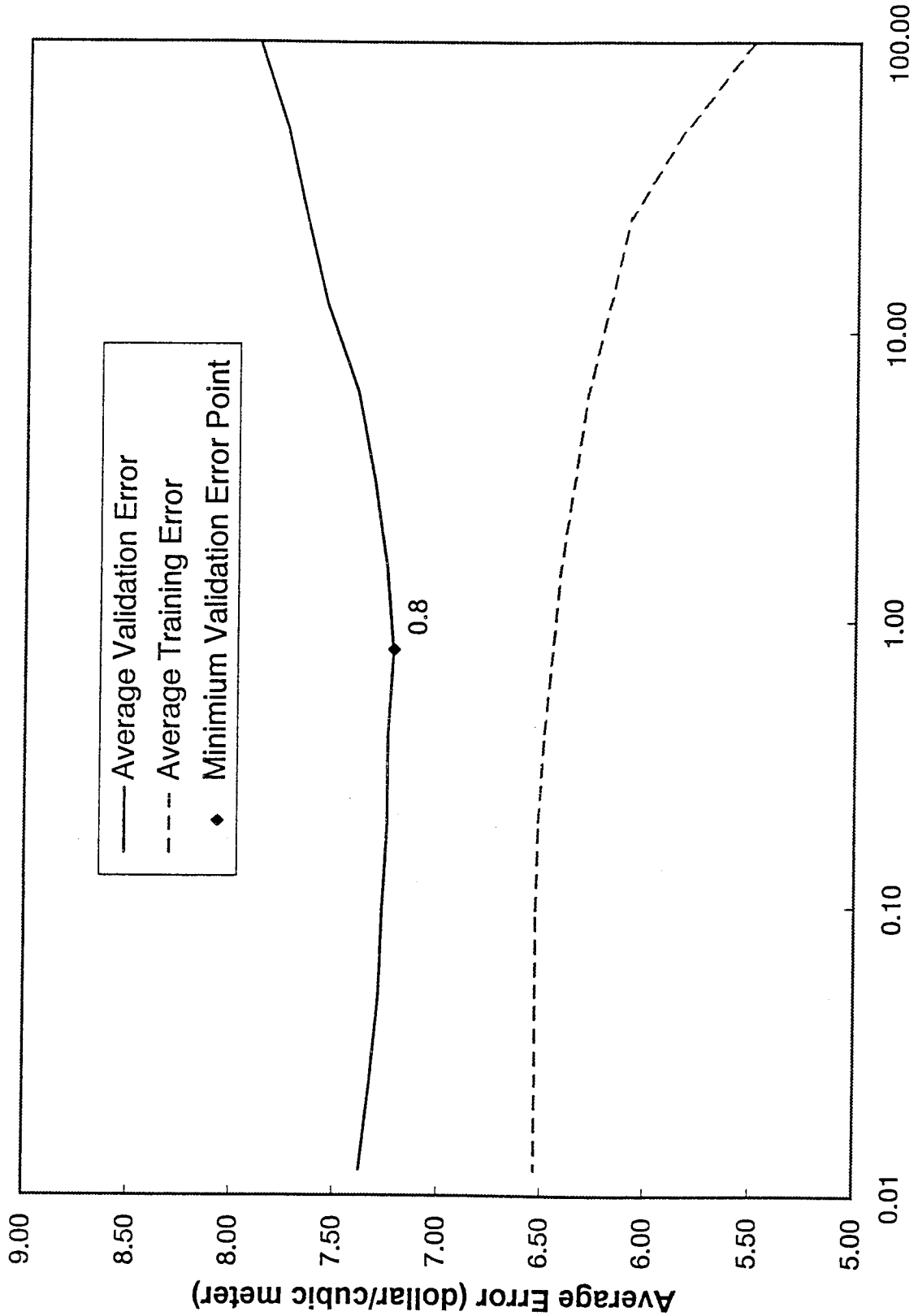

Average Validation Error
Average Training Error
Minimium Validation Error Point
0.8
Average Error (dollar/cubic meter)
9.00
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
0.01
0.10
1.00
10.00
100.00
σ

Figure 5

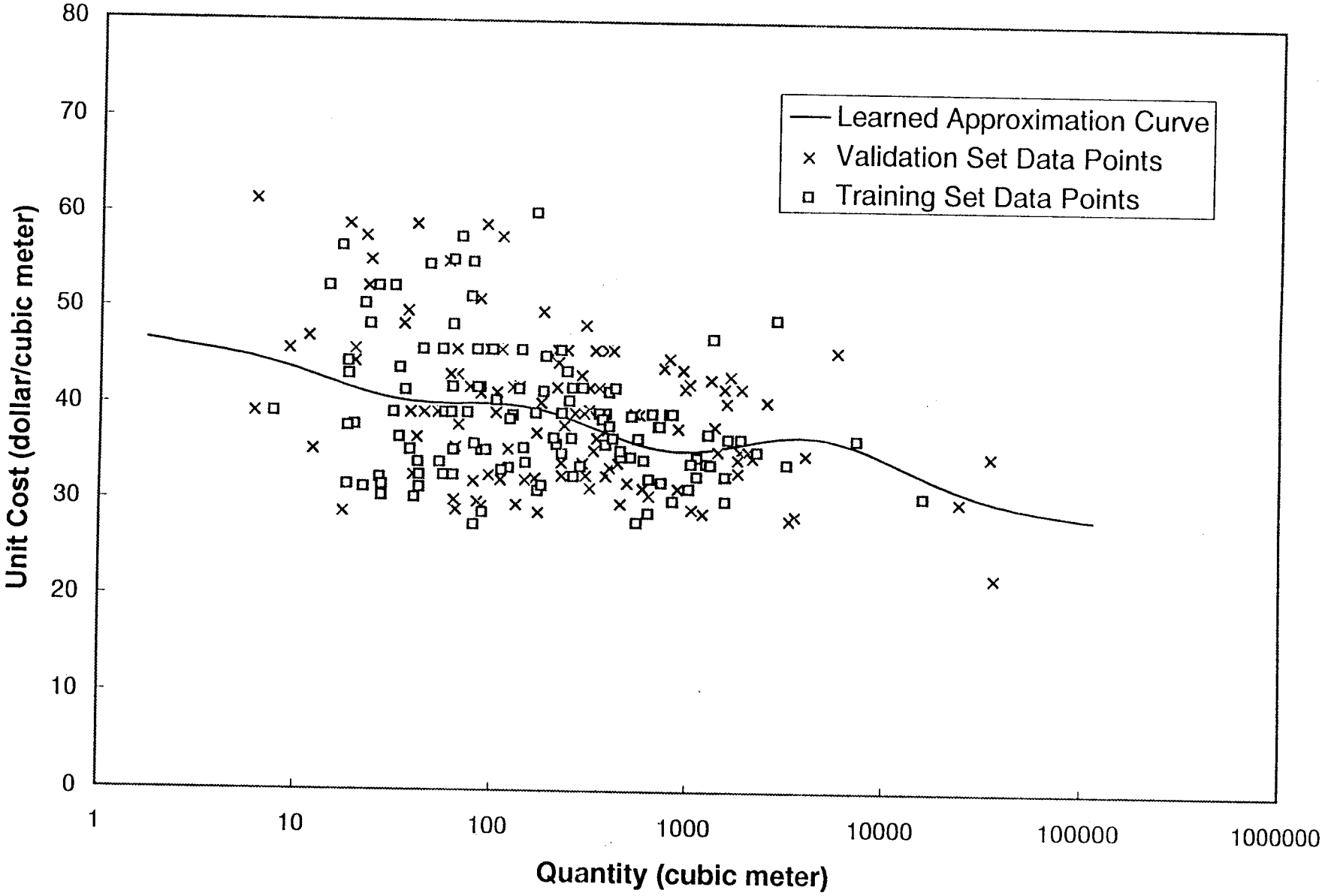

Learned Approximation Curve
Validation Set Data Points
Training Set Data Points
Unit Cost (dollar/cubic meter)
Quantity (cubic meter)
0
10
20
30
40
50
60
70
80
1
10
100
1000
10000
100000
1000000

Figure 6

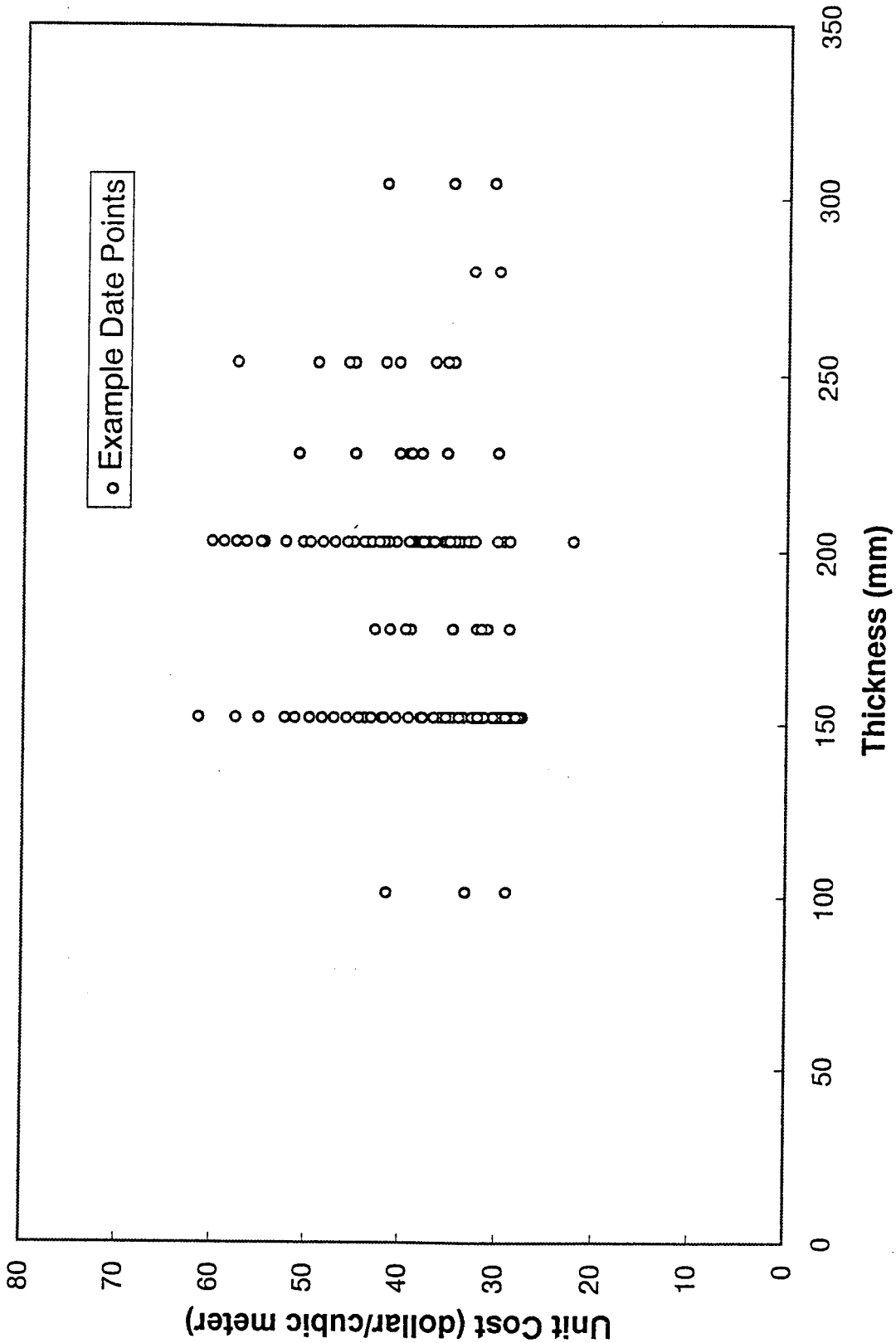

Example Date Points
Unit Cost (dollar/cubic meter)
Thickness (mm)
0
10
20
30
40
50
60
70
80
0
50
100
150
200
250
300
350

Figure 7

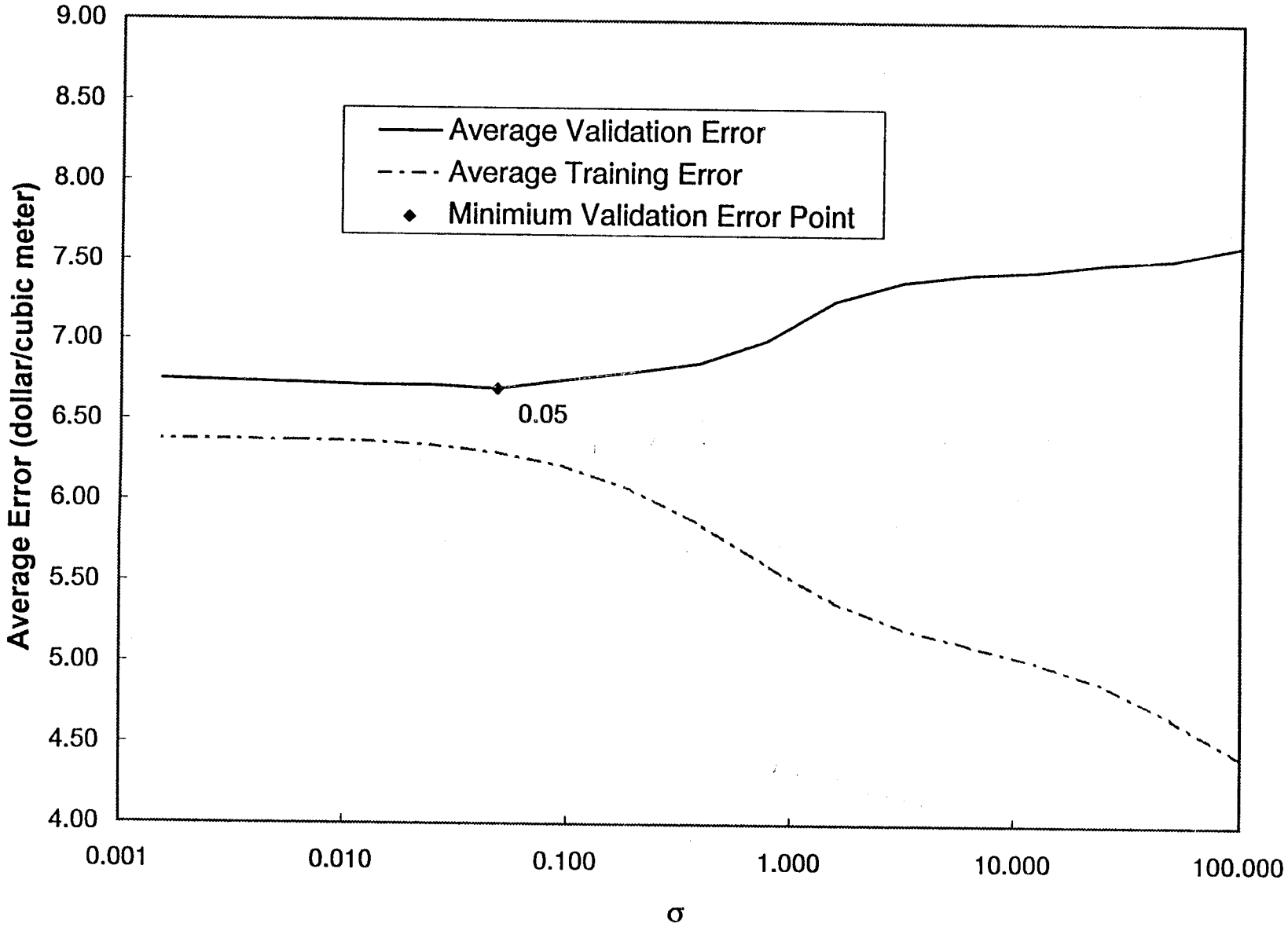

9.00
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
Average Error (dollar/cubic meter)
Average Validation Error
Average Training Error
Minimium Validation Error Point
0.05
0.001
0.010
0.100
1.000
10.000
100.000
σ

Figure 8

OBJECT-ORIENTED INFORMATION MODEL FOR CONSTRUCTION PROJECT MANAGEMENT

Asim Karim[1] and Hojjat Adeli[2], Fellow, ASCE

ABSTRACT: Recently, the authors developed a general and powerful mathematical model for scheduling of the construction projects. An optimization formulation was presented with the goal of minimizing the direct construction cost. The nonlinear optimization problem was solved by the recently patented neural dynamics model of Adeli and Park. In this article an object-oriented (OO) information model is presented for construction scheduling, cost optimization, and change order management based on the new construction scheduling model. The goal is to lay the foundation for a new generation of flexible, powerful, maintainable, and reusable software system for the solution of construction scheduling problem. The model is presented as a domain-specific development *framework* using the Microsoft Foundation Class (MFC) library and utilizing the software reuse feature of the *framework*. The *framework* reuse architecture is more flexible and powerful than other reuse techniques such as *components* and *patterns*. An accompanying article presents the implementation of the OO information model in a prototype software system for management of construction projects, called CONSCOM.

[1] Graduate Research Associate, Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University.
[2] Professor, Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 470 Hitchcock Hall, 2070 Neil Avenue, Columbus, OH 43210, USA.

INTRODUCTION

Change in construction projects after bid are a common occurrence. Change orders often lead to disputes between the owner and the contractor. These disputes are usually on the issue of compensation for change. The lack of a consistent method of change order management can result in long delays and costly legal battles.

Broadly, change in a construction project after bid may be required because of the following reasons (Saunders, 1996):

- An error or inconsistency in the original specification or design
- A change in the functional requirements of the project after bid
- An unforeseen or unknown condition requiring a change in the original construction method and/or design.

Computer support for change order management, as reported in the literature, is minimal. Leymeister et al. (1993) describe the use of databases to analyze and process contractors' direct cost payment requests. Their goal is to reduce the possibility of errors, omissions, and duplication of claims made by the contractor. No support is provided for determining the validity of the claims and/or their impact on the project. De Leon and Knoke (1995) present a probabilistic procedure based on the critical path method (CPM) to determine the extension needed for contract time as a result of changes. Impact of changes on cost is not considered. Other recent publications on construction change order management includes a survey of construction cost markups paid on direct cost as a result of changes (Saunders, 1996) and a study of changes and their impact based on the survey of private sector projects (Ibbs, 1997). These papers show the lack of a consistent and

sufficiently formal procedure for handling change orders in construction. In most cases the change orders are handled somewhat arbitrarily.

The project schedule is the most important piece of information through which progress can be monitored, change order effects studied, and owner/contractor conflicts resolved. The contractor submits a schedule periodically to the owner as a statement of the state of the work. A schedule is also submitted when a contractor claims a change order. The owner has to check these schedules for correctness and approve them. A powerful scheduling tool that does not restrict schedule-modeling options is desirable for this process.

Recently, Adeli and Karim (1997) developed a general and powerful mathematical model for scheduling of the construction projects. The model can handle various conditions such as repetitive and non-repetitive tasks, work continuity considerations, multiple-crew strategies, and the effects of varying job conditions on the performance of a crew. They presented an optimization formulation for the construction project scheduling problem with the goal of minimizing the direct construction cost. The nonlinear optimization problem is then solved by the recently patented neural dynamics model of Adeli and Park (Adeli and Park, 1996; Park and Adeli, 1997a,b). For any given construction duration, the model yields the optimum construction schedule for minimum construction cost automatically. By varying the construction duration, one can solve the cost-duration trade-off problem and obtain the global optimum schedule and the corresponding minimum construction cost.

In this paper an object-oriented information model is presented for construction scheduling, cost optimization, and change order management based on the new construction scheduling model of Adeli and Karim (1997). The model can be used by the

owner/client who has to approve any change order requests made by the contractor as well as by the contractor. The model provides support for schedule generation and review, cost estimation, and cost-time trade-off analysis. The model implemented in a software system can be used as an intelligent decision support system to resolve change order conflicts.

OWNER 'S ROLE IN CONSTRUCTION PROJECT MANAGEMENT

The majority of major construction projects are awarded by government agencies such as state Departments of Transportation (DOT). These agencies are involved throughout the life-cycle of a project from the conception of the project to its maintenance and operation. The owner's role from project conception to final award of the contract and the phases involved are shown schematically in Figure 1. The only significant change that has to be handled in this stage is that of scope change after work has started on the detailed design and planning phases. Such changes can be handled relatively easily and without significant additional costs.

After the contract is awarded, the owner is responsible for progress monitoring, schedule reviews, change order management, and conflict management. Any change occurring during this stage of the project life-cycle is bound to have a profound impact on the project cost and duration. Furthermore, any such change has a potential to create a conflict between the contractor and the owner on issues such as the compensation for change. The major phases in change order management from the owner's perspective are presented in Figure 2.

The owner-contractor interaction during the change order involves a lot of information exchange. The owner has to analyze this information carefully and make

intelligent decisions in a timely manner. An owner, such as a state DOT, has to protect its interests while at the same time be fair to the contractor. A flexible computer support system for change order can provide efficiency, consistency, reliability, and credibility to the decision making process.

OBJECT ORIENTED METHODOLOGY AND CONSTRUCTION ENGINEERING

Use of the object technology has gained increasing popularity in development of flexible, maintainable, and reusable software systems (Yu and Adeli, 1991, 1993; Adeli and Yu, 1993a,b; Adeli and Kao, 1996; Kao and Adeli, 1997). The basic concept of object-orientation is the object that abstracts a real-world entity by encapsulating its characteristics (data and functionality). An object provides an interface for communication with other objects. Constructs such as inheritance and polymorphism allow easy extension and reusability of previously developed objects. The object-oriented methodology provides an information processing paradigm for efficient development and management of complicated software systems.

Construction scheduling techniques have evolved slowly over the past decades. The critical path method (CPM), developed in the late 50's, is still used despite its shortcomings (Adeli and Karim, 1997). CPM is an easy technique but is based on many simplifying assumptions in modeling construction projects. With the recent advances in computer and information technology, however, more advanced and powerful techniques can be used to provide more accurate and realistic results efficiently. The newly-developed scheduling/cost optimization model of Adeli and Karim (1997) is both more

general and flexible and provides the mathematical foundations for development of a new generation of construction scheduling software systems. The computational model is compatible with CPM but provides additional features like multiple-crew support, separate constructs for repetitive and non-repetitive tasks, time and space buffer constraints, and the capability to model distance/locations of tasks in scheduling. Further, it provides a robust cost optimization capability that can handle both linear and nonlinear cost-duration relationships. Cost optimization of construction projects reported in the literature is limited to CPM or CPM-like models of the project (Liu et al., 1995, Feng et al., 1997). Such models can not model various construction projects such as highway construction accurately.

Use of the object-oriented methodology in computer-integrated construction (CIC) has been reported in the recent literature. However, most of this research is on developing standard project information models to support concurrent engineering in the architecture, engineering, and construction (AEC) industry (Fischer and Froese, 1996; Froese, 1996; Stumpf et al., 1996). A common characteristic of these models is their emphasis on domain modeling with little discussion on computational modeling. Froese and Paulson (1994) present an object model-based project information system as a standard information model for the AEC industry. System integration is achieved by a shared object database. A construction scheduling application module is tested as an example. Fischer and Aalami (1996) advocate the use of construction method knowledge in the generation of construction schedules. They describe object models of the construction methods that can dynamically link construction design and scheduling information.

The object-oriented information model developed in this research for construction scheduling, cost optimization, and change order management is implemented as an application development *framework* in Visual C++, called CONSCOM. The use of *framework* allows software design to be encoded in a reusable format for rapid development of compatible software systems. As an example, an intelligent decision making tool is developed that can be used by the owner in its dealings with a contractor. Such a model can also be an important part of a concurrent engineering model for the AEC industry.

AN OBJECT-BASED INFORMATION MODEL FOR CONSTRUCTION SCHEDULING, COST OPTIMIZATION AND CHANGE MANAGEMENT

How do we arrange objects to solve a particular problem keeping in mind the goals of effective reusability, maintainability, and extensibility for complex software systems? A number of different software designs and architectures have been proposed for this fundamental software engineering problem. Broadly speaking, software design involves developing architectures, techniques, and strategies for data handling and manipulation, user-interface design, and program control. The software design decision may be a major one like whether to use a relational or an OO database management system, or it may be minor like what data type to use for a particular kind of data. A software design approach with attractive aforementioned characteristics is the *framework*-based design.

A framework is a collection of interacting and cooperating software components for solving a generic category of tasks such as database management or user interface design

and/or a domain-specific task such as construction scheduling. In an OOP environment, the components can be connected through various constructs such as object references or pointers. Software design decisions such as the construct to use to allow cooperation between two components are captured by a framework in a reusable format. In addition to the advantages gained by software reuse in general such as reduction in development time and cost and increase in software reliability, encapsulating an OO model in a framework leads to the following benefits:

- The OO model does not have to be reinterpreted and coded whenever a new software application in the domain is developed.
- The use of frameworks leads to compatible software systems because each shares the same underlying software design.

Software Reuse Techniques: Components, Design Patterns, and Frameworks

Reuse is a major concern in creating significant software systems. Effective reuse techniques can reduce development and maintenance costs substantially. Reuse in software engineering is not limited to code only. But rather, reuse encompasses information and knowledge that has been gained over time including software development methods and designs that have been tested and proven efficient (Jacobson et al., 1992). Object-oriented technology provides a reusable software environment that can take advantage of both design and code reuse. This is a primary advantage of the OO technology.

The most common form of reuse is through the use of *components*. Components can be regarded as software building blocks. As such, they raise the level of abstraction for the developer who no longer has to worry about the low level implementation details.

Examples of components are subroutines in FORTRAN and classes in C++. Software development using components involves coding the application architecture and control and plugging in components whenever an appropriate one is available (Figure 3a). This approach to software reuse has a number of limitations including: (1) Components only reuse code; (2) components are not very flexible, that is, they are good at performing one particular task only and can not be customized; and (3) components require a detailed and problem-specific archiving plan. Even though components can be developed in an OOP language their use requires a bottom-up approach, which is not desirable for object-oriented software development. The concept of component reuse is shown in Figure 3a. A major amount of programming is still required for application development including lower level details; components just fill in for specialized tasks.

A *pattern* is a reusable software design (Gamma et al., 1995). Patterns abstract commonly used tried and tested approaches in a descriptive format. The description includes the problem statement, the motivation for and the intent of the solution, the solution, the context in which the solution works, cost/benefit information, and important implementation details and tips. The solution is usually presented in a graphical manner that shows the static and dynamic relationships between the classes involved. A pattern usually provides descriptive information. Whenever a pattern is used it has to be coded in a programming language (Figure 3b). But this, in turn, makes patterns platform independent and widely applicable. Few collections of patterns are available in the current literature. Gamma et al. (1995) catalog general-purpose commonly recurring software design patterns in software systems. More recently, Fowler (1997) surveys software design patterns in the business, finance, and health care industry software

systems. When widely recognized and adopted design patterns can improve software reliability and facilitate design communication and understanding.

Framework, an important concept in object-oriented reuse technology (Johnson, 1997; Demeyer et al., 1997; Rogers, 1997), consists of a set of inter-related abstract and concrete classes that form the skeleton of an application in a particular domain. A framework reuses software designs because it abstracts patterns and other software design decisions. But unlike patterns, frameworks are expressed in code. They are therefore simple or partially complete applications with common functionality and built-in control. Software development using frameworks involves customizing the frameworks for the particular application. One way to do this is by providing implementations for the abstract classes (Figure 3c). A software application may make use of more than one framework. The use of multiple frameworks may be required when, for example, one framework depends on another for its working. Dependencies are usually specified in the overall application architecture.

The framework approach to reuse in software engineering is more flexible and extensible and hence more powerful than reuse through components. It also requires lesser amount of coding because the common lower level details are already implemented in the framework. Further, the use of frameworks results in compatible, manageable, and extensible software systems. A significant distinction between application development with components and that with frameworks is *inversion of control* (Gamma et al., 1995). In a framework-based software system, the developer writes code that is called by the framework. On the other hand, the developer is responsible for calls to components in a component-based approach. On the downside, practical framework design is iterative and

time consuming. Its use also requires good documentation and experience on the part of the developer (Schmidt and Fayad, 1997). However, these limitations are not unique to frameworks but apply to all OO reuse techniques. Fichman and Kemerer (1997) present case studies of OO reuse problems and lessons learned. Typical examples of frameworks for generic tasks are the C++ Standard Template Library (STL) and the Microsoft Foundation Class (MFC) library (Stepanov and Lee, 1995; Microsoft, 1997).

No research has been reported in the literature on the development and use of domain-specific software frameworks in civil engineering.

Development Environment

A construction management application development framework is presented using Microsoft Foundation Class (MFC) library (Microsoft, 1997). For object-oriented programming in the Windows environment MFC is fast becoming the standard for software development. MFC is a general-purpose application development framework. The set of classes it provides abstracts the design and functionality of a simple Windows application. An actual application is built by reusing and adding to MFC. MFC provides basic application software design and control support, user-interface support, and limited file storage/retrieval support. Further, MFC provides general support classes for data handling and manipulation, and foundation classes for database management. A simplified class diagram of the MFC library is shown in Figure 4. Note that in the MFC convention, class names start with a capital C. The notation in this and figure is based on the new standard Unified Modeling Language (UML) (Fowler and Scott, 1997). Classes identified in Figure 4 are defined in Appendix I.

The MFC library supports the single document, multiple view concept of application design and control. The basic idea in this OO concept is the separation of the application data (the document) from its visual presentation (the view). A document object handles data processing, manipulation, storage, and integrity while data presentation and user feedback control are managed by the view objects. This is implemented by objects of classes derived from *CDocTemplate*, *CDocument*, and *CView* (Figure 4). These are abstract classes and cannot be instantiated. Application program control is provided by means of message maps which route Windows and user-generated messages to the appropriate handling functions. For a class to be able to receive messages it must be derived from *CCmdTarget* (Figure 4). This is also an abstract or virtual class. It defines the interface for message processing. Each application must have a single global instance of the class *CWinApp* (Figure 4). This class encapsulates the execution of an application.

The user interface support classes abstract most of the common window elements and their functionality. Classes in this category include those for frame windows, views, dialog boxes, and control bars displayed on the viewing screen. The root of all the window classes is *CWnd* (Figure 4). This class gives the common functionality to all windows like dragging and resizing characteristics. To draw graphics on an output device like a monitor screen MFC provides a device context abstraction in the class *CDC* and its derivatives (Figure 4). These classes provide a common interface for output that is independent of the type of device. By writing to the interface the output can be directed to either a monitor or a printer, for example.

Other classes in the MFC library include those for file services like *CFile* and *CArchive*, exception handling like *CException*, simple data values like *CString* and *CPoint*, data collections like *CArray* and *CList*, and database support like *CDatabase* and *CDaoDatabase* (Figure 4).

A majority of classes in the MFC library are derived directly or indirectly form class *CObject*. This virtual class provides support for a number of capabilities for its derived classes including debugging and file input/output. Generally, application software classes are also derived from the class *CObject* either directly or indirectly.

The MFC library is an abstraction of the Windows application development interface (API). The Windows API is basically a non-OOP C-language interface that provides nearly one thousand functions, messages, data structures, and data types needed to program in the Windows environment. MFC encapsulates the most common functionality of the API into an object-oriented interface and gives a more logical and conceptual view to the API. However, it still retains the power and flexibility of the underlying Windows API and makes it available to the software engineer. Development of a complex software system using MFC library requires some effort. This is because: (1) MFC is a framework of classes that interact with one another. Understanding this interaction is very important as an application program has to "hook" into the framework, (2) MFC is rather large with about 200 classes, and (3) MFC provides an object-oriented encapsulation only for the most simple cases. In more complex situations, the software engineer has to develop an extension using the Windows API (Shepherd and Wingo, 1996).

FINAL COMMENTS

The OO information model and ideas presented in this article have been implemented in a prototype software system for management of construction projects, called CONSCOM, which is described in an accompanying article (Karim and Adeli, 1998).

ACKNOWLEDGMENT

This manuscript is based on a research project sponsored by the Ohio Department of Transportation and Federal Highway Administration.

APPENDIX I. BRIEF DESCRIPTION OF CLASSES IN FIGURE 4

CArchive: Provides data streaming support for input and output of objects from a permanent storage.

CArray: Template-based array data structure.

CCmdTarget: Provides user- and system-generated message handling support.

CDatabase: Provides basic database support.

CDC: Abstracts a device-context such as an output screen or a printer.

CDialog: Abstracts a dialog box display in the Windows environment.

CDocTemplate: Manages document and screen views in an application.

CFrameWnd: Abstracts a frame window display in the Windows environment.

CGdiObject: Base class for graphic output objects such as brushes and pens.

CList: Template-based list data structure (a collection in which each element maintains a pointer to its previous and next element).

CMap: Template-based map data structure (a collection in which each element is identified by a unique identification string).

CMenu: Abstracts a menu display in the Windows environment.

CObject: Provides support for debugging, diagnostics, dynamic object identification, and serialization (object storage and retrieval).

CPen: Abstracts a pen object (for drawing) for output purposes.

CRect: Encapsulates the coordinates of a rectangle

CString: Provides support for managing strings (collection of characters).

CView: Manages the display of the data on the viewing screen.

CWinApp: Encapsulates the execution of a single-threaded Windows application.

CWinThread: Encapsulates a single program thread (process in the operating system) in the Windows environment.

CWnd: The base class for all window elements (Provides common display functionality).

APPENDIX II. REFERENCES

Adeli, H. and Kao, W.-M. (1996), "Object-Oriented Blackboard Models for Integrated Design of Steel Structures", *Computers and Structures*, Vol. 61, No. 3, pp. 545-561.

Adeli, H. and Karim, A. (1997), "Scheduling/Cost Optimization and Neural Dynamics Model for Construction," *Journal of Construction Engineering and Management*, ASCE, Vol. 123, No. 4, pp. 450-458.

Adeli, H. and Park, H.S. (1996), "Hybrid CPN-Neural Dynamics Model for Discrete Optimization of Steel Structures", *Microcomputers in Civil Engineering*, Vol. 11, No. 5, pp. 355-366.

Adeli, H. and Yu, G. (1993a), "An Object-Oriented Data Management Model for Numerical Analysis in Computer-Aided Engineering", *Microcomputers in Civil Engineering*, Vol. 8, No. 3, pp. 199-209.

Adeli, H. and Yu, G. (1993b), "A Concurrent OOP Model for Computer-Aided Engineering using Blackboard Architecture", *Journal of Parallel Algorithms and Applications*, Vol. 1, No. 2, pp. 315-337.

Baumer, D., Gryczan, G., Knoll, R., Lilienthal, C., Riehle, D. and Zullighoven, H. (1997), "Framework Development for Large Systems," *Communications of the ACM*, Vol. 40, No. 10, pp. 52-59.

De Leon, G. P. and Knoke, J. R. (1995), "Probabilistic Analysis of Claims for Extensions in the Contract Time," *Computing in Civil Engineering*, ASCE, New York, NY. Vol. 2, pp. 1513-1520.

Demeyer, S., Meijler, T. D., Nierstrasz, O. and Steyaert, P. (1997), "Design Guidelines for 'Tailorable' Frameworks," *Communications of the ACM*, Vol. 40, No. 10, pp. 60-64.

Feng, C. -W., Liu, L. and Burns, S. A. (1997), "Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems, " *Journal of Computing in Civil Engineering*, Vol. 11, No. 3, pp. 184-189.

Fischer, M. A. and Aalami, F. (1996), "Scheduling with Computer-Interpretable Construction Method Models," *Journal of Construction Engineering and Management*, ASCE, Vol. 122, No. 4, pp.337-347.

Fischer, M. and Froese, T. (1996), "Examples and Characteristics of Shared Project Models," *Journal of Computing in Civil Engineering*, Vol. 10, No. 3, pp. 174-182.

Fichman, R. G. and Kemerer, C. F. (1997), Object Technology and Reuse: Lessons Learned from Early Adopters," *Computer*, IEEE, Oct. 1997, pp. 47-59.

Fowler, M. (1997), *Analysis Patterns: Reusable Object Models*, Addison-Wesley Longman, Inc., Reading, MA.

Fowler, M. and Scott, K. (1997), *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley Longman, Inc., Reading, MA.

Froese, T. M. (1996), "Models of Construction Process Information," *Journal of Construction Engineering and Management*, Vol. 10, No. 3, pp. 183-193.

Froese, T. M. and Paulson, B. C., Jr. (1994), "OPIS: An Object Model-Based Project Information System", *Microcomputers in Civil Engineering*, Vol. 9, No. 1, pp. 13-28.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Publishing Company, Reading, MA

Ibbs, C. W. (1997), "Quantitative Impacts of Project Change: Size Issues," *Journal of Construction Engineering and Management*, ASCE, Vol. 23, No. 3, pp. 308-311.

Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G. (1992), *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, New York, NY.

Johnson, R. E. (1997), "Frameworks = (Components + Patterns)," *Communication of the ACM*, Vol. 40, No. 10, pp. 39-42.

Kao, W.-M. and Adeli, H. (1997), "Distributed Object-Oriented Blackboard Model for Integrated Design of Steel Structures", *Microcomputers in Civil Engineering*, Vol. 12, No. 2, pp. 141-155.

Karim, A. and Adeli, H. (1998), "CONSCOM: An OO Construction Scheduling and Change Management System," *Journal of Construction Engineering and Management*, submitted.

Leymeister, D. J. Shah, D. and Jain, S. K. (1993), "Computer Application in Analyzing Change Order Work," *Proceedings of the Fifth International Conference on Computing in Civil and Building Engineering*, Anaheim, CA, June 7-9, ASCE, New York, NY, Vol. 1, pp. 137-144.

Liu, L., Burns, S. A. and Feng, C. -W. (1995), "Construction Time-Cost Trade-Off Analysis Using LP/IP Hybrid Method," *Journal of Construction Engineering and Management*, ASCE, Vol. 121, No. 4, pp. 446-454.

Microsoft (1997), *Microsoft Visual C++ MFC Library Reference*, Parts 1 and 2, Microsoft Press, Redmond, WA.

Park, H.S. and Adeli, H. (1997a), "Data Parallel Neural Dynamics Model for Integrated Design of Large Steel Structures", *Microcomputers in Civil Engineering*, Vol. 12, No. 5, pp. 311-326.

Park, H.S. and Adeli, H. (1997b), "Distributed Neural Dynamics Algorithms for Optimization of Large Steel Structures", *Journal of Structural Engineering*, ASCE, Vol. 123, No. 7, pp. 880-888.

Rogers, G. F. (1997), *Framework-Based Software Development in C++*, Prentice-Hall, Inc., Upper Saddle River, NJ.

Saunders, H. (1996), "Survey of Change Order Markups," *Practice Periodical on Structural Design and Construction*, ASCE, Vol. 1, No. 1, pp. 15-19.

Schmidt, D. C. and Fayad, M. E. (1997), Lessons Learned Building Reusable OO Frameworks for Distributed Software," *Communications of the ACM*, Vol. 40, No. 10, pp. 85-87.

Shepherd, G. and Wingo, S. (1996), *MFC Internals--Inside the Microsoft Foundation Class Architecture*, Addison-Wesley Developers Press, Reading, MA.

Stepanov, A. and Lee, M. (1995), *The Standard Template Library*, Hewlett-Packard Laboritories.

Stumpf, A. L., Ganeshan, R., Chin, S. and Liu, L. Y. (1996), "Object-Oriented Model for Integrating Construction Product and Process Information," *Journal of Computing in Civil Engineering*, Vol. 10, No. 3, pp. 204-212.

Yu, G. and Adeli, H. (1991), "Computer-Aided Design using Object-Oriented Programming Paradigm and Blackboard Architecture", *Microcomputers in Civil Engineering*, Vol. 6, No. 3, pp. 177-189.

Yu, G. and Adeli, H. (1993), "Object-Oriented Finite Element Analysis using an EER Model", *Journal of Structural Engineering*, ASCE, Vol. 119, pp. 2763-2781.

LIST OF CAPTIONS FOR FIGURES

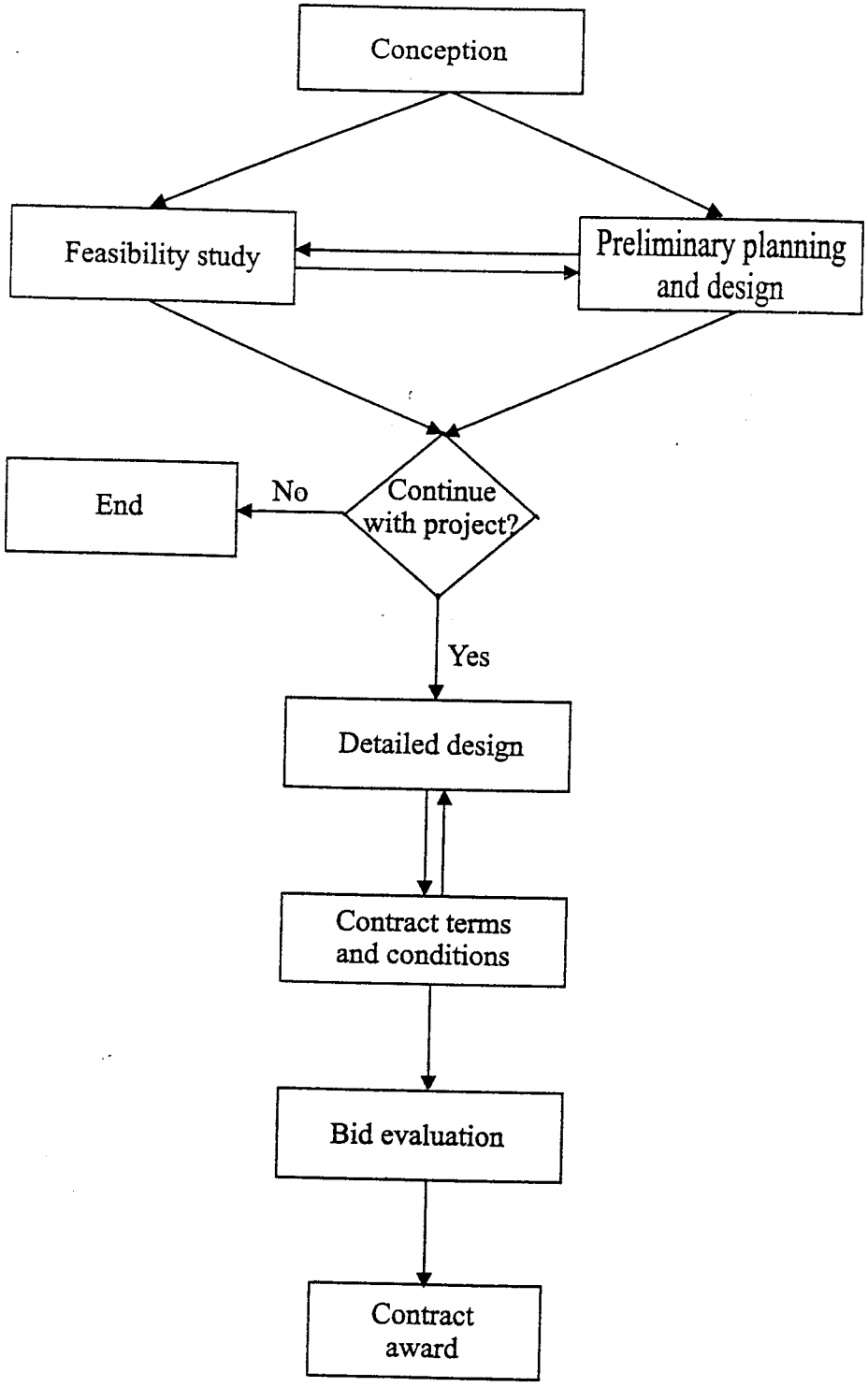

Conception
Feasibility study
Preliminary planning and design
End
No
Continue with project?
Yes
Detailed design
Contract terms and conditions
Bid evaluation
Contract award

Figure 1

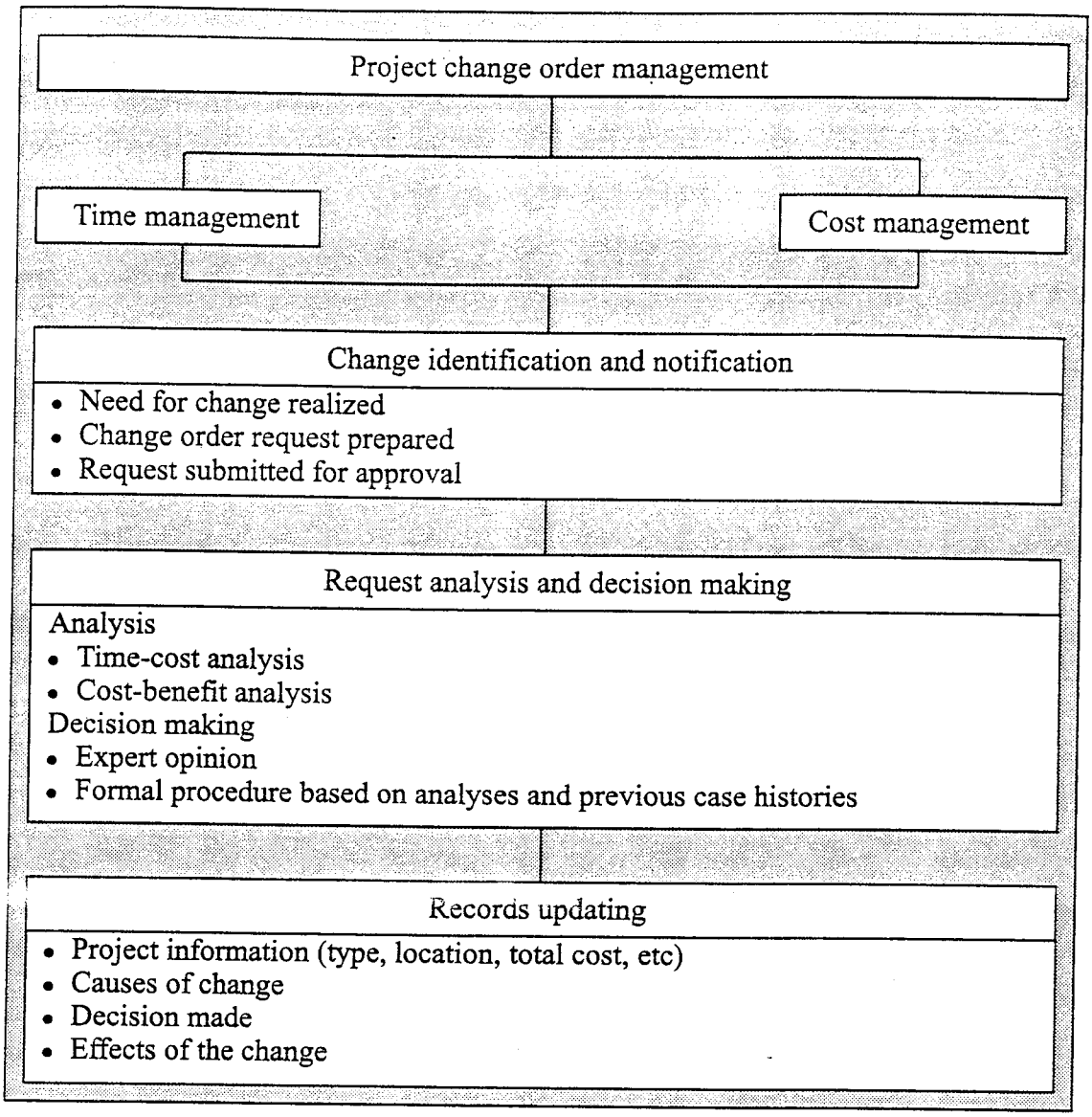

Project change order management
Time management
Cost management
Change identification and notification
• Need for change realized
• Change order request prepared
• Request submitted for approval
Request analysis and decision making
Analysis
• Time-cost analysis
• Cost-benefit analysis
Decision making
• Expert opinion
• Formal procedure based on analyses and previous case histories
Records updating
• Project information (type, location, total cost, etc)
• Causes of change
• Decision made
• Effects of the change

Figure 2

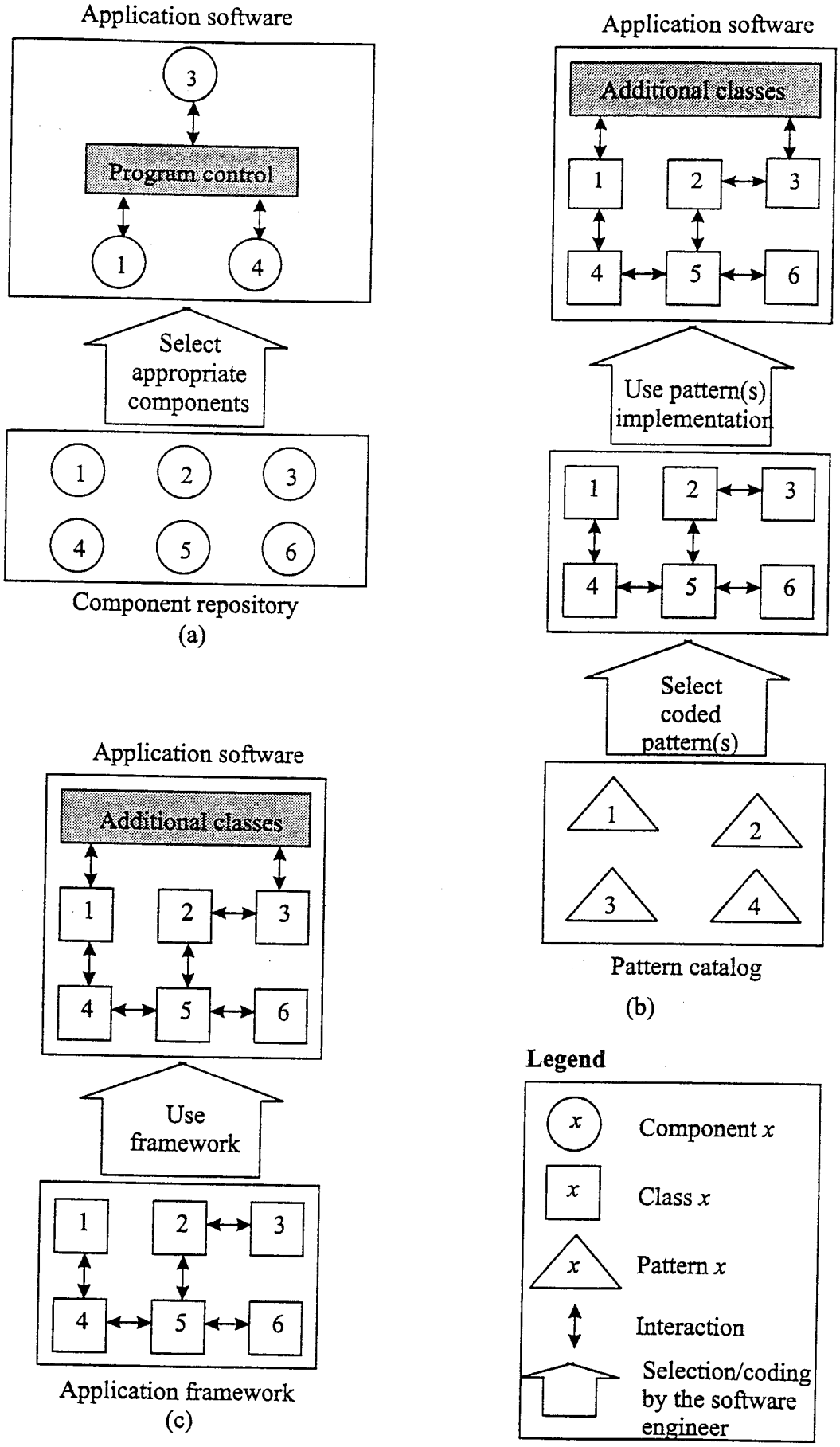

Application software
3
Program control
1
4
Select appropriate components
1
2
3
4
5
6
Component repository
(a)
Application software
Additional classes
1
2
3
4
5
6
Use pattern(s) implementation
1
2
3
4
5
6
Select coded pattern(s)
1
2
3
4
Pattern catalog
(b)
Application software
Additional classes
1
2
3
4
5
6
Use framework
1
2
3
4
5
6
Application framework
(c)
Legend
x
Component x
x
Class x
x
Pattern x
Interaction
Selection/coding by the software engineer

Figure 3

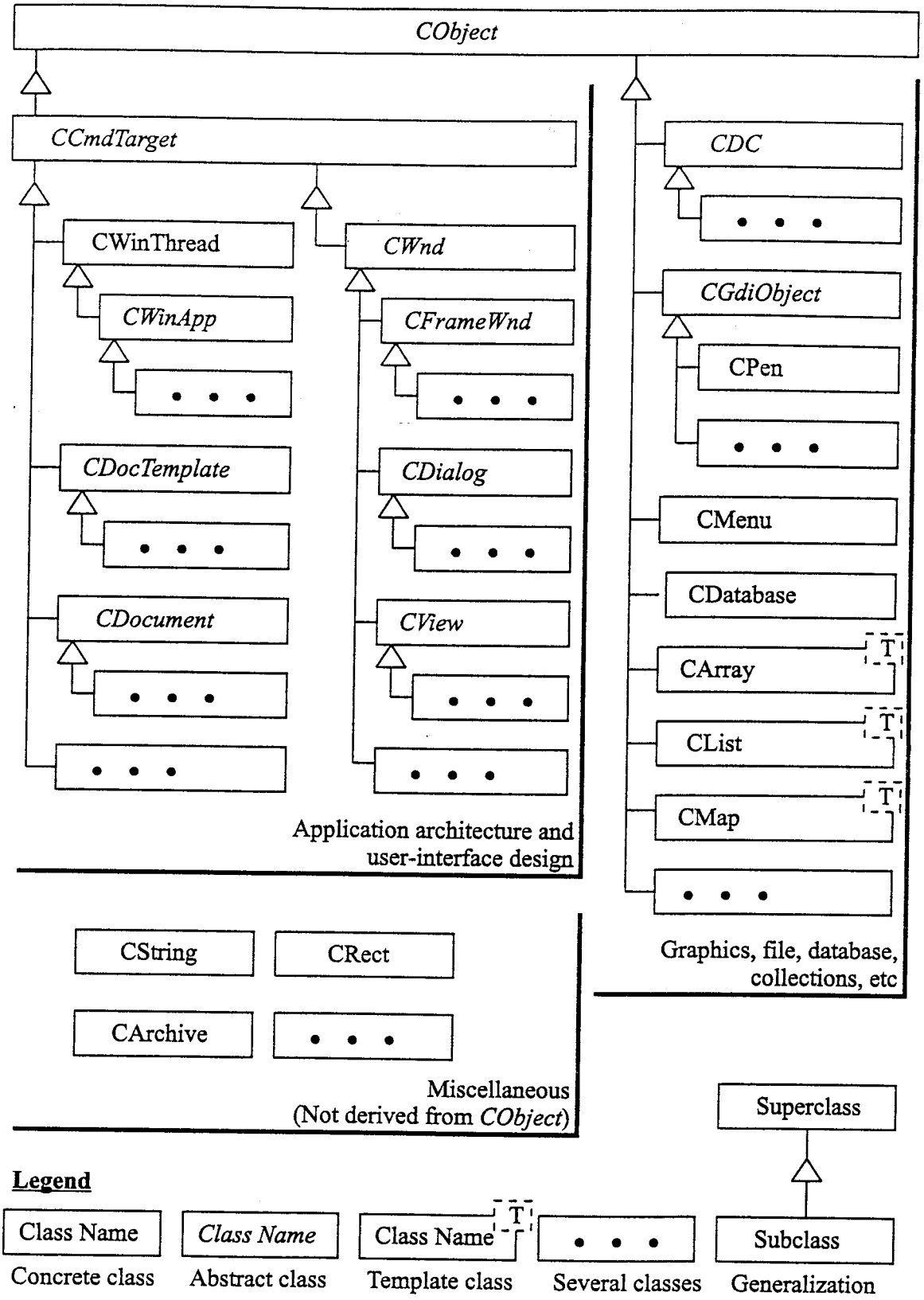

CObject
CCmdTarget
CWinThread
CWinApp
CDocTemplate
CDocument
CWnd
CFrameWnd
CDialog
CView
Application architecture and user-interface design
CDC
CGdiObject
CPen
CMenu
CDatabase
CArray
CList
CMap
Graphics, file, database, collections, etc
CString
CRect
CArchive
Miscellaneous (Not derived from CObject)
Legend
Class Name
Concrete class
Class Name
Abstract class
Class Name
Template class
Several classes
Superclass
Subclass
Generalization

Figure 4

CONSCOM: AN OO CONSTRUCTION SCHEDULING AND CHANGE MANAGEMENT SYSTEM

Asim Karim[1] and Hojjat Adeli[2], Fellow, ASCE

ABSTRACT: In an accompanying article, an object-oriented (OO) information model was presented for construction scheduling, cost optimization, and change order management based on creation of a domain-specific development *framework*. The framework architecture is developed using generic software design elements, called *patterns*, which provide effective low-level solutions for creating, organizing, and maintaining objects. The OO model has been implemented in a prototype software system for management of construction projects, called CONSCOM, using the Microsoft Foundation Class (MFC) library in Visual C++. CONSCOM is particularly suitable for highway construction change order management. It can be used by the owner as an intelligent decision support system in schedule reviews, progress monitoring, and cost-time trade-off analysis for change order approval. The OO information model for construction scheduling and cost management can be integrated into a concurrent engineering model for the Architecture, Engineering, and Construction (AEC) industry.

INTRODUCTION

In an accompanying article, Karim and Adeli (1998) presented an object-oriented (OO) information model for construction scheduling, cost optimization, and change order

[1] Graduate Research Associate, Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University.
[2] Professor, Department of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 470 Hitchcock Hall, 2070 Neil Avenue, Columbus, OH 43210, USA.

management. In this article we present its implementation in a prototype software system called CONSCOM (CONstruction Scheduling, Cost Optimization, and Change Order Management.

CONSCOM APPLICATION ARCHITECTURE

Application development often requires the use of multiple specialized frameworks. In the design of complex software systems, it is essential that the role of each framework be defined accurately. Further, their dependencies must be clearly outlined to avoid any conflicts. In this work, a layered approach is used (Baumer, et al., 1997). This approach is based on high-level object decomposition and generalization of the application domain. This results in self-contained modules that can be implemented by one or more frameworks. Further, the apriori separation clearly outlines the scope of the frameworks and avoids duplication of functionality in their development. Note that a framework is a collection of cooperating classes relevant to a specific domain. Therefore, a related subset of a framework's classes may also be called a framework. For example, the MFC library is a framework but it can also be thought of as the collection of general-purpose application architecture, user-interface, and data management frameworks.

The high-level application architecture for the construction management domain is shown schematically in Figure 1. Three levels of abstraction are modeled. The outermost layer is the shell layer. All other layers depend on and use the services of this layer. Typically it provides commonly used data structures, mathematical functions, client/server middleware (low-level transaction management software), and request brokers (software that manages cooperation and communication among heterogeneous

software components) for distributed computing. The shell layer is closely linked to the operating system, and its implementation in the form of frameworks is usually available for most application development environments.

Depending on the shell layer is the productivity layer which is subdivided into database and user-interface layers. The database layer provides an interface to the applications for data management, storage, and retrieval. The frameworks in this layer may support OO or relational database management under either a centralized or distributed environment. The frameworks in the user-interface layer aid in the design of user-friendly application/user interactions. Depending on both the shell and productivity layers is the application domain layer which contains domain specific details. Generally, this layer contains algorithms and models for the solution of problems in the domain. It is often subdivided to further categorize and generalize the application domain requirements. One or more frameworks may be used to implement this layer.

Figure 2 shows a detailed view of the architecture in the form of a *package* diagram. The notation used in this and all the subsequent figures is based on the new standard Unified Modeling Language (UML) (Fowler and Scott, 1997). This diagram shows the breakdown of the application into packages and their dependencies. A package is a collection of related software elements. These elements may be classes, components, or frameworks. In Figure 2, the packages represent collection of classes. The dashed-line arrows indicate dependency of a package on another. A software dependency exists if any change in a package requires a change in the dependent package. Note that software dependencies are not transitive. That is, if package *A* depends on package *B* and package

B depends on package *C*, package *A* may not necessarily depend on package *C* in terms of code implementation.

The application domain layer of the high-level architecture view is divided into two packages: a *Domain* package and an *Application* package (Figure 2). This is done because all application areas in construction scheduling, cost optimization, and management share common domain information. A framework-based implementation of this layer is presented in this article.

The *Domain* package provides a common information base for software development in construction engineering. It contains organizational, construction schedule, and construction cost software building blocks that encapsulate domain specific knowledge and dependency logic in a reusable format. This layer depends on the "*File and database support*" package and the "*Miscellaneous support classes*" package of the MFC library.

The *Application* package has two nested packages: *User-interface* and *Model*. The package *Model* is a collection of various computational models and methods in the domain. This package depends on the *Domain* package and provides program logic and control for the solution of specific problems in the domain. The package *User interface* provides a unified user interface for the applications. This package depends strongly on the general-purpose *User interface* and *Graphics* packages provided by the MFC library.

The MFC library is used for all general-purpose functionality required for the development of CONSCOM. Software dependencies also exist between packages in the MFC library. However, these are not shown in Figure 2 for simplicity. By factoring out common concepts it is easier to maintain uniformity and consistency among applications

in an area. The software design details of the domain-specific framework for construction scheduling, cost optimization and change order management (CONSCOM) is presented in the following section.

THE CONSCOM FRAMEWORK

The CONSCOM framework is a white box-black box framework. A framework is called a white box if it provides abstract classes only and no concrete classes. Use of such a framework requires the development and implementation of concrete classes that are derived from the framework classes. This in turn requires an understanding of the framework design. Hence, the name white-box. On the other hand, black box frameworks provide default implementations that can be used directly with little or no change. The CONSCOM framework has elements of both. It defines an interface that can be implemented as desired. It also provides default implementations of the new problem solving techniques in scheduling, cost optimization and, cost estimation (Adeli and Karim, 1997; Adeli and Wu, 1998).

The CONSCOM framework is built using software patterns. This makes the software design robust. Software patterns will also serve as documentation for the framework (Odenthal and Quibeldey-Cirkel, 1997). However, most of the software design patterns discovered and presented in the literature deal with generic situations as opposed to domain-specific situations. Generic software patterns are usually low-level software design ideas for creating, organizing, and maintaining objects. Domain-specific patterns, on the other hand, are usually more elaborate and provide domain modeling ideas. In the development of CONSCOM generic patterns are used when applicable. The

OO design principles are also exploited for cases where the documented patterns do not provide a solution. The software design concepts presented in this work for modeling construction management problems can be generalized into domain-specific software design patterns for construction management. In the following discussion software design pattern names are identified in small caps.

Figures 3 and 4 show the classes in CONSCOM, their characteristics (abstract or concrete), and their inheritance hierarchy. The classes in these figures correspond to the *Domain* and *Application* packages, respectively. Brief description of classes in Figures 3 and 4 are given in Appendix I. Most of the classes have the MFC class *CObject* as the root to take advantage of the services it provides such as object storage and retrieval. Note that most of the higher level classes are abstract. This is a fundamental design concept in frameworks that allows customization through subclassing. The basic framework (consisting of the abstract classes only) just provides an interface. Derived classes bind the interface to a specific implementation. As will be described shortly, an abstract class controls which implementation class is instantiated. Classes starting with the prefix 'CI' are implementation classes for construction scheduling, cost optimization, and cost estimation (Adeli and Karim, 1997; Adeli and Wu, 1998).

The focal point in construction engineering and management is the construction project. A project (construction or otherwise) is defined for every collection of related activities that needs to be completed under certain constraints and hence needs to be managed. Some key elements of a construction project are the proposed and implemented plan of work, the cost and time reference, the cost estimate, and the descriptive progress report. Effective management requires that a track record of all these information

elements be maintained. Figure 5 shows a class diagram of how this is modeled in CONSCOM. Attributes and operations shown in Figure 5 to 11 are defined in Appendix II. The classes *CProject*, *CPlan*, *CCostEstimate*, and *CProgressReport* abstract their real-world equivalents. Objects of each one of these classes are associated with a *CVersion* object. Class *CVersion* maintains the current version number, the time of creation and last update, and a pointer to the person or organization that made the last change. The class *CActor* abstracts a role or job function. A project has several job functions associated with it such as manager, contractor, and owner.

The class *CProject* plays the role of an abstract factory as described in the software pattern ABSTRACT FACTORY (Gamma et al., 1995). This software pattern describes how to solve the problem of creating several related objects without explicitly specifying their concrete classes. For example, class *CProject* only defines the interface (*NewPlan()*) for creating a new plan (Figure 5). The subtype of *CPlan* object created will depend on which concrete class of *CProject* is used. In Figure 5, *CIProject* creates objects of class *CIPlan*. Therefore, the set of objects created will depend on which 'factory' is used. The method or operation *NewPlan()* (and other new operations) in class *CProject* represents a factory method as defined in the pattern FACTORY METHOD. This pattern describes a technique to delegate the creation of an object to its concrete subclasses. This technique uses the dynamic binding construct in OOP languages in order to make it transparent to the user. The classes *CProject*, *CPlan*, *CTask*, *CConstraint*, *CCostEstimate*, *CProgressReport*, and *COptimizationProblem* all provide a factory method to isolate the object creation process from the user. The type of the object created will depend on the class to which the pointer points. These classes are also the *hot spots*

of the CONSCOM framework. A hot spot represents a point of variability which can be used to customize the framework (Schmid, 1996, 1997).

Many individuals and organizations become associated with a construction project throughout its life-span. However, in several situations no difference is made between whether the associated party is an individual or an organization. For example, the role of the project's owner may be played by either an individual or an organization. Figure 6 shows how this concept is modeled in CONSCOM. This software design is based on the PARTY pattern (Fowler, 1997). The class *CParty* abstracts characteristics common to persons and organizations such as contact address and job function or role.

Effective representation of measurements is essential in any software system. The construction engineering domain has a wide range of measurement types such as distance, time, and volume. Further, a wide range of measurement units are used such as hectares or squared meters for measuring areas. To maintain the integrity of the measurement and the calculations based on them both the value and the unit of the measurement must be encapsulated in a single object. The class *CQuantity* (Figure 7a) provides this functionality. It also allows conversion of a value from one unit to another. This conversion is done transparently whenever any arithmetic or logical operators are used on objects of *CQuantity*. In the CONSCOM framework all measurements are of type *CQuantity*. Ratios for converting a value from one unit to another are maintained by an object of class *CConversionRatio* (Figure 7b). Only one instance of *CConversionRatio* is required. Multiple instances are not only unnecessary but they also create the problem of data consistency maintenance among all the instances. To prevent multiple instances

of class *CConversionRatio*, it is designed as a singleton as defined in the software pattern SINGLETON (Gamma et al. 1995).

The plan is the most significant element in the management of a construction project. Traditionally, a plan is defined as the timetable of the tasks that needs to be carried out. In other words, a plan tells us when each task will be executed in the future. In this research a broader view of the construction plan is adopted. The construction plan is considered to represent the current state of the project. This includes the tasks that have been completed, those that are in progress, and those proposed to be carried out in the future. Similarly a construction task may be a proposed task, a completed task, or an in-progress task. Using these broader conceptual views it is possible to capture the state of work at any given point in time. This is very important in change order management and conflict resolution because comparisons can be made much more easily. Note that unless a plan has been executed it can be scheduled and optimized no matter which state it is in.

Figure 8 shows how the construction plan and task are modeled in CONSCOM. The classes *CPlan* and *CTask* are defined as generalization (subtypes) of the class *CAction*. The class *CAction* abstracts the common characteristics in classes *CPlan* and *CTask* and gives them similar interfaces. For example, both construction plans and tasks have a time reference and both of them can be scheduled. To model this in software a supertype is created which abstracts the common characteristics. A status variable identifies which state a plan or task is in. The states relevant to construction management are: Proposed, started, completed on-time, delayed, abandoned, and suspended. Each object of class *CAction* is associated with a previous version of itself. This generates a

hierarchy that captures the history of changes that have occurred over time. Each object of class *CAction* ensures that the previous version object pointed to is valid.

Construction task and plan have some similarities of behavior as described above. A construction plan, however, is a collection of construction tasks. In a computer model this structural difference must be hidden from the user when executing a common behavior operation. Figure 8 shows how this is modeled in CONSCOM. The class *CPlan* composes objects of class *CActionReference* that holds a reference to a single object of class *CAction*. The class *CAction* provides an interface for managing collection of objects. These operations are only implemented in the composite class (*CPlan*) and not in the child class (*CTask*). This design is based on the COMPOSITE pattern (Gamma et al., 1995). The composition relationship between *CPlan* and *CActionReference* means that an object of class *CPlan* owns objects of class *CActionReference*. In other words, the creation and destruction of *CActionReference* objects depend on the *CPlan* object. Creation of a *CActionReference* object by an object of *CPlan* (in *Add(CAction a)*) will occur only when the *CAction* object *a* being added is not already present. This prevents a *CPlan* object from having multiple references to the same *CAction* object. Note that with the present software design it is also possible to model a plan that contains a combination of both tasks and plans.

Scheduling constraints between *CAction* objects (tasks and/or plans) are modeled by the *CActionReference* and *CConstraint* classes (Figure 8). The class *CConstraint* encapsulates a single scheduling constraint between two objects of class *CAction*. It provides an interface for calculating constraint violation and determining constraint satisfaction. In CONSCOM, two concrete classes are provided which implement these

operations. The class *CITimeConstraint* models the precedence relation constraint, while the class *CIBufferConstraint* models time and distance buffer constraints. These implementations are developed based on the general scheduling model of Adeli and Karim (1997).

In construction scheduling accurate modeling of the construction task is essential. A task may be repetitive or non-repetitive and, if it is repetitive, it may be executed by a single crew or multiple crews, and/or it may require distance modeling. This latter concept is essential in linear construction projects, such as highway construction. Also, effective cost control requires that the resources required and used by a construction task are accurately modeled. A mathematical model of these concepts is presented by Adeli and Karim (1997). Its representation in CONSCOM is shown in Figure 9. The class *CCrew* represents a construction crew. A segment or location of work for a crew is encapsulated by the class *CSegment*. An object of class *CSegment* also encapsulates the job conditions and quantity of work for that segment. Each *CCrew* object owns one or more *CSegment* objects. However, two *CSegment* objects cannot have overlapping locations. This constraint is enforced by the *CCrew* object that is creating the *CSegment* object. The class *CResource* abstracts a construction resource such as construction equipment and labor. Depending on the productivity data for the resource a cost duration relationship can be developed. This information is encapsulated by the class *CCostDurationRln*. The class *CCost* provides an interface for managing the construction cost.

Decisions involving change order management and conflict resolution often require cost comparisons among different versions and states of the construction project.

The scheduling/cost optimization model (Adeli and Karim, 1997) provides a mathematical formulation to solve the construction project direct cost optimization problem. The optimization model used is the patented neural dynamics model of Adeli and Park (1996). This provides the basis for consistently reliable evaluation of the cost of a project at any point in time. Figure 10 shows how the general neural dynamics model is used to solve the problem of construction cost minimization. The class *CNeuralDynamics* encapsulates the neural dynamics optimization algorithm. The class *COptimizationProblem* defines the interface required for an optimization problem in order to be solved by the neural dynamics model. In CONSCOM, the construction direct cost optimization problem is defined by an object of type *CPlan*. The goal is to map this problem (defined by an object of type *CPlan*) to another object that has an interface of class *COptimizationProblem*. In this design two software patterns are used. The ADAPTER (Gamma et al. 1995) pattern solves the problem of how to adapt one interface to another and the TEMPLATE METHOD pattern solves the problem of how to choose and create the correct adapter object. The *Optimize()* operation in *CPlan* is the template method. The type of *COptimizationProblem* created depends on the implementation of the *CreateOptProb()* operation. In the present case, the *CIPlan* object creates an object of *CIOptimizationProblem*.

Figure 11 shows an example of how an MFC application document is associated with the corresponding domain information. For example, class *CProjectDoc* encapsulates an application document that contains construction project information. The FACTORY METHOD pattern is again used to control which concrete subtype of *CProject* is created.

A major concern of the construction project owner is the effective management of change orders. A change order can be initiated by the contractor, the owner, the designer, or any other stakeholder in the project. As a result of a change the contractor may claim an extension in the project duration in addition to compensation for any additional work required. To support its claim an updated schedule is submitted to the owner for review. The owner's change order management process can be divided into two steps. First, a qualitative review of the schedule is carried out to identify any omissions, errors, inaccuracies, and non-compliance with the contract documents. Second, a quantitative analysis is done to determine the impact of the change and its comparison with what is being claimed by the contractor. Contractors often modify the schedule to exaggerate their claims. Therefore, these steps must be executed with care and accuracy. Further, consistency and reliability of the owner's decision is essential for it to be acceptable to the contractor. The CONSCOM framework provides all these capabilities to automate and expedite the decision of the project owner when faced with a change order.

The use case diagram for construction change order management is shown in Figure 12. A use case diagram captures the users and their uses of a software system. A use case represents a requirement of a software system that must be satisfied. When faced with a change order the owner's primary use of the system is decision support. This is represented by the *Change order management* use case (Figure 12). The qualitative and the quantitative steps of the change order management process are represented by the use cases *Review plan* and *Analyze scenarios*, respectively. The *Review plan* use case involves a compliance check with the contract documents and a comparison with a previous version of the plan. If any error or non-compliance is identified the contractor is

asked to send an updated plan. The *Analyze scenarios* use case involves the generation of several construction plan scenarios and carrying out time-cost trade-off analyses on them. Typical scenarios include trying different project durations or following an alternate logic for the construction tasks. These analyses are based on scheduling and cost optimization of the generated scenarios. Therefore, the results are consistent across all scenarios. From these analyses the owner is able to make a decision on the reasonableness of the claims made by the contractor. Note that the owner can also analyze different construction plan scenarios to determine the feasibility of a change that it (or he) wants in the project.

Figure 13 shows an object interaction diagram in CONSCOM to solve the problem of change order management. The sequence of operations executed by each object is identified from top to bottom along the dashed line. The latest plan is analyzed under multiple scenarios to determine the exact impact of the change. The process terminates with updating of the *CProject* object and recording of the decision taken in the project progress report.

FINAL COMMENTS

CONSCOM is a reusable and extendible software system for scheduling and management of construction projects with cost optimization capability. It is particularly suitable for construction change order management. Change in construction projects after contract award is a common occurrence. It consumes a lot of resources in its implementation and in the resolution of any conflicts that often arise among the stakeholders. CONSCOM takes the owner's view of the problem. It aids the owner in schedule reviews, progress monitoring, and cost-time analyses for change order approval.

ACKNOWLEDGMENT

This manuscript is based on a research project sponsored by the Ohio Department of Transportation and Federal Highway Administration.

APPENDIX I.BRIEF DESCRIPTION OF CLASSES IN THE CONSCOM FRAMEWORK (FIGURES 3 AND 4)

CAction: Provides an interface for managing actions (tasks and plans).

CActionReference: Encapsulates a reference to a *CAction* object.

CActor: Encapsulates a role played by a person or organization.

CAddress: Encapsulates a street address.

CConstraint: Encapsulates a construction scheduling constraint.

CContinuousCDR: Encapsulates a continuous cost duration relation.

CContractor: Abstracts a contractor of a construction project.

CConversionRatio: Encapsulates ratios for the conversion of values from one unit to another.

CCost: Provides an interface for managing construction costs.

CCostDurRln: Encapsulates a cost duration relation.

CCostEstimate: Provides an interface for managing construction cost estimates.

CCrew: Abstracts a construction crew.

CDirectCost: Provides support for construction direct cost management.

CDiscrerteCDR: Encapsulates a discrete cost duration relation.

CEquipment: Abstracts a construction equipment.

CI: Class names starting with 'CI' are implementation classes for the corresponding class names starting with 'C' in CONSCOM.

CIBufferConstraint: Provides support for distance or buffer constraints.

CIndirectCost: Provides support for construction indirect cost management.

CITimeConstraint: Provides support for time constraints

CLabor: Abstracts information of construction labor.

CNeuralDynamics: Abstracts the neural dynamics model for the solution of optimization problems.

COptimizationProblem: Provides an interface for the solution of optimization problems using the neural dynamics model.

COwner: Abstracts an owner of a construction project.

CParty: Base class of *CContractor*, *COwner*, and *CPerson* (encapsulates their common features).

CPerson: Abstracts a person.

CPlan: Provides an interface for managing construction plans.

CPlanDoc: Provides an interface for a construction plan application document.

CPlanView: Provides an interface for a construction plan application view.

CProgressReport: Provides support for managing progress reports.

CProject: Provides an interface for managing construction projects.

CProjectDoc: Provides an interface for a construction project application document.

CProjectView: Provides an interface for a construction project application view.

CQuantity: Encapsulates a measurement value and its unit.

CRegularizationNetwork: Abstracts the regularization neural network model for cost estimation.

CReportDoc: Provides an interface for a construction progress report application document.

CReportView: Provides an interface for a construction progress report application view.

CResource: Abstracts a construction resource.

CSegment: Abstracts a segment of construction work.

CTask: Provides an interface for managing construction tasks.

CVersion: Provides support for version control.

APPENDIX II. BRIEF DESCRIPTION OF THE ATTRIBUTES AND OPERATIONS SHOWN IN FIGURES 5 TO 11

Attributes

Description: Description of the object.

DirectCost: Direct cost.

Duration: Time that indicates the duration of an activity.

Email: Email address.

ID: Identification

IndirectCost: Indirect cost.

Instance: Instance of an object.

JobCondFactor: Job condition factor.

Location: Address information.

Name: Name of the object.

PercentComplete: Percentage of a work that is complete.

QtyOfWork: Quantity of work required.

Quantity: Quantity.

Ratios: Conversion rations between units..

Responsibilities: List of responsibilities.

StartDist: Starting distance of a segment of work.

StartTime: Starting time of an activity.

Status: Status information.

StopDist: Stopping distance of a segment of work.

StopTime: End time of an activity.

Telephone: Telephone number.

Type: Type information.

Unit: Measurement unit.

Value: Numeric value.

X: vector of variables.

Operations:

Add(): Adds an object into a collection.

AddRatio(): Adds a ratio to a collection.

Create(): Creates an object

CreateOptProb(): Creates an object of type *COptimizationProblem*.

Delete(): Deletes an object

EqualityConstraint(): Evaluates the equality constraints of an optimization problem.

Estimate(): Estimates the cost of a project.

GetEstimate(): Returns a pointer to an object of type *CCostEstimate*.

GetInstance(): Returns an instance of an object.

GetPlan(): Returns a pointer to an object of type *CPlan*.

GetRatio(): Returns a ratio from a collection.

GetReport(): Returns a pointer to an object of type *CProjectReport*.

InequalityConstraint(): Evaluates the inequality constraints of an optimization problem.

New(): Creates a new object.

NewEstimate(): Creates a new object of type *CCostEstimate*.

NewPlan(): Creates a new object of type *CPlan*.

NewReport(): Create a new object of type *CProgressReport*.

ObjectFunction(): Returns a value of the objective function of the optimization problem.

OperatorX(): Implements arithmetic and logical operators of type *CQuantity*.

ConvertTo(): Converts a measurement from one unit to another.

Optimize(): Minimizes the cost of a construction plan.

RemoveRatio(): Removes a ratio from a collection.

Satisfy(): Satisfies a scheduling constraint.

Schedule(): Schedules a construction plan or task.

SetOptProb(): Sets up a reference to an optimization problem.

Update(): Updates the state of an object.

Violation(): Returns the amount of violation of a scheduling constraint.

APPENDIX III. REFERENCES

Adeli, H. and Karim, A. (1997), "Scheduling/Cost Optimization and Neural Dynamics Model for Construction," *Journal of Construction Engineering and Management*, ASCE, Vol. 123, No. 4, pp. 450-458.

Adeli, H. and Park, H.S. (1996), "Hybrid CPN-Neural Dynamics Model for Discrete Optimization of Steel Structures", *Microcomputers in Civil Engineering*, Vol. 11, No. 5, pp. 355-366.

Adeli, H. and Wu, M, (1998), "Regularization Neural Network Model for Highway Construction Cost Estimation", *Journal of Construction Engineering and Management*, ASCE, Vol. 124, No. 1.

Baumer, D., Gryczan, G., Knoll, R., Lilienthal, C., Riehle, D. and Zullighoven, H. (1997), "Framework Development for Large Systems," *Communications of the ACM*, Vol. 40, No. 10, pp. 52-59.

Fowler, M. (1997), *Analysis Patterns: Reusable Object Models*, Addison-Wesley Longman, Inc., Reading, MA.

Fowler, M. and Scott, K. (1997), *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley Longman, Inc., Reading, MA.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Publishing Company, Reading, MA

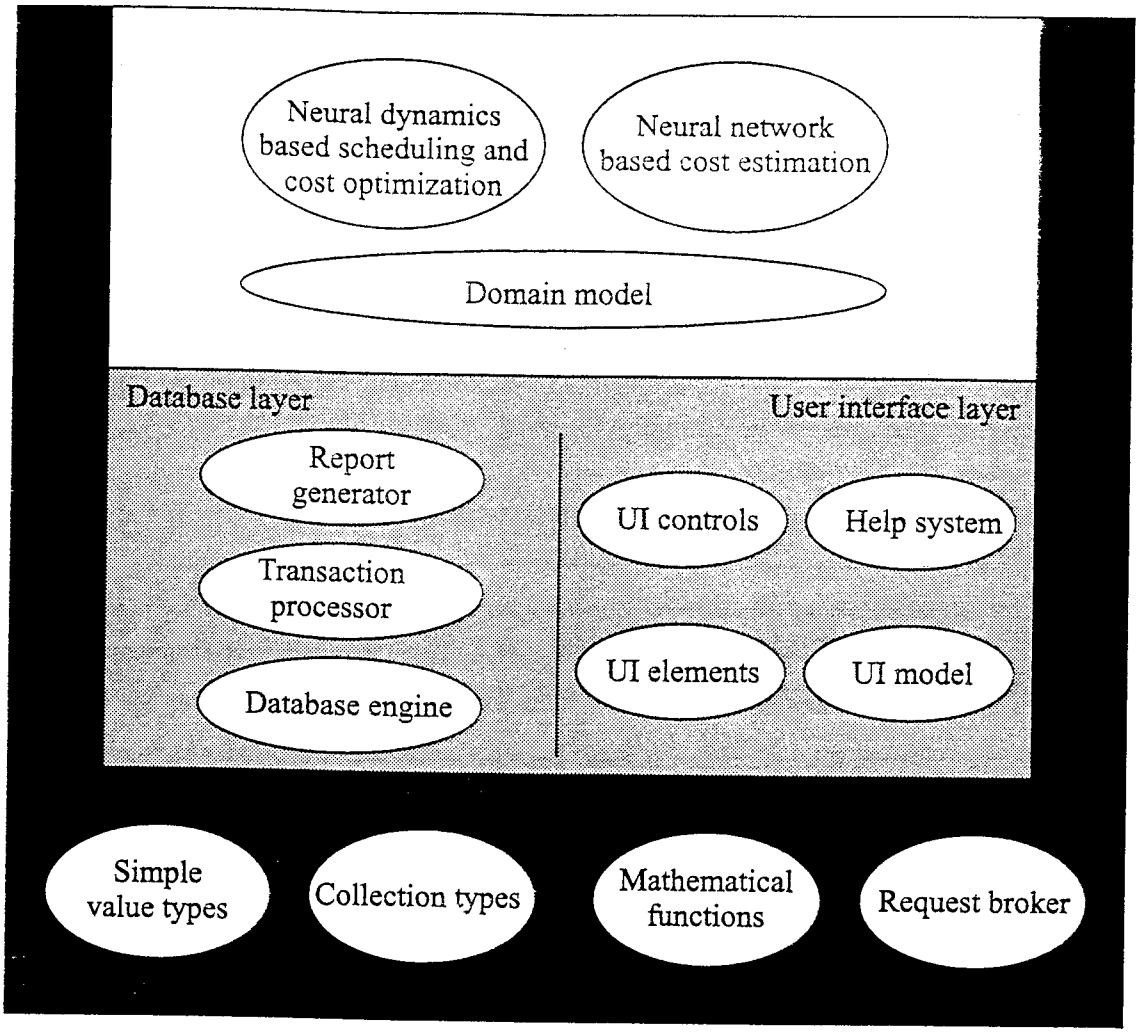
Karim, A. and Adeli, H. (1998), "Object-Oriented Information Model for Construction Project Management," *Journal of Construction Engineering and Management*, ASCE, submitted.

Odenthal, G. and Quibeldey-Cirkel, K. (1997), "Using Patterns for Design and Documentation," *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP '97)*, Jyvaskyla, Finland, June 1997, pp. 511-529.

Schmid, H. A. (1996), "Creating Applications from Components: A Manufacturing Framework Design," *IEEE Software*, Vol. 13, No.11, Nov. 1996, pp.67-75.

Schmid, H. A. (1997), "Systematic Framework Design By Generalization," *Communications of the ACM*, Vol. 40, No. 10, pp. 48-51.

LIST OF CAPTIONS FOR FIGURES



Legend

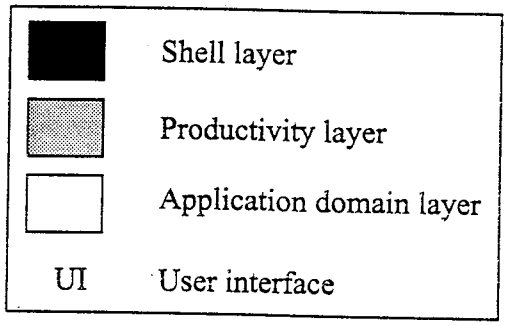


Figure 1

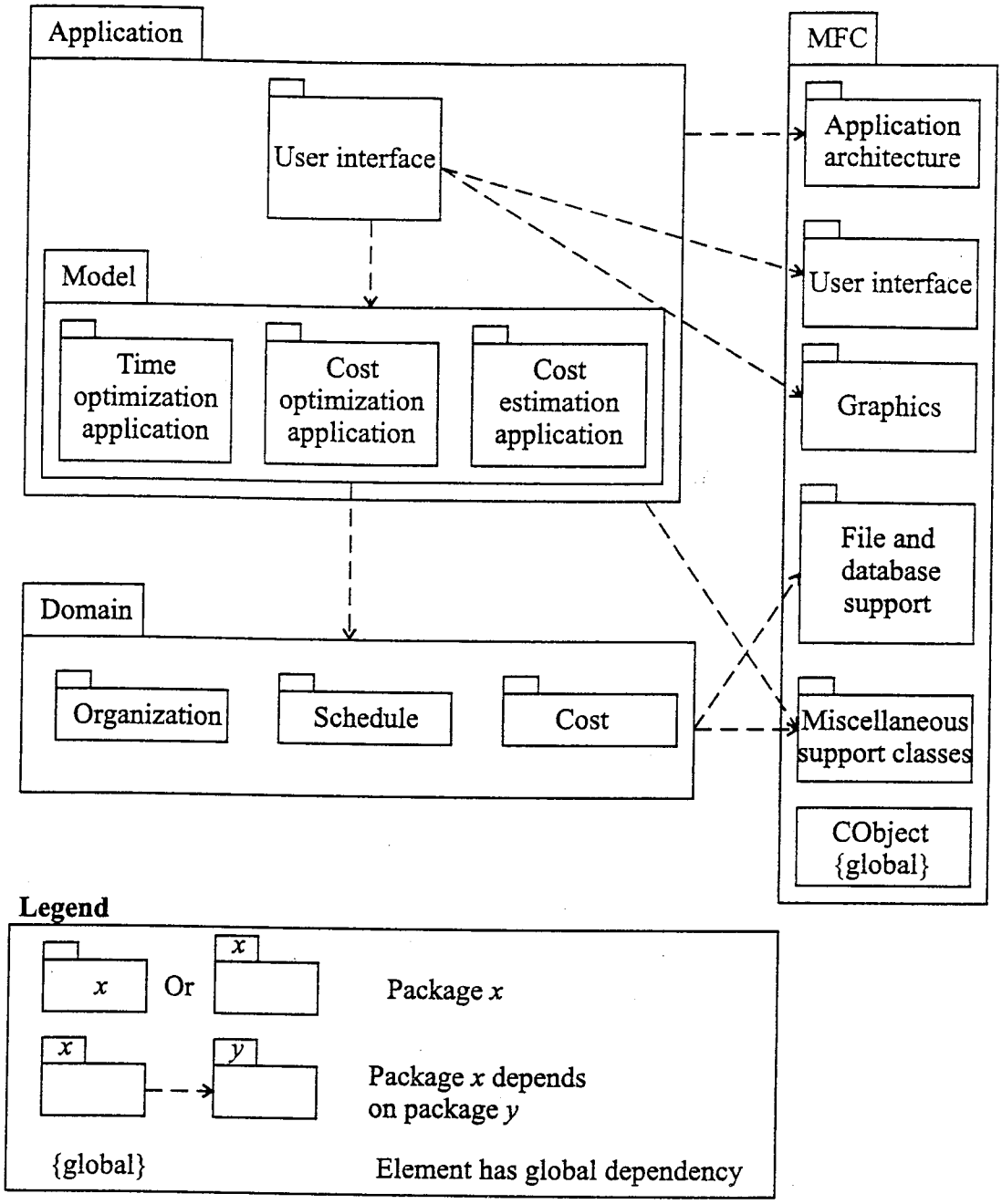

Application
User interface
Model
Time optimization application
Cost optimization application
Cost estimation application
Domain
Organization
Schedule
Cost
MFC
Application architecture
User interface
Graphics
File and database support
Miscellaneous support classes
CObject {global}
Legend
x
Or
x
Package x
x
y
Package x depends on package y
{global}
Element has global dependency

Figure 2

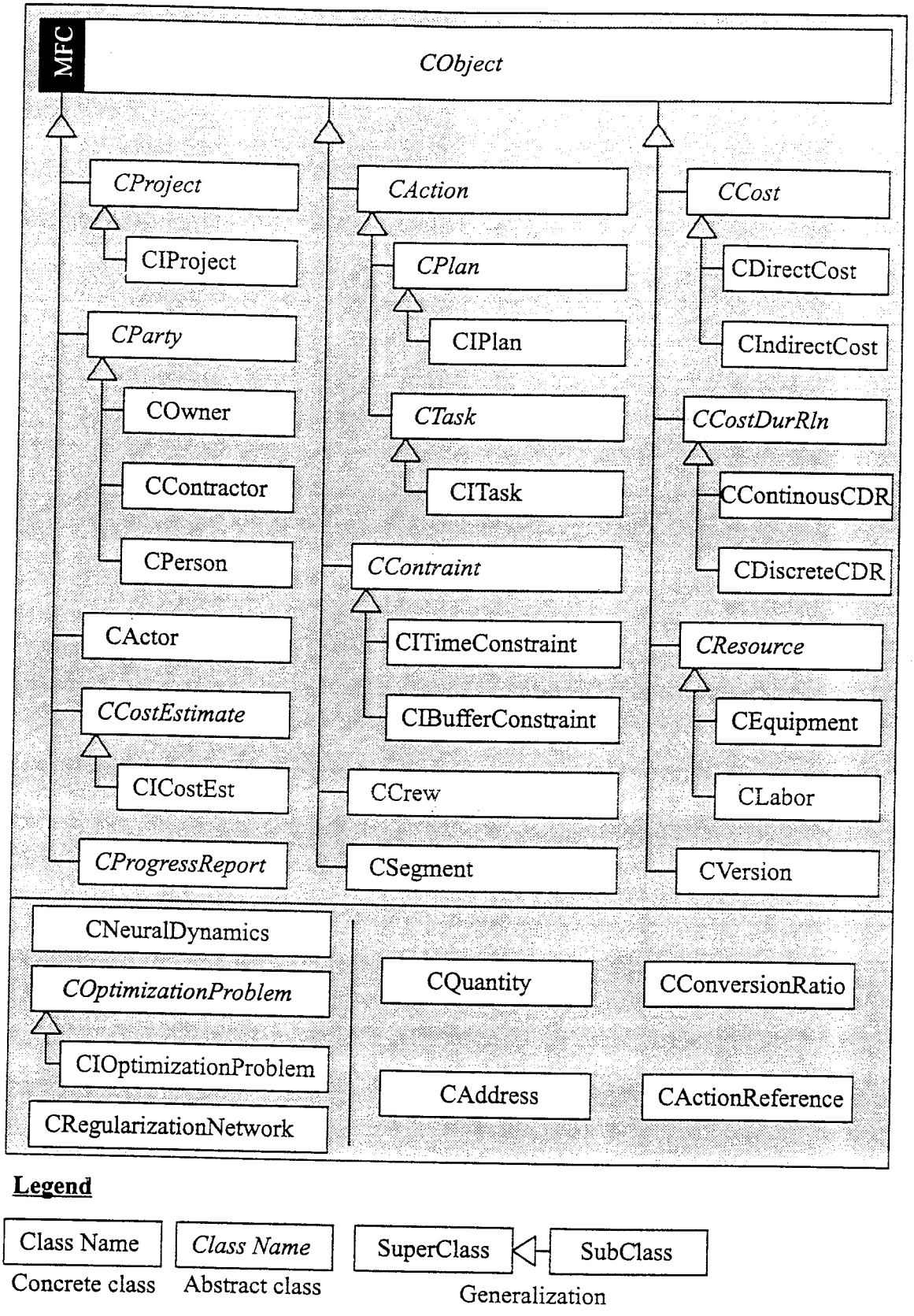
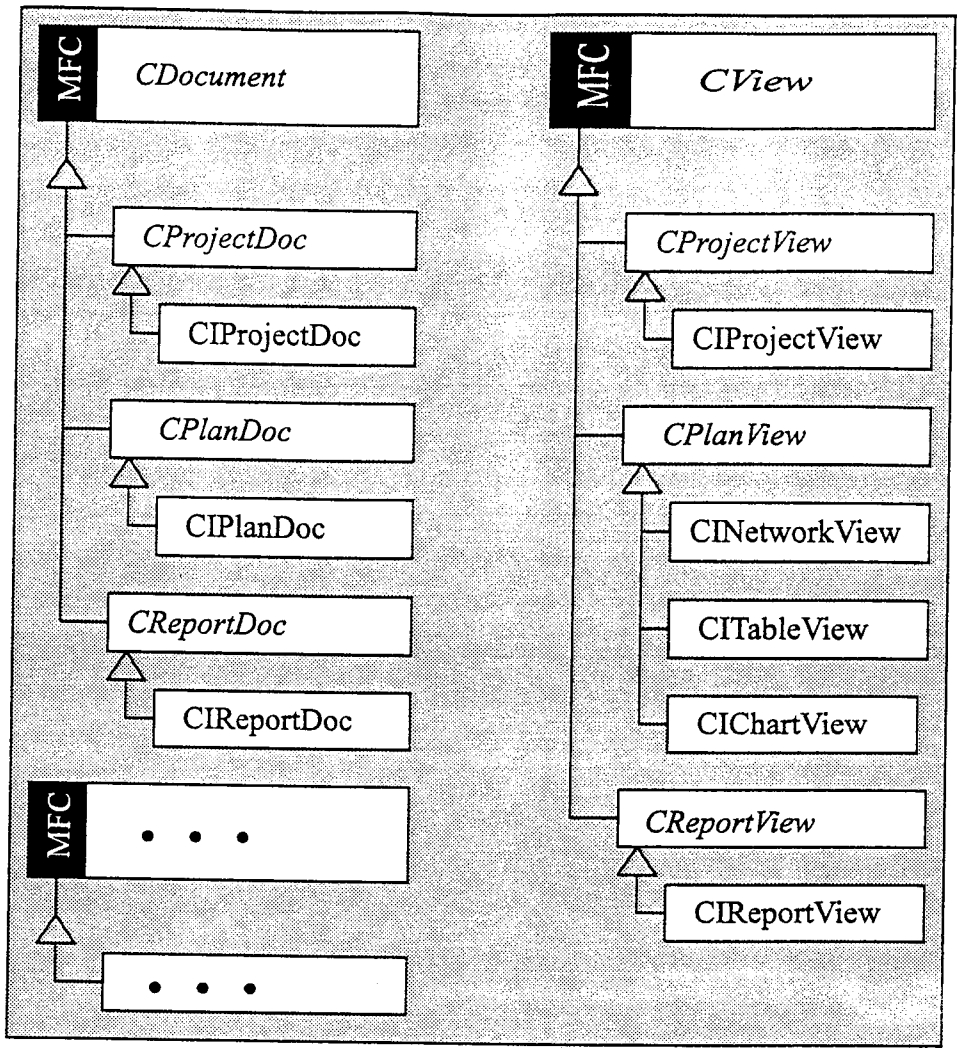

MFC
CObject
CProject
CIProject
CParty
COwner
CContractor
CPerson
CActor
CCostEstimate
CICostEst
CProgressReport
CAction
CPlan
CIPlan
CTask
CITask
CContraint
CITimeConstraint
CIBufferConstraint
CCrew
CSegment
CCost
CDirectCost
CIndirectCost
CCostDurRln
CContinousCDR
CDiscreteCDR
CResource
CEquipment
CLabor
CVersion
CNeuralDynamics
COptimizationProblem
CIOptimizationProblem
CRegularizationNetwork
CQuantity
CConversionRatio
CAddress
CActionReference
Legend
Class Name
Concrete class
Class Name
Abstract class
SuperClass
SubClass
Generalization

Figure 3



Legend

Figure 4

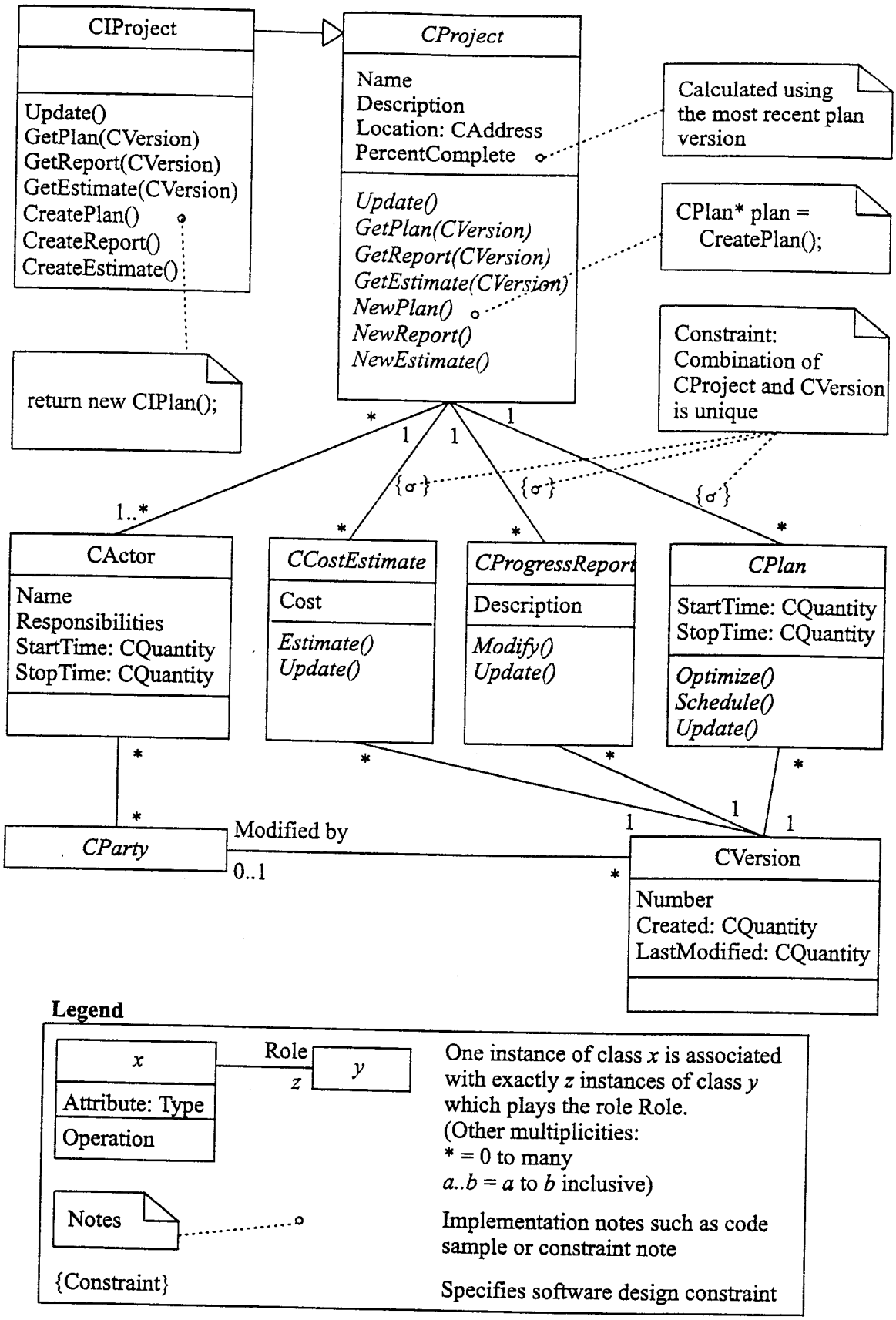

CIProject
Update()
GetPlan(CVersion)
GetReport(CVersion)
GetEstimate(CVersion)
CreatePlan()
CreateReport()
CreateEstimate()
return new CIPlan();
CProject
Name
Description
Location: CAddress
PercentComplete
Update()
GetPlan(CVersion)
GetReport(CVersion)
GetEstimate(CVersion)
NewPlan()
NewReport()
NewEstimate()
Calculated using the most recent plan version
CPlan* plan = CreatePlan();
Constraint: Combination of CProject and CVersion is unique
{σ}
CActor
Name
Responsibilities
StartTime: CQuantity
StopTime: CQuantity
CCostEstimate
Cost
Estimate()
Update()
CProgressReport
Description
Modify()
Update()
CPlan
StartTime: CQuantity
StopTime: CQuantity
Optimize()
Schedule()
Update()
CParty
Modified by
0..1
CVersion
Number
Created: CQuantity
LastModified: CQuantity
Legend
x
Attribute: Type
Operation
Role
z
y
One instance of class x is associated with exactly z instances of class y which plays the role Role.
(Other multiplicities:
* = 0 to many
a..b = a to b inclusive)
Notes
Implementation notes such as code sample or constraint note
{Constraint}
Specifies software design constraint

Figure 5

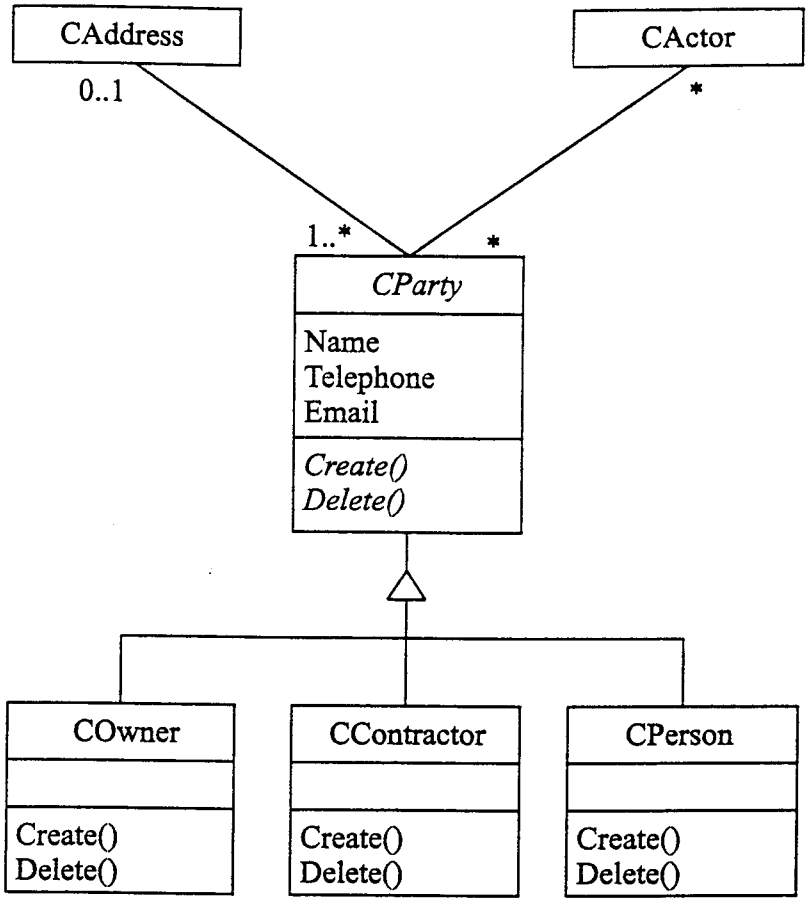
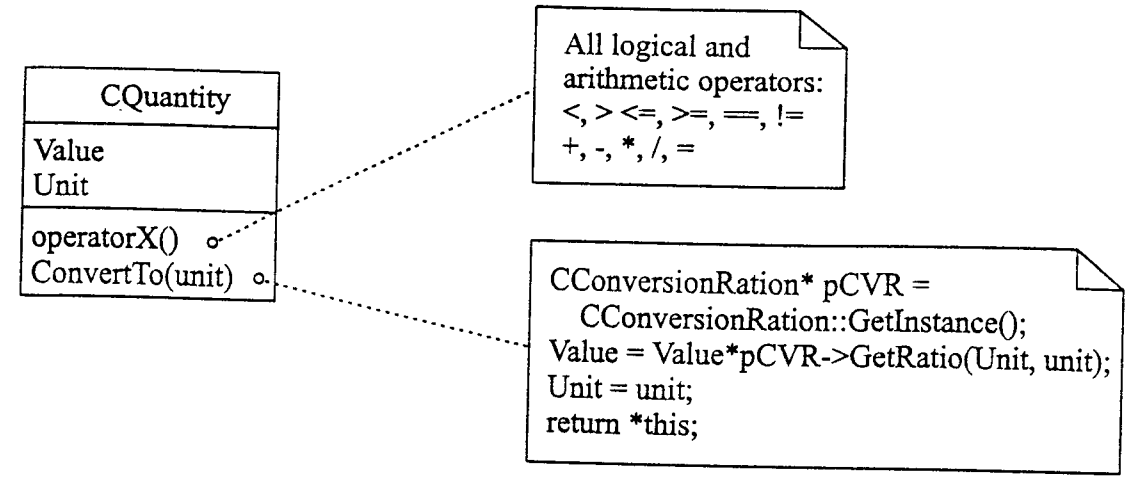
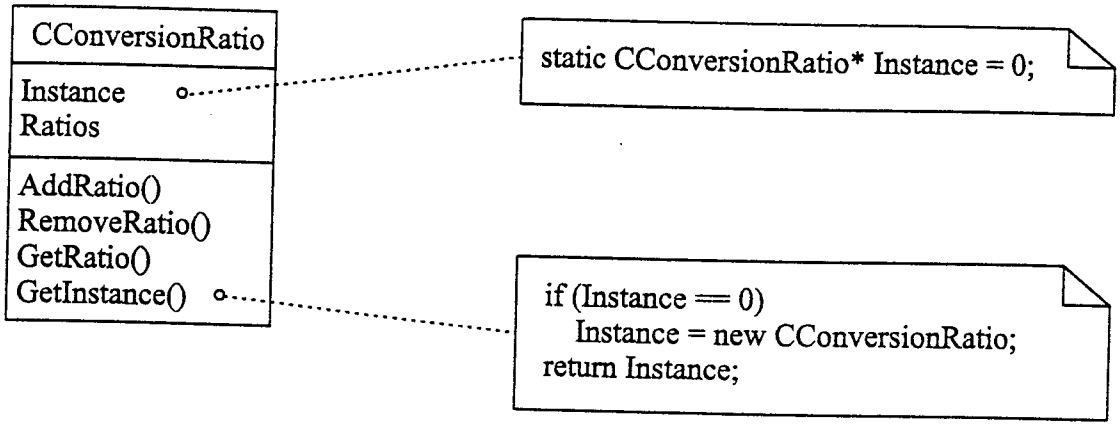

CAddress
CActor
0..1
*
1..*
*
CParty
Name
Telephone
Email
Create()
Delete()
COwner
Create()
Delete()
CContractor
Create()
Delete()
CPerson
Create()
Delete()

Figure 6


CQuantity
Value
Unit
operatorX()
ConvertTo(unit)
All logical and
arithmetic operators:
<, > <=, >=, ==, !=
+, -, *, /, =
CConversionRation* pCVR =
CConversionRation::GetInstance();
Value = Value*pCVR->GetRatio(Unit, unit);
Unit = unit;
return *this;

(a)


CConversionRatio
Instance
Ratios
AddRatio()
RemoveRatio()
GetRatio()
GetInstance()
static CConversionRatio* Instance = 0;
if (Instance == 0)
Instance = new CConversionRatio;
return Instance;

(b)

Figure 7

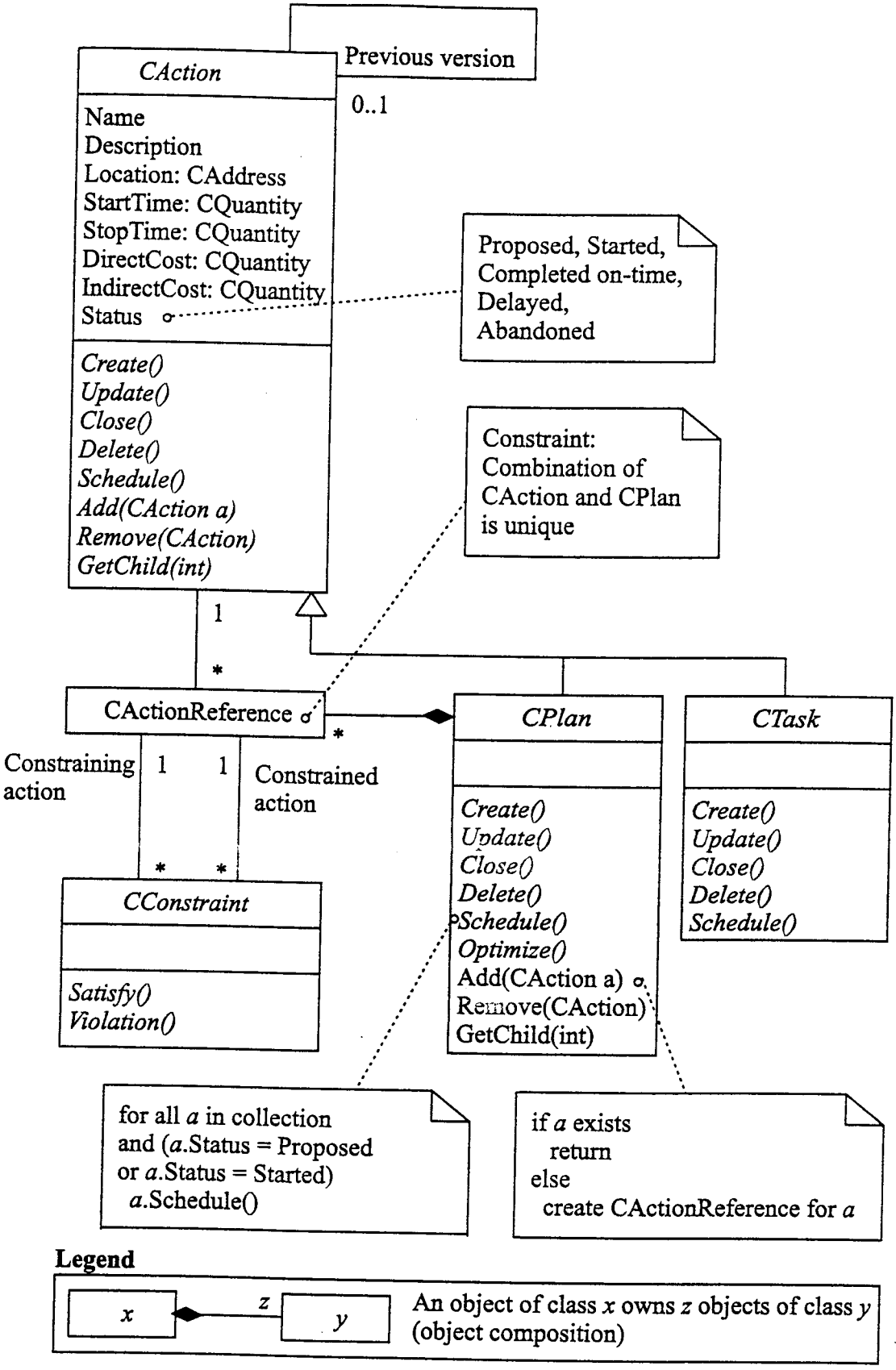

CAction
Name
Description
Location: CAddress
StartTime: CQuantity
StopTime: CQuantity
DirectCost: CQuantity
IndirectCost: CQuantity
Status
Create()
Update()
Close()
Delete()
Schedule()
Add(CAction a)
Remove(CAction)
GetChild(int)
Previous version
0..1
Proposed, Started, Completed on-time, Delayed, Abandoned
Constraint: Combination of CAction and CPlan is unique
1
*
CActionReference
*
Constraining action
1
1
Constrained action
* *
CConstraint
Satisfy()
Violation()
CPlan
Create()
Update()
Close()
Delete()
Schedule()
Optimize()
Add(CAction a)
Remove(CAction)
GetChild(int)
CTask
Create()
Update()
Close()
Delete()
Schedule()
for all a in collection
and (a.Status = Proposed
or a.Status = Started)
a.Schedule()
if a exists
return
else
create CActionReference for a
Legend
x
z
y
An object of class x owns z objects of class y
(object composition)

Figure 8

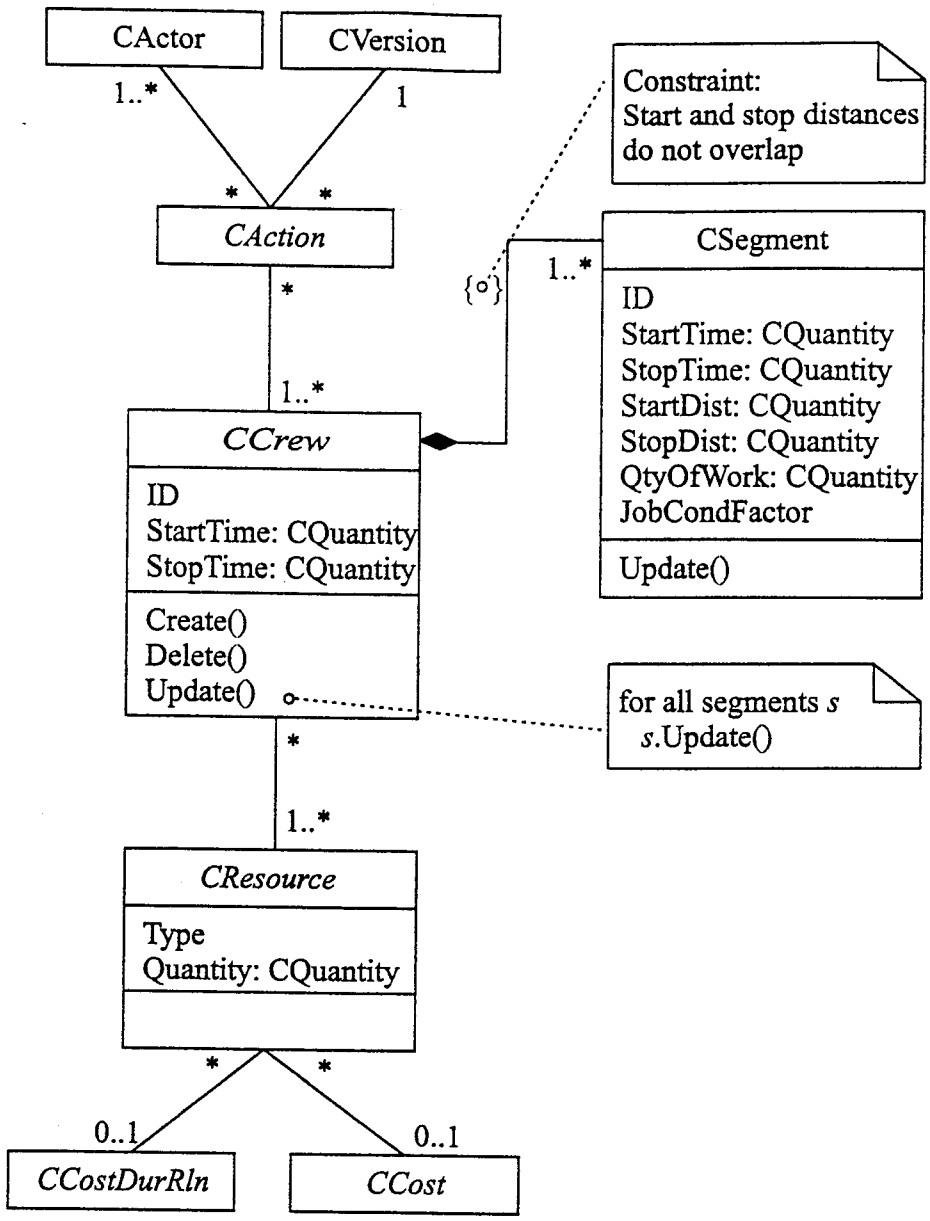

CActor
CVersion
1..*
1
*
*
CAction
*
1..*
CCrew
ID
StartTime: CQuantity
StopTime: CQuantity
Create()
Delete()
Update()
*
1..*
CResource
Type
Quantity: CQuantity
*
*
0..1
0..1
CCostDurRln
CCost
Constraint:
Start and stop distances
do not overlap
{o}
1..*
CSegment
ID
StartTime: CQuantity
StopTime: CQuantity
StartDist: CQuantity
StopDist: CQuantity
QtyOfWork: CQuantity
JobCondFactor
Update()
for all segments s
s.Update()

Figure 9

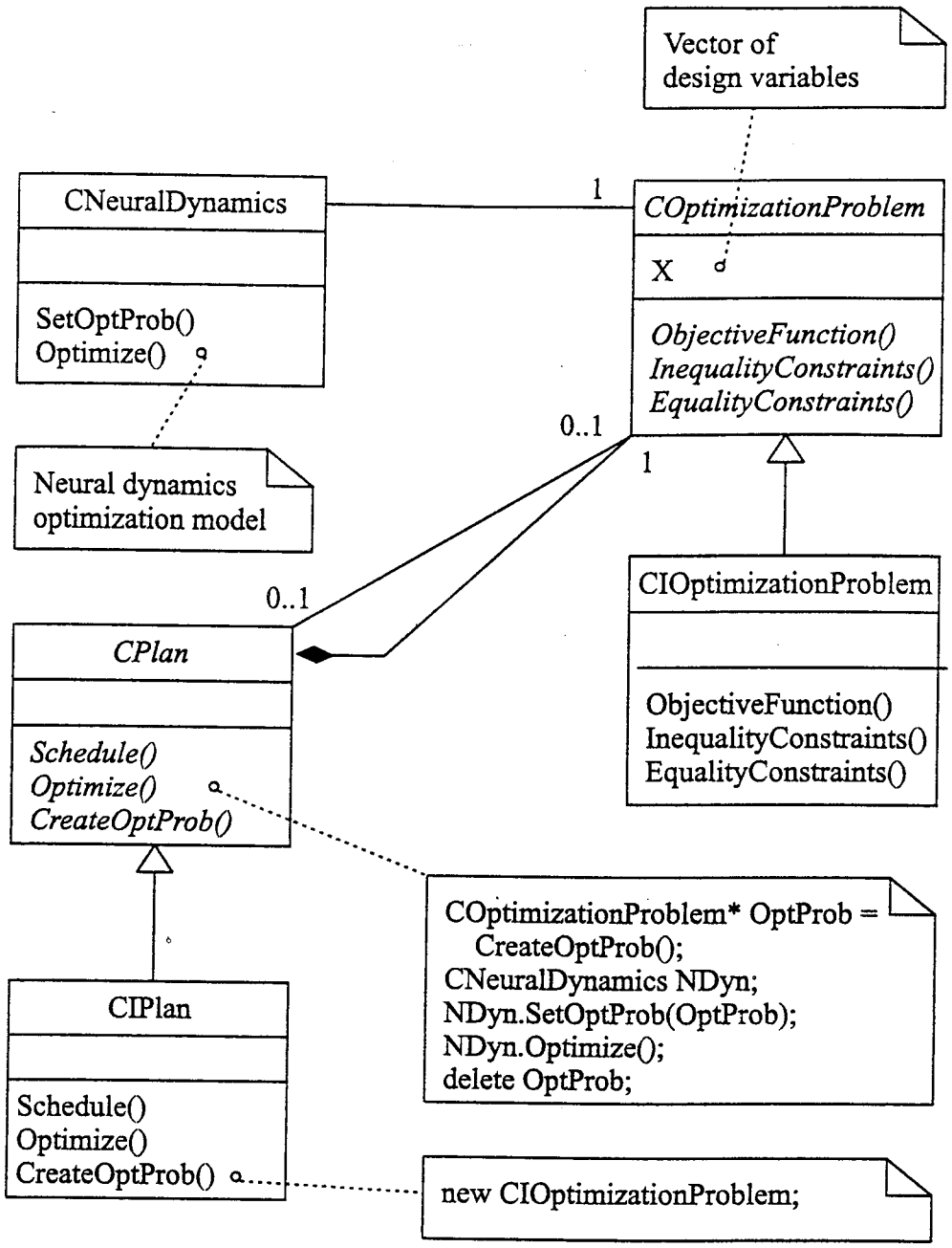

Vector of
design variables
CNeuralDynamics
SetOptProb()
Optimize()
1
COptimizationProblem
X
ObjectiveFunction()
InequalityConstraints()
EqualityConstraints()
0..1
1
Neural dynamics
optimization model
CIOptimizationProblem
ObjectiveFunction()
InequalityConstraints()
EqualityConstraints()
0..1
CPlan
Schedule()
Optimize()
CreateOptProb()
COptimizationProblem* OptProb =
CreateOptProb();
CNeuralDynamics NDyn;
NDyn.SetOptProb(OptProb);
NDyn.Optimize();
delete OptProb;
CIPlan
Schedule()
Optimize()
CreateOptProb()
new CIOptimizationProblem;

Figure 10

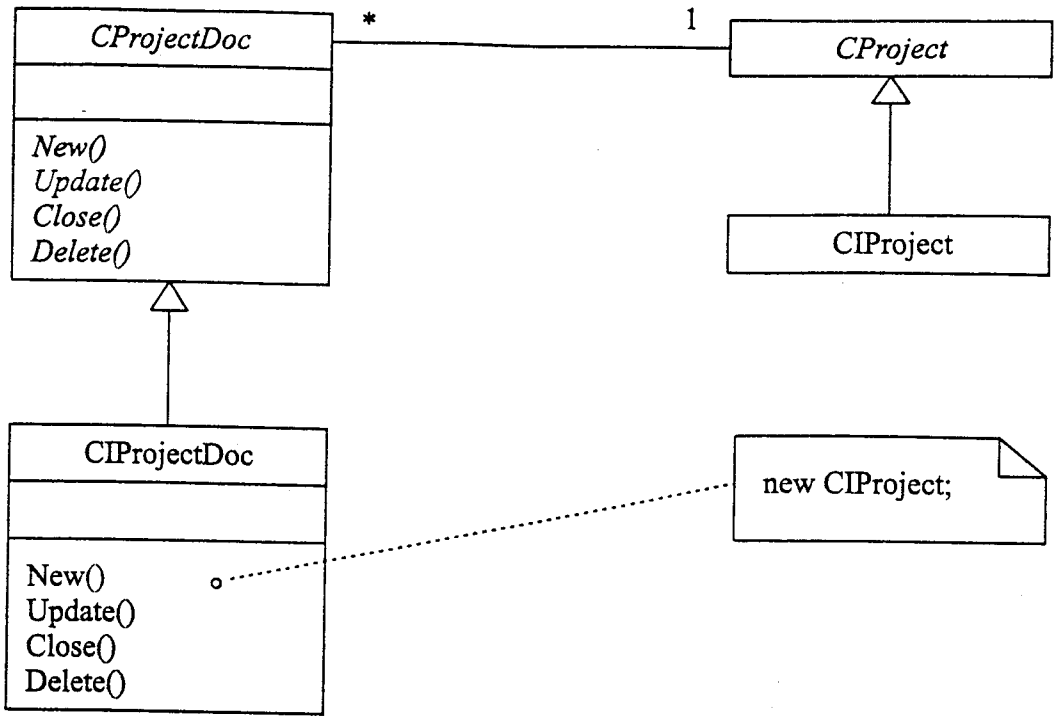

CProjectDoc
*
1
CProject
New()
Update()
Close()
Delete()
CIProject
CIProjectDoc
new CIProject;
New()
Update()
Close()
Delete()

Figure 11

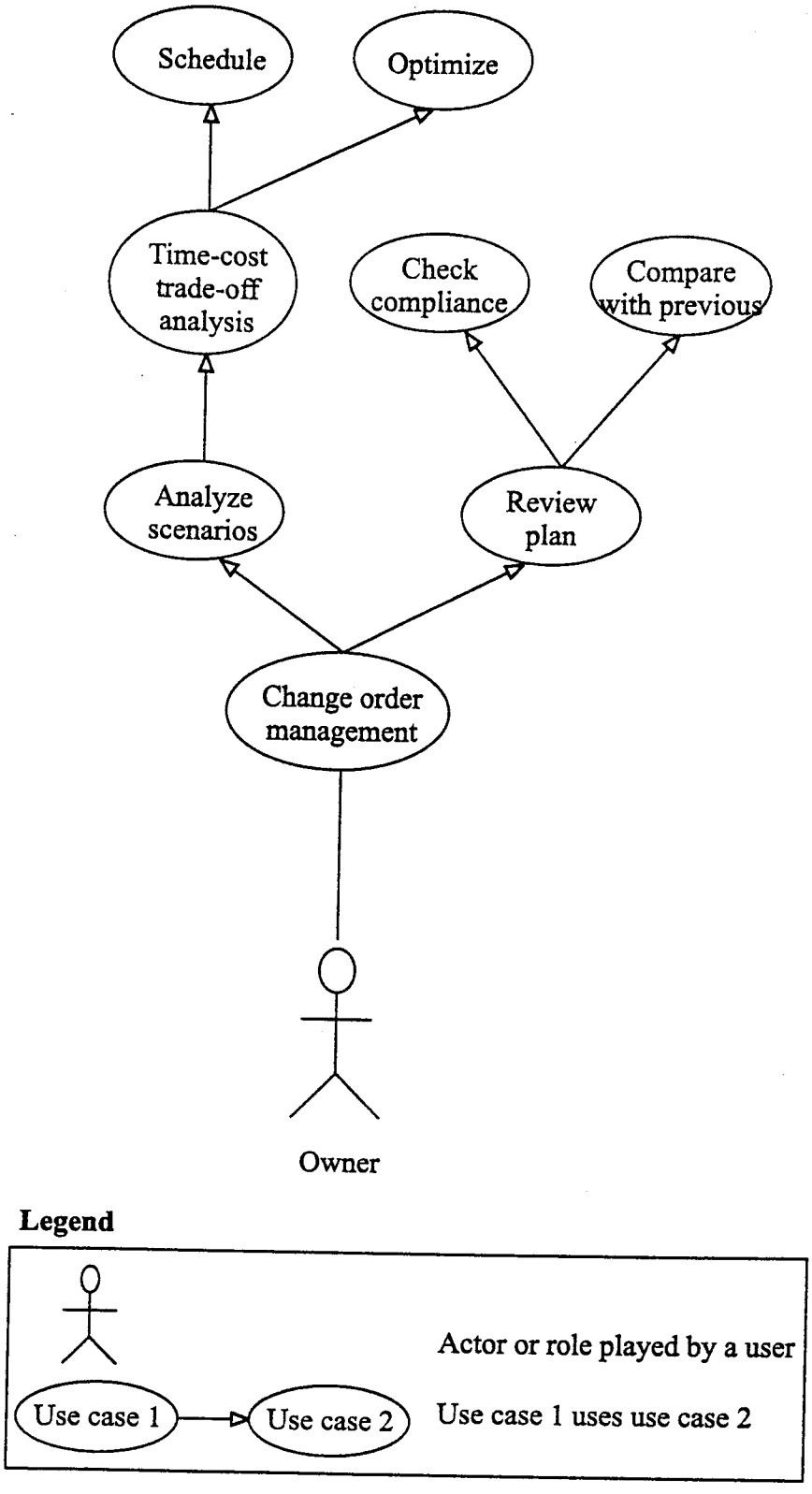

Schedule
Optimize
Time-cost trade-off analysis
Check compliance
Compare with previous
Analyze scenarios
Review plan
Change order management
Owner
Legend
Actor or role played by a user
Use case 1
Use case 2
Use case 1 uses use case 2

Figure 12

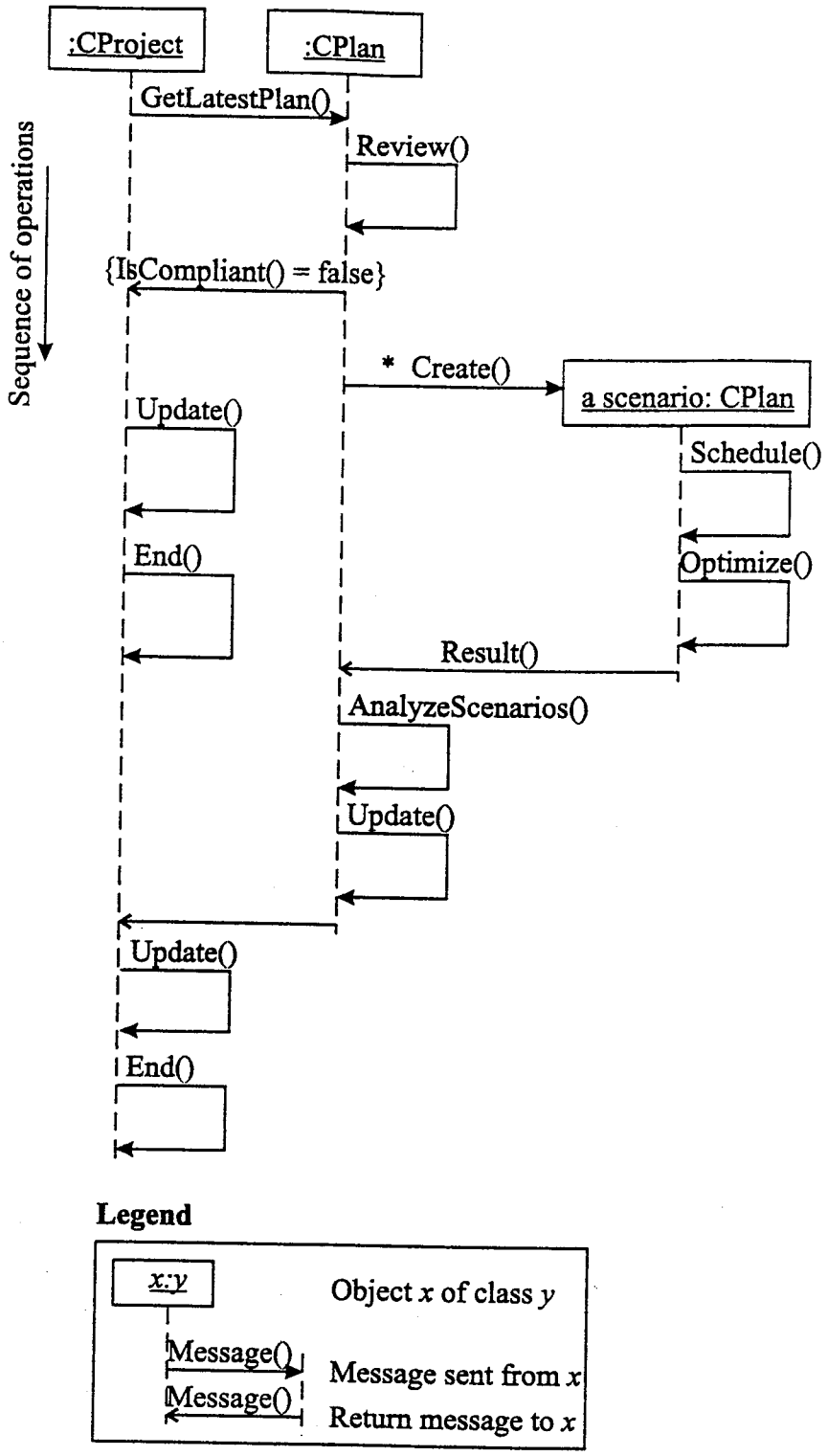

:CProject
:CPlan
GetLatestPlan()
Review()
Sequence of operations
{IsCompliant() = false}
* Create()
a scenario: CPlan
Update()
Schedule()
End()
Optimize()
Result()
AnalyzeScenarios()
Update()
Update()
End()
Legend
x:y
Object x of class y
Message()
Message sent from x
Message()
Return message to x

Figure 13

A NEW GENERATION SOFTWARE FOR CONSTRUCITON SCHEDULING AND MANAGEMENT

Asim Karim[1] and Hojjat Adeli[2]

ABSTRACT

The authors were motivated to overcome some of the limitations and shortcomings of the existing software systems for management of construction projects. The result is a new generation software system for CONstruction Scheduling, Cost Optimization, and Change Order Management, called CONSCOM. CONSCOM uses the recently patented Neural Dynamics model of Adeli and Park[3] as its computational engine for construction cost optimization and advanced software engineering and object-oriented programming techniques such as *framework* and *pattern*. This article presents some of its recent and innovative capabilities and features. CONSCOM includes an integrated management environment (IME) as its user interface for effective control and management of construction projects. An example of a highway construction project is presented to demonstrate the unique modeling capabilities of CONSCOM that cannot be modeled by CPM or CPM-like networks.

[1] Graduate Research Associate, Dept. of Civil and Environmental Engineering and Geodetic Science, The Ohio State University.
[2] Professor, Dept. of Civil and Environmental Engineering and Geodetic Science, The Ohio State University, 2070 Neil Ave., Columbus, OH 43210, USA.
[3] U.S. Patent 5,815,394 issued on September 29, 1998.

INTRODUCTION

The construction schedule is an important document in the construction industry. The construction project participants, such as the contractor and the owner, use the construction schedule to plan, monitor, and control project work. The goal is the completion of the project within budget and time. Further, in recent years the construction schedule has increasingly been used as a legal document in resolving disputes and verifying claims among the project participants. Therefore, the value of the construction schedule cannot be overstated. For its effectiveness, every construction schedule must have two essential characteristics. First, it must be based on an accurate model of the construction project. Second, it must provide features necessary for project control and management. In recent years, the use of construction scheduling software has made the use of schedules in the construction industry widespread. However, the underlying modeling technology used and the process of control and management has not progressed over the years. For example, the critical path method (CPM) is still used despite its documented shortcomings, particularly for projects involving repeating tasks (Adeli and Karim, 1997).

A construction schedule is traditionally defined as the timetable of the execution of tasks in a project. Resources are assigned to the tasks before the project is scheduled. Thus, resources are handled separately and independently of the time. This results in a cost model that is disconnected from the time model thus making cost and time control difficult and imprecise. Further, the scheduled times of the tasks in the project are not updateable in a structured manner to reflect changes that have occurred since the project

started. This makes project monitoring, control, and change order management cumbersome and difficult.

To address these shortcomings we have developed an advanced general construction scheduling model and a prototype software system that fulfils the need for accuracy in modeling and effectiveness in project control and management. It is called CONSCOM (CONstruction Scheduling, Cost Optimization, and Management). This article describes novel features of CONSCOM and its user-interface with particular emphasis on the integration of the modeling, control, and management features.

INTEGRATED CONSTRUCTION SCHEDULING AND COST MANAGEMENT MODEL

Over the years, a number of scheduling techniques have been presented. The motivation for developing new techniques is to overcome some of the shortcomings of the techniques currently in use. The limitations and shortcomings of the existing software systems used in practice are also recognized by the construction industry. For example, a majority of the members of the Associated General Contractors of America are dissatisfied with the critical path method (CPM) (Mattila and Abraham, 1998). Despite these concerns and shortcomings, the CPM is still widely used and none of the new methods presented over the years have gained widespread acceptance. Three reasons can be cited for this lack of acceptance of newer techniques. First, there is the compatibility problem. To ease migration to the new technique it must provide all the modeling capabilities of the existing technique (CPM). If the new technique is a superset of CPM then the initial training and learning cost is minimized as the user can learn the new

capabilities gradually with time. Second, the techniques presented over the last two decades do not provide substantial improvements or advantages to justify their widespread use by the construction industry. Third, there is the issue of ease of use and software maintenance.

To overcome the first two problems, Adeli and Karim (1997) developed a new integrated construction scheduling and cost management model. This model was initially motivated by the need to handle repetitive task projects such as highway construction. However, the model is general in applicability providing a complete set of features, including the four precedence constraint relationships that are considered standard in the CPM. In addition, the new model includes location modeling of tasks with time and distance buffer constraints, resource allocation features such as multiple crew assignment and management, resource allocation that can vary either linearly or nonlinearly with the duration of work, and construction progress tracking, control, and management. The mathematical model incorporates a robust cost minimization algorithm based on the recently patented neural dynamics model of Adeli and Park (1998). This latter feature provides an essential tool to the user for time-cost trade-off analysis and change order management. Although CONSCOM is based on a mathematically rigorous foundation, the scheduling concepts and structures used in this model are generalizations of those in CPM. Therefore, the users of CPM-based software systems can adapt to the new system without significant investment in learning new concepts. However, the solid mathematical foundation makes the implementation unambiguous and serves as a baseline for extension and further development.

OBJECT-ORIENTED MODEL

Software development is often a major cost in the total cost of development and distribution of a new technology. Further, the development of software is incremental and evolutionary in nature with new requirements and features requested by users incorporated into it over time. This in turn requires that the software model be based on a reusable and extensible architecture that can be evolved over time.

A hierarchical software architecture is adopted for the new integrated construction scheduling and cost management model (Karim and Adeli, 1999a). This layered approach separates the key functionality of the system and allows for ease of development and maintenance especially since software components in a higher layer are independent of lower layer components. An object-oriented pattern-based framework is developed to implement the functionality in the layers. These software engineering techniques are fundamental in the development of reusable and extensible software systems.

The prototype software system CONSCOM is based on the object-oriented software architecture using Visual C++ and the Microsoft Foundation Class (MFC) library (Karim and Adeli, 1999b). CONSCOM is designed to run under all 32-bit Windows environments such as Windows 95/98 and Windows NT 4.0. The software presently has over 19,000 lines of code with over 100 classes. Figures 1 and 2 show a subset of classes in CONSCOM, their characteristics (abstract or concrete), and their inheritance hierarchy. Note that the notations in these figures are based on the new standard Unified Modeling language (UML) (Fowler and Scott, 1997). Brief description of classes in Figures 1 and 2 are given in Appendix I. These classes correspond to the *Domain* and *Application* packages, respectively, described in Karim and Adeli (1999a).

The *Domain* package (collection of classes) encapsulates the construction scheduling and cost knowledge. This package is the computational engine of the software system. The *Application* package provides support for application-specific domain knowledge and the application's user interface. Note that Figures 1 and 2 do not show all the classes that support the user interface of CONSCOM.

Most of the classes have the MFC class *CObject* as the root to take advantage of the services it provides for object storage and retrieval. Note that most of the higher level classes are abstract. This is a fundamental design concept in frameworks that allows customization through suclassing. The basic framework (consisting of the abstract classes only) just provides an interface. Derived classes bind the interface to a specific implementation. An abstract class controls which implementation class is instantiated. Classes starting with the prefix 'CI' are implementation classes for construction scheduling, cost optimization, and cost estimation.

FEATURES OF CONSCOM

CONSCOM is an advanced new generation software system for construction scheduling and management that includes a powerful scheduling model, has robust optimization capabilities, and provides strong change management features, all integrated into a compact Integrated Management Environment (IME). Some of the key features of CONSCOM are delineated in the following paragraphs:

- CONSCOM features an advanced construction scheduling model. This model is a superset of all currently available models such as CPM plus new features such as:
  - Integrated construction scheduling and minimum cost model.

- Support for a hierarchical work breakdown structure with tasks, crews, and segments of work.
- Capability to handle multiple-crew strategies.
- Support for location (distance) modeling of work breakdown structures (very useful for modeling linear projects such as highway construction).
- A mechanism to handle varying job conditions.
- Nonlinear and piecewise linear cost modeling capability for work crews.
- Capability to handle time and distance buffer constraints in addition to all the standard precedence relationships.
- Ability to provide construction plan milestone tracking.

- CONSCOM's computational engine is based on the recently patented robust and powerful Neural Dynamics optimization model of Adeli and Park. The Neural Dynamics model of Adeli and Park provides reliable cost minimization of the construction plan, time-cost trade-off analyses, and change order management.
- CONSCOM provides an integrated user-interface with all the tools, capabilities, and information necessary for effective control of construction projects.
- CONSCOM provides a context-sensitive help facility readily available at any point of execution of the software.

INTEGRATED MANAGEMENT ENVIRONMENT (IME)

CONSCOM provides an integrated user interface for effective construction management and control. The interface is an integrated management environment (IME) providing ready access to all the tools and information needed to plan, monitor, analyze,

and control construction projects. Figure 3 shows a screen shot of CONSCOM's main window. Three types of output display windows are used to provide information to the user. The primary output display is the Plan View window (the two windows on the top-right in Figure 3). Each of these windows displays the details for a specific plan. All the tasks are listed together with their start time, stop time, duration, and cost. The two-part icon in front of each row indicates the status or state of the task. Two different colors are used for the two parts of the icon to provide visual information as to whether the task is a proposed task or an implemented task (a task currently in progress).

Multiple plans may be opened in CONSCOM at a single time. All the plans that are open plus all the plans that were previously open in the current CONSCOM session are listed in the output display called the Project Workspace window (left window in Figure 3). This output display lists all the plans in the workspace with their current cost and duration values. The Project Workspace window is the project management and control interface. The information available from this window facilitates the owner or the contractor to keep track of all the plans in the project. Further, if CONSCOM is used for change order management then this window allows one step access to all the versions of the plan enabling quick and effective decision making. Similar to the Plan View windows the two-part icon in front of each plan in the Project Workspace window indicates the status or state of the plan.

The third output display is the Task Details window (the bottom window in Figure 3). This window shows the detailed information for each work breakdown structure of the selected task. The information in this window is especially useful for multiple-crew and multiple-segment tasks. The information displayed includes the start

time, stop time, duration, and cost for the work of each crew and each segment of the work of each crew. In addition, for each segment of the work, the start and stop locations, the quantity of work, and the job condition factor are also displayed. Further information about each task that is less frequently required is provided in dialog boxes (an input/output window that is usually not resizable). Figure 4 shows the Modify Task property dialog that allows modification and display access to all the information about a selected task. Certain information such as the constraints, cost-duration relationship, and descriptions of crews is available from this property dialog only.

USER INTERFACE CHARACTERISTICS

A highway construction project is used to demonstrate the modeling capabilities of CONSCOM (Adeli and Karim, 1997). It uses some of the new scheduling features provided by the model. This plan cannot be modeled by CPM or CPM-like networks currently available in commercial packages. In this section, we describe the handling of these features by CONSCOM's user interface.

The work for the construction of the 2-lane 5-km long highway is divided into 7 repetitive and 7 non-repetitive tasks. This project requires the following modeling features: multiple crews, multiple segments of work per crew, location modeling of work, distance and time buffer constraints, work continuity constraints, and job condition factors. Both linear and nonlinear relationships are used to describe the direct cost-duration relationship of crews. A minimum-duration plan generated by CONSCOM is shown in Figure 5. Note that in CONSCOM no distinction is made between a repetitive

and a non-repetitive task. The differentiation is only for user convenience. Computationally, both types of tasks are handled similarly by CONSCOM.

To illustrate the data input process and show some of the modeling capabilities of CONSCOM consider adding a new task 10 to the plan. In CONSCOM, entry and modification of data for a task is handled by four dialog boxes. When adding a new task these dialog boxes are presented in a logical sequence. When the user is modifying a selected task all the dialog boxes are presented simultaneously. The four dialog boxes showing the input data required for task 10 are shown in Figures 6a-6d. General information of the task is entered in the General dialog box (Figure 6a). In this figure, ID is an alphanumeric string that uniquely identifies the task in the plan. The description and comments are optional fields. In this example, the task is a proposed task.

However, if progress of work is being monitored or change order scenarios are studied a corresponding implemented task can be added by selecting the appropriate button. An implemented task may or may not have a corresponding proposed task. For example, the contractor may add and start working on a new task because of an unexpected working condition on the field which was not anticipated in the original proposed plan. This will require a new implemented task that has no corresponding task in the proposed plan. For further classification of the state of an implemented task the task can be set as either one that is in progress or one that is finished. When a task has the implemented status its start time and cost are fixed and it will not take part in scheduling and cost optimization.

The work crew information for the task is entered in the Crews dialog box (Figure 6b). Each task can have multiple crews that are uniquely identified by a numeric value.

Associated with each crew is a direct cost-duration relationship. The default relationship is a two-point linear relationship corresponding to the minimum cost (maximum duration) and maximum cost (minimum duration) data points. If a piecewise linear or a nonlinear relationship is desired then it can be specified in the Cost-duration relationship dialog box (Figure 7). The piecewise linear relationship is defined by a finite number of cost-duration data points which are then connected by straight lines.

The nonlinear relationship is modeled in the following form:

$$C(d) = \frac{f(d)}{g(d)}$$

where $C(d)$ is the cost of completing work in duration d, and $f(.)$, $g(.)$ are the numerator and the denominator fourth order polynomial expressions, respectively. In the present example, task 10 has two crews and both of them have the identical nonlinear relationship C(d) = (1600 + d)/d where C(d) yields the cost for completing work in duration d. The cost-duration relationship is bounded by the minimum and maximum duration values that have to be specified.

The third input dialog box, called Segments dialog box (Figure 6c), gathers information for the segments of work for each crew. Each segment is uniquely identified for each crew by a numeric value. The breakdown of work into segments is necessary to capture the changed conditions at each stage of the crew's work. These variations may include the difference in location, change in quantity of work, or a better or worse work condition than originally expected. In CONSCOM, locations are modeled using distances. This is useful in linear projects such as highway construction where it is necessary to track the location of work not only for progress monitoring purposes but also to ensure that sufficient distances are maintained between tasks. Distance buffers are

required when, for example, sufficient space is required for equipment and labor to perform optimally and safely. When distance modeling is not needed the location fields should be set to zero. This does not mean that location is not considered in modeling but locations of work are not defined by linear distance.

Task 10 has 2 crews and 2 segments of work per crew (Figures 6b and c). Crew 1 works over the distance 1000 to 3500 m while crew 2 works over the distance 3500 to 6000 m. The work of crew 1 is broken down into two segments from 1000 to 3000 m and from 3000 to 3500m (Figure 6c). This breakdown is done to model the more difficult job conditions in the 3000-3500 m section. This segment is assigned a job condition factor of 1.15 that reflects a 15% increase in time needed to move a unit quantity of earth as compared to that required in the first segment of work. Another reason to breakdown the work of a crew is when the quantity of work required per unit length of the highway is not constant. Segments of work are selected so that the quantity of work per unit length in each segment remains roughly constant.

The last input dialog box, called the Relative Constraint dialog box (Figure 6d), is used to specify constraints on the task. CONSCOM supports all the standard precedence constraints. In addition, it also supports time and distance buffer constraints. A constraint can be specified on any segment of the task, on any crew of the task, or on the whole task. Similarly, the constraining element can be any segment, crew, or task. This flexibility in specifying constraints allows multiple-crew strategies, work continuity, and other resource-based constraints to be modeled effectively. Each constraint can also have a time or distance lag value. The two crews of task 10 have a start-to-start relationship. With this constraint both crews will start work at the same time. Also, note that this

constraint is specified as a binding constraint. This means the constraint is an equality constraint. On the other hand, a non-binding constraint is an inequality constraint that is satisfied as long as its value is less than or equal to zero. For example, the 150 m distance buffer constraint between task 10 and task 9 is non-binding. This ensures a minimum distance of 150 m (not exactly 150 m) is maintained between the work crews of task 10 and task 9. The two segments of work of each crew have a work continuity constraint to ensure continuity of work.

Data in CONSCOM can be entered in any appropriate unit. The user can define a default set of units for each plan or project. These are the units in which all output is displayed. However, at all the data entry points a set of appropriate units is available to the user to choose from (see Figures 6b-d). CONSCOM will convert the entered data from the specified unit to the default unit automatically.

In CONSCOM a plan can be scheduled in three ways. First, the plan may be scheduled so that all tasks are completed with minimum duration. Second, the plan may be scheduled so that all tasks are completed with minimum cost. Third, the plan may be scheduled for given fixed durations of tasks. The example schedule shown in Figure 5 is for minimum duration.

CONCLUDING REMARKS

CONSCOM is based on an advanced integrated construction scheduling and cost model. Furthermore, it is developed using the latest software engineering techniques. Future developments of the system will focus on adding resource-leveling capability,

improvement of the user and inter-program interface, and refinement of the underlying software framework so that it can be customized easily for other applications.

ACKNOWLEDGMENT

This manuscript is based on a research project sponsored by the Ohio Department of Transportation and Federal Highway Administration.

APPENDIX I. BRIEF DESCRIPTION OF CLASSES IN THE CONSCOM FRAMEWORK (FIGURES 1 AND 2)

CAction: Provides an interface for managing actions (tasks and plans).

CActionReference: Encapsulates a reference to a *CAction* object.

CActor: Encapsulates a role played by a person or organization.

CAddress: Encapsulates a street address.

CConstraint: Encapsulates a construction scheduling constraint.

CContinuousCDR: Encapsulates a continuous cost duration relation.

CContractor: Abstracts a contractor of a construction project.

CConversionRatio: Encapsulates rations for the conversion of values from one unit to another.

CCost: Provides an interface for managing construction costs.

CCostDurRln: Encapsulates a cost duration relation.

CCostEstimate: Provides an interface for managing construction cost estimates.

CCrew: Abstracts a construction crew.

CDirectCost: Provides support for construction direct cost management.

CDiscreteCDR: Encapsulates a discrete cost duration relation.

CEquipment: Abstracts a construction equipment.

CI: Class names starting with 'CI' are implementation classes for the corresponding class name starting with 'C' in CONSCOM.

CIBufferConstraint: Provides support for distance or buffer constraints.

CIndirectCost: Provides support for construction indirect cost management.

CITimeConstraint: Provides support for time constraints.

CLabor: Abstracts information of construction labor.

CNeuralDynamics: Abstracts the neural dynamics model for the solution of optimization problems.

COptimizationProblem: Provides an interface for the solution of optimization problems using the neural dynamics model.

COwner: Abstracts an owner of a construction project.

CParty: Base class of *CContractor*, *COwner*, and *CPerson* (encapsulates their common features).

CPerson: Abstracts a person.

CPlan: Provides an interface for managing construction plans.

CPlanDoc: Provides an interface for a construction application document.

CPlanView: Provides an interface for a construction plan application view.

CProgressReport: Provides support for managing progress reports.

CProject: Provides an interface for managing construction projects.

CProjectDoc: Provides an interface for a construction project application document.

CProjectView: Provides an interface for a construction project application view.

CQuantity: Encapsulates a measurement value and its unit.

CRegularizationNetwork: Abstracts the regularization neural network model for cost estimation.

CReportDoc: Provides an interface for a construction progress report application. document.

CReportView: Provides an interface for a construction progress report application view.

CResource: Abstracts a construction resource.

CSegment: Abstracts a segment of construction work.

CTask: Provides an interface for managing construction tasks.

CVersion: provides support for version control.

APPENDIX II. REFERENCES

Adeli, H. and Karim, A. (1997), "Scheduling/Cost Optimization and Neural Dynamics Model for Construction," *Journal of Construction Engineering and Management*, ASCE, Vol. 123, No. 4, pp. 450-458.

Adeli, H. and Park, H. S. (1998), *Neurocomputing for Design Automation*, CRC Press, Boca Raton, FL.

Fowler, M. and Scott, K (1997), *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley Longman, Inc., Reading, MA.

Karim, A. and Adeli, H. (1999a), "Object-Oriented Information Model for Construction Project Management," *Journal of Construction Engineering and Management*, ASCE, Vol. 125, Accepted for publication.

Karim, A. and Adeli, H. (1999b), "CONSCOM: An OO Construction Scheduling and Change Management System," *Journal of Construction Engineering and Management*, ASCE, Vol. 125, Accepted for publication.

Mattila, K. G. and Abraham, D. M. (1998), "Linear Scheduling: Past Research Efforts and Future Directions," *Engineering Construction and Architectural Management*, Vol. 5, No. 3, pp. 294-303.

LIST OF CAPTIONS FOR FIGURES

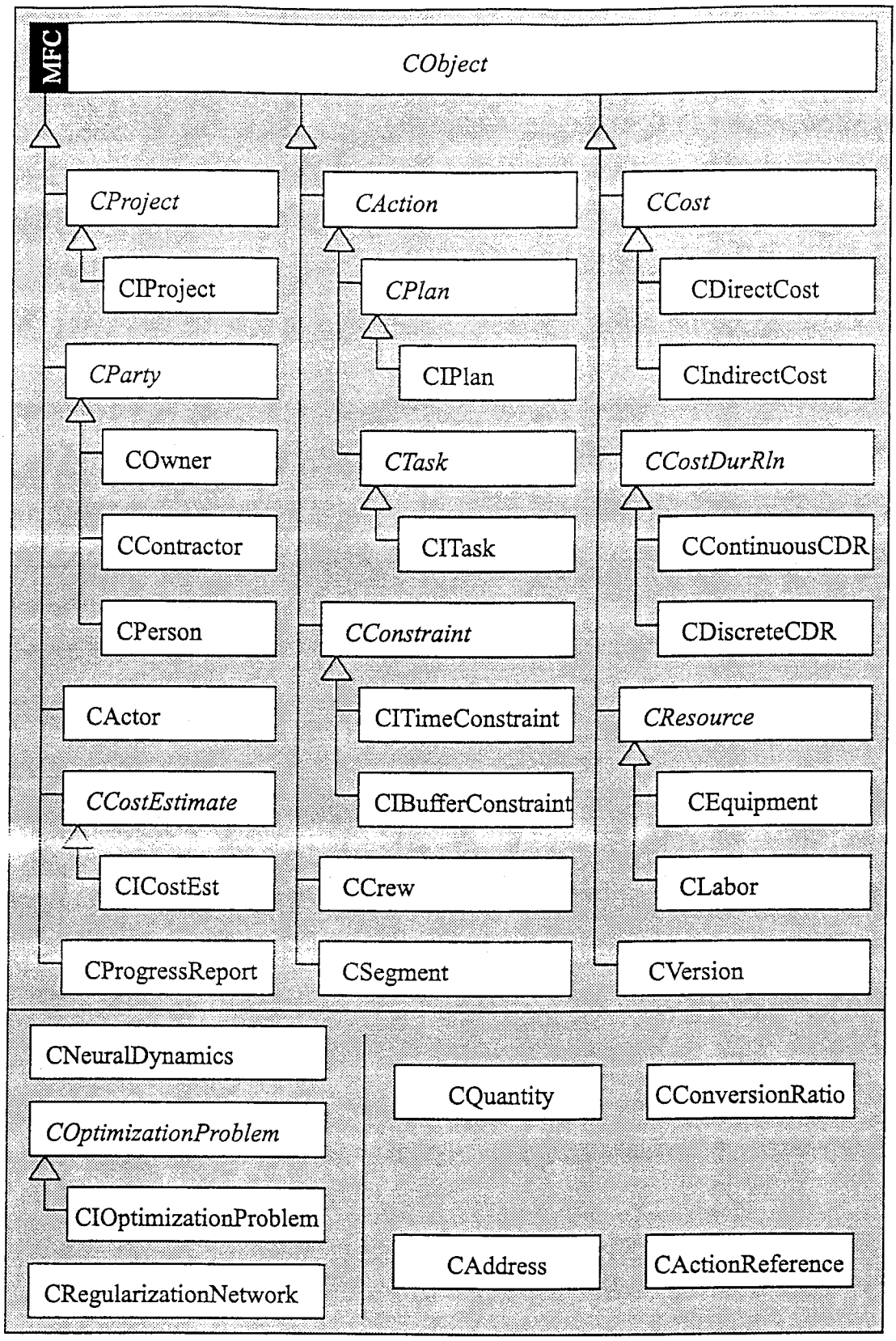

MFC
CObject
CProject
CIProject
CParty
COwner
CContractor
CPerson
CActor
CCostEstimate
CICostEst
CProgressReport
CAction
CPlan
CIPlan
CTask
CITask
CConstraint
CITimeConstraint
CIBufferConstraint
CCrew
CSegment
CCost
CDirectCost
CIndirectCost
CCostDurRln
CContinuousCDR
CDiscreteCDR
CResource
CEquipment
CLabor
CVersion
CNeuralDynamics
COptimizationProblem
CIOptimizationProblem
CRegularizationNetwork
CQuantity
CConversionRatio
CAddress
CActionReference
Legend
Class Name
Concrete class
Class Name
Abstract class
Superclass
Subclass
Generalization

Figure 1

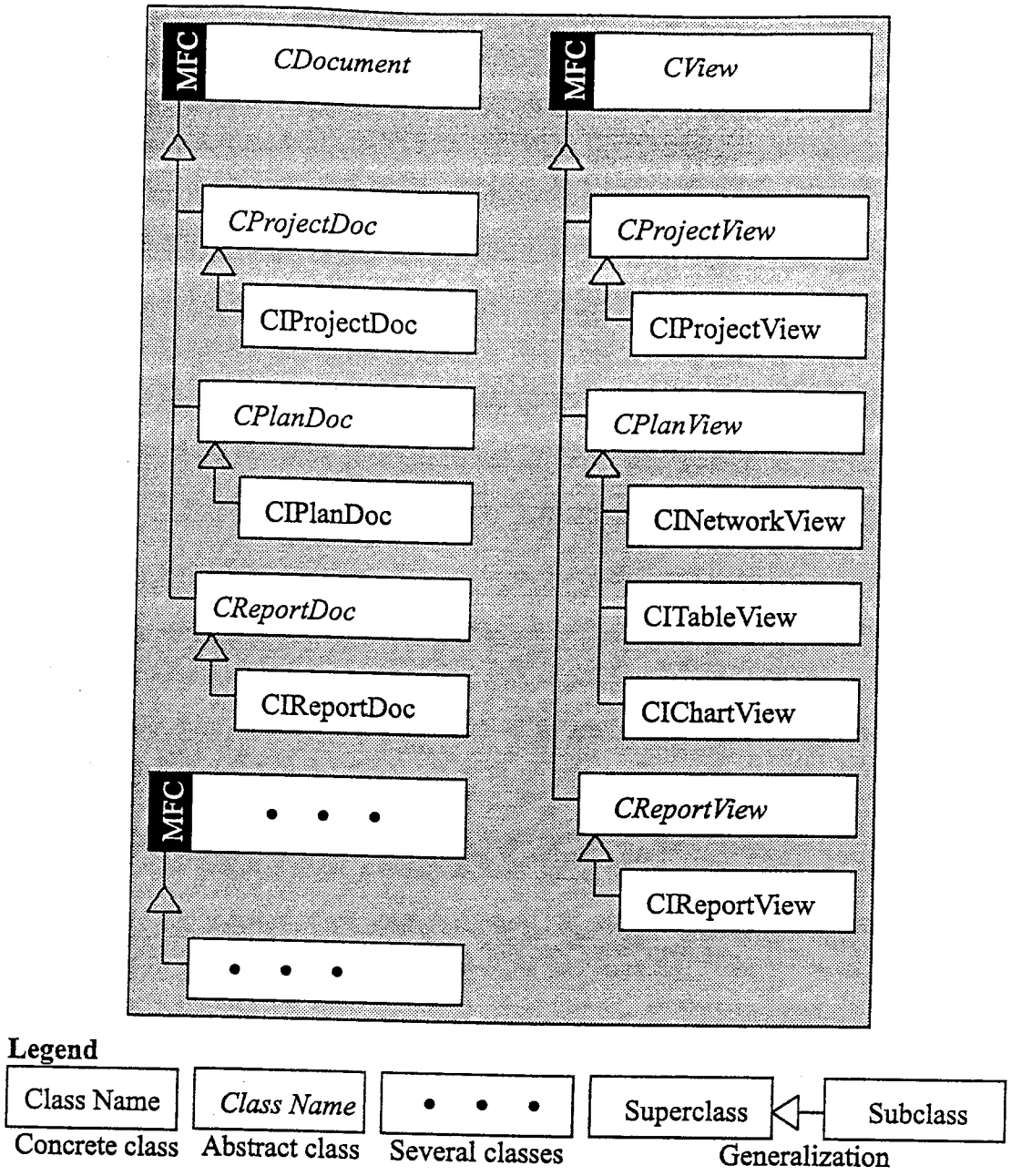

MFC
CDocument
MFC
CView
CProjectDoc
CProjectView
CIProjectDoc
CIProjectView
CPlanDoc
CPlanView
CIPlanDoc
CINetworkView
CReportDoc
CITableView
CIReportDoc
CIChartView
MFC
CReportView
CIReportView
Legend
Class Name
Concrete class
Class Name
Abstract class
Several classes
Superclass
Subclass
Generalization

Figure 2

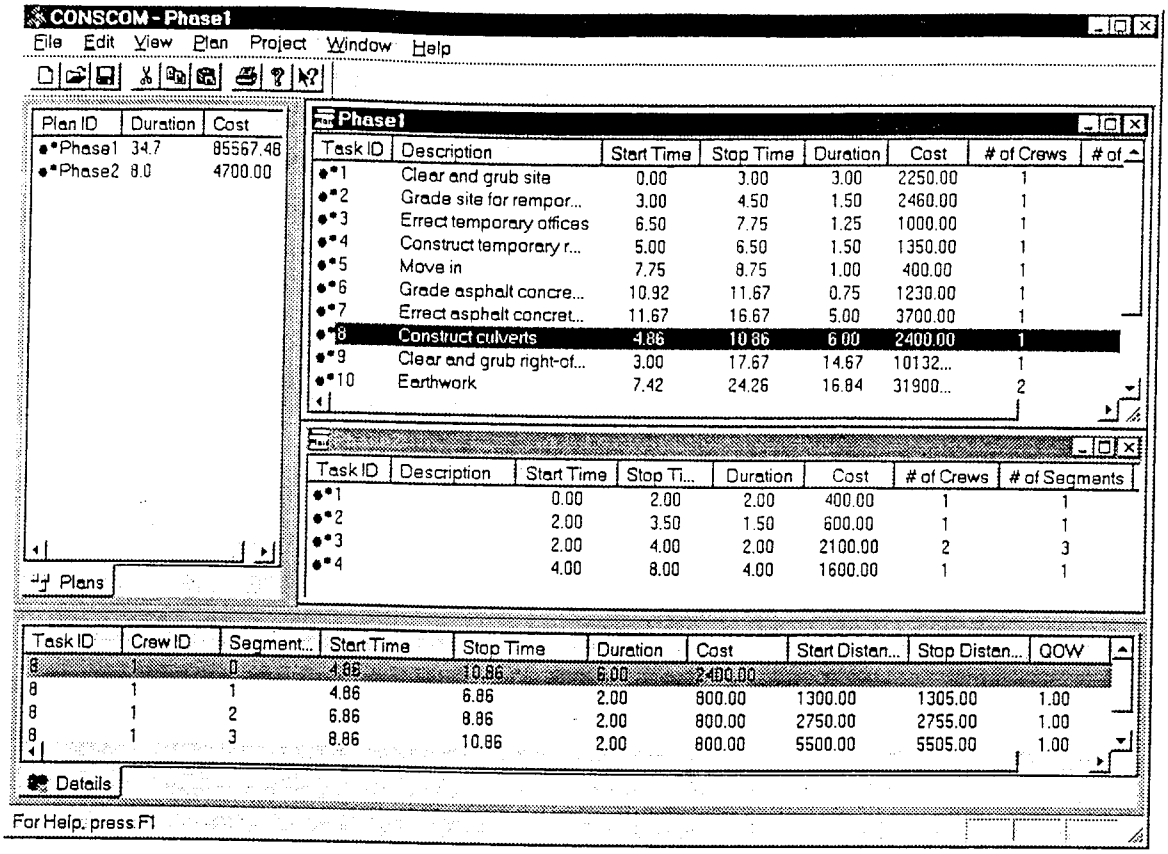

CONSCOM - Phase1
File Edit View Plan Project Window Help
Plan ID Duration Cost
Phase1 34.7 85567.48
Phase2 8.0 4700.00
Plans
Phase1
Task ID Description Start Time Stop Time Duration Cost # of Crews # of
1 Clear and grub site 0.00 3.00 3.00 2250.00 1
2 Grade site for rempor... 3.00 4.50 1.50 2460.00 1
3 Erect temporary offices 6.50 7.75 1.25 1000.00 1
4 Construct temporary r... 5.00 6.50 1.50 1350.00 1
5 Move in 7.75 8.75 1.00 400.00 1
6 Grade asphalt concre... 10.92 11.67 0.75 1230.00 1
7 Erect asphalt concret... 11.67 16.67 5.00 3700.00 1
8 Construct culverts 4.86 10.86 6.00 2400.00 1
9 Clear and grub right-of... 3.00 17.67 14.67 10132... 1
10 Earthwork 7.42 24.26 16.84 31900... 2
Task ID Description Start Time Stop Ti... Duration Cost # of Crews # of Segments
1 0.00 2.00 2.00 400.00 1 1
2 2.00 3.50 1.50 600.00 1 1
3 2.00 4.00 2.00 2100.00 2 3
4 4.00 8.00 4.00 1600.00 1 1
Task ID Crew ID Segment... Start Time Stop Time Duration Cost Start Distan... Stop Distan... QOW
8 1 0 4.86 10.86 6.00 2400.00
8 1 1 4.86 6.86 2.00 800.00 1300.00 1305.00 1.00
8 1 2 6.86 8.86 2.00 800.00 2750.00 2755.00 1.00
8 1 3 8.86 10.86 2.00 800.00 5500.00 5505.00 1.00
Details
For Help, press F1

Figure 3

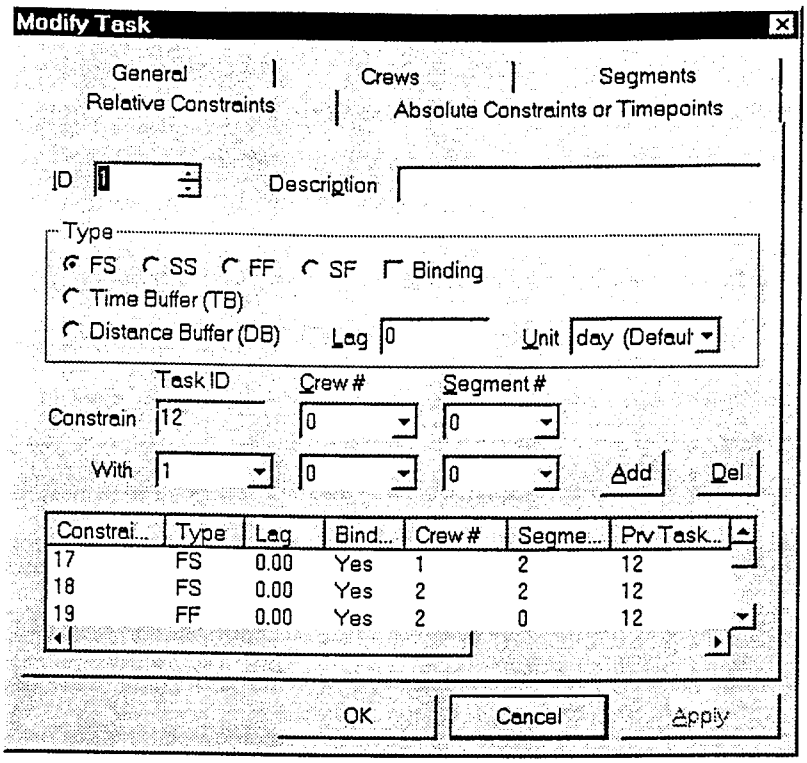

Modify Task
General
Crews
Segments
Relative Constraints
Absolute Constraints or Timepoints
ID 1
Description
Type
FS SS FF SF Binding
Time Buffer (TB)
Distance Buffer (DB)
Lag 0
Unit day (Defaul
Task ID
Crew #
Segment #
Constrain 12 0 0
With 1 0 0
Add
Del
Constrai... Type Lag Bind... Crew # Segme... Prv Task...
17 FS 0.00 Yes 1 2 12
18 FS 0.00 Yes 2 2 12
19 FF 0.00 Yes 2 0 12
OK
Cancel
Apply

Figure 4

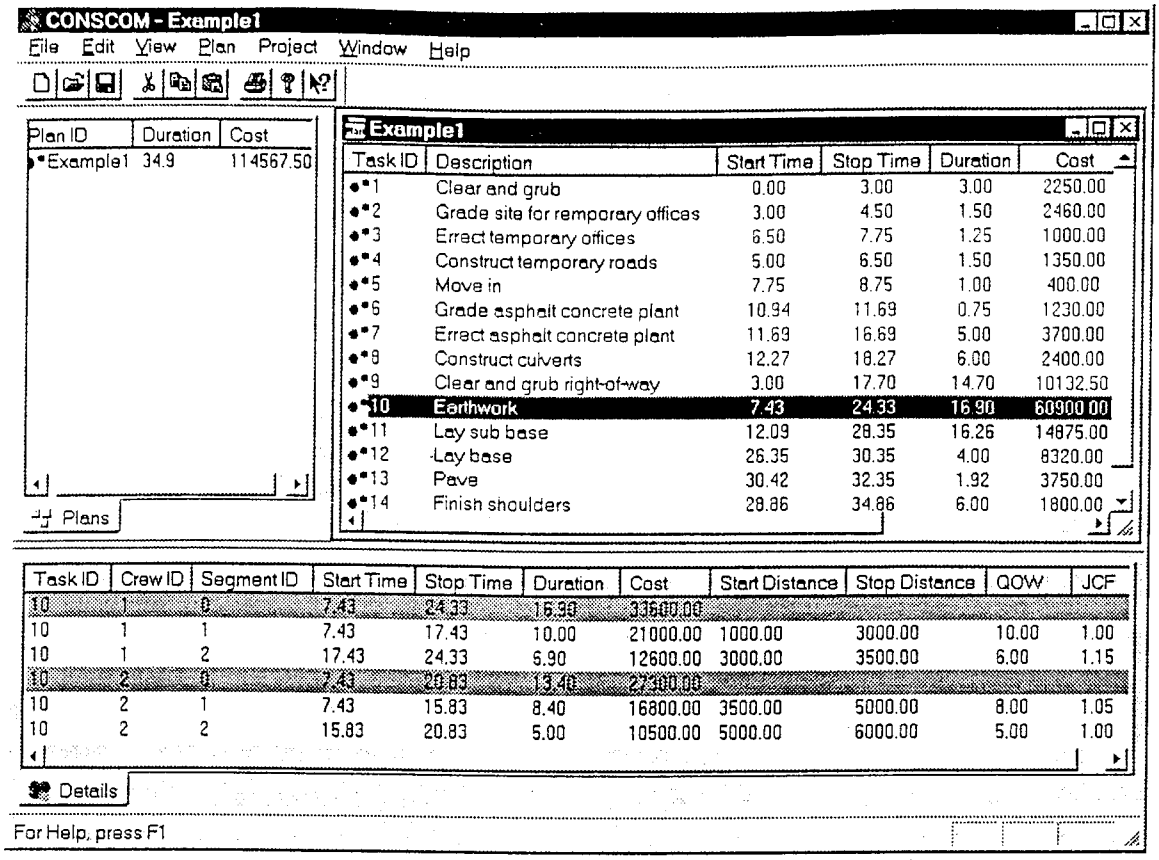

CONSCOM - Example1
File Edit View Plan Project Window Help
Plan ID Duration Cost
Example1 34.9 114567.50
Plans
Example1
Task ID Description Start Time Stop Time Duration Cost
1 Clear and grub 0.00 3.00 3.00 2250.00
2 Grade site for remporary offices 3.00 4.50 1.50 2460.00
3 Errect temporary offices 6.50 7.75 1.25 1000.00
4 Construct temporary roads 5.00 6.50 1.50 1350.00
5 Move in 7.75 8.75 1.00 400.00
6 Grade asphalt concrete plant 10.94 11.69 0.75 1230.00
7 Errect asphalt concrete plant 11.69 16.69 5.00 3700.00
8 Construct culverts 12.27 18.27 6.00 2400.00
9 Clear and grub right-of-way 3.00 17.70 14.70 10132.50
10 Earthwork 7.43 24.33 16.90 60900.00
11 Lay sub base 12.09 28.35 16.26 14875.00
12 Lay base 26.35 30.35 4.00 8320.00
13 Pave 30.42 32.35 1.92 3750.00
14 Finish shoulders 28.86 34.86 6.00 1800.00
Task ID Crew ID Segment ID Start Time Stop Time Duration Cost Start Distance Stop Distance QOW JCF
10 1 0 7.43 24.33 16.90 33600.00
10 1 1 7.43 17.43 10.00 21000.00 1000.00 3000.00 10.00 1.00
10 1 2 17.43 24.33 6.90 12600.00 3000.00 3500.00 6.00 1.15
10 2 0 7.43 23.83 15.40 27300.00
10 2 1 7.43 15.83 8.40 16800.00 3500.00 5000.00 8.00 1.05
10 2 2 15.83 20.83 5.00 10500.00 5000.00 6000.00 5.00 1.00
Details
For Help, press F1

Figure 5

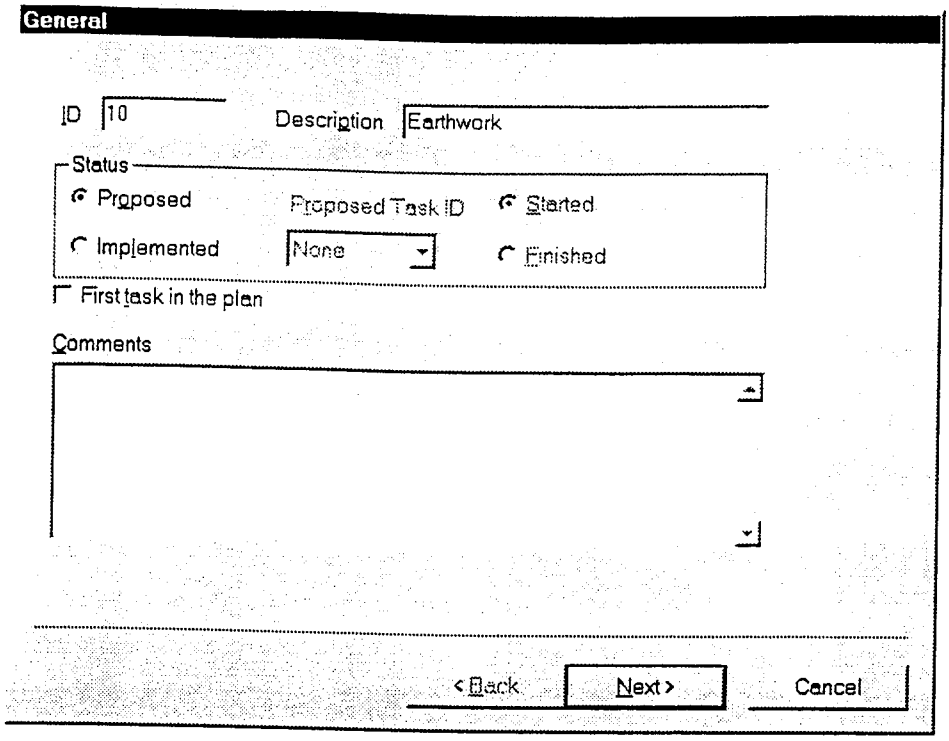

General
ID
10
Description
Earthwork
Status
Proposed
Proposed Task ID
Started
Implemented
None
Finished
First task in the plan
Comments
< Back
Next >
Cancel

Figure 6a

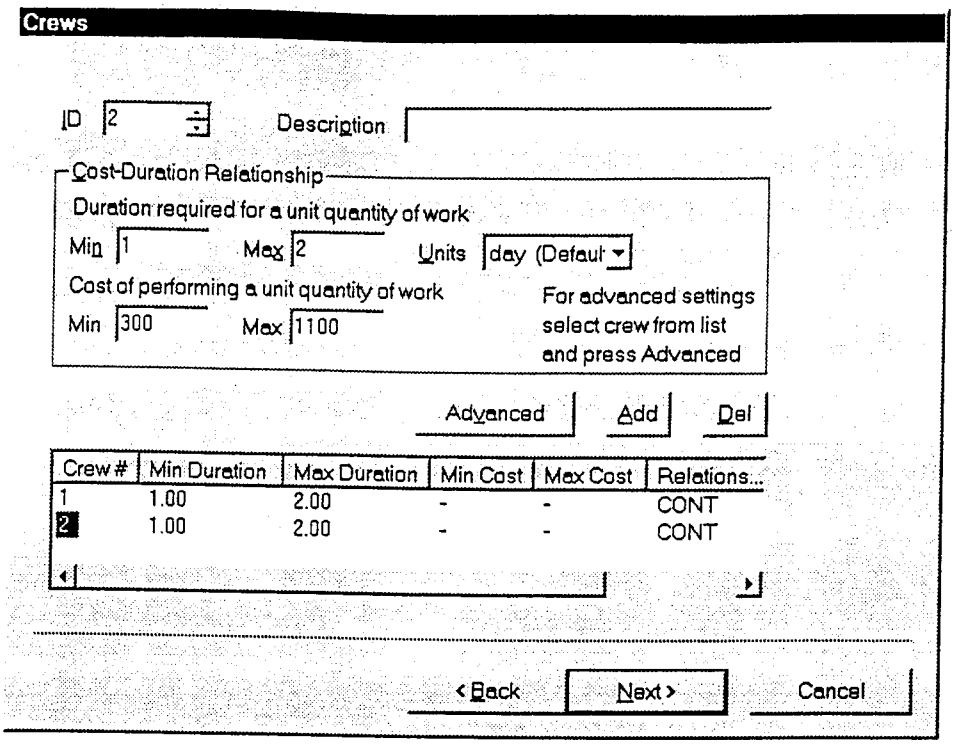

Crews
ID 2
Description
Cost-Duration Relationship
Duration required for a unit quantity of work
Min 1
Max 2
Units day (Defaul
Cost of performing a unit quantity of work
Min 300
Max 1100
For advanced settings select crew from list and press Advanced
Advanced
Add
Del
Crew # | Min Duration | Max Duration | Min Cost | Max Cost | Relations...
1 | 1.00 | 2.00 | - | - | CONT
2 | 1.00 | 2.00 | - | - | CONT
< Back
Next >
Cancel

Figure 6b

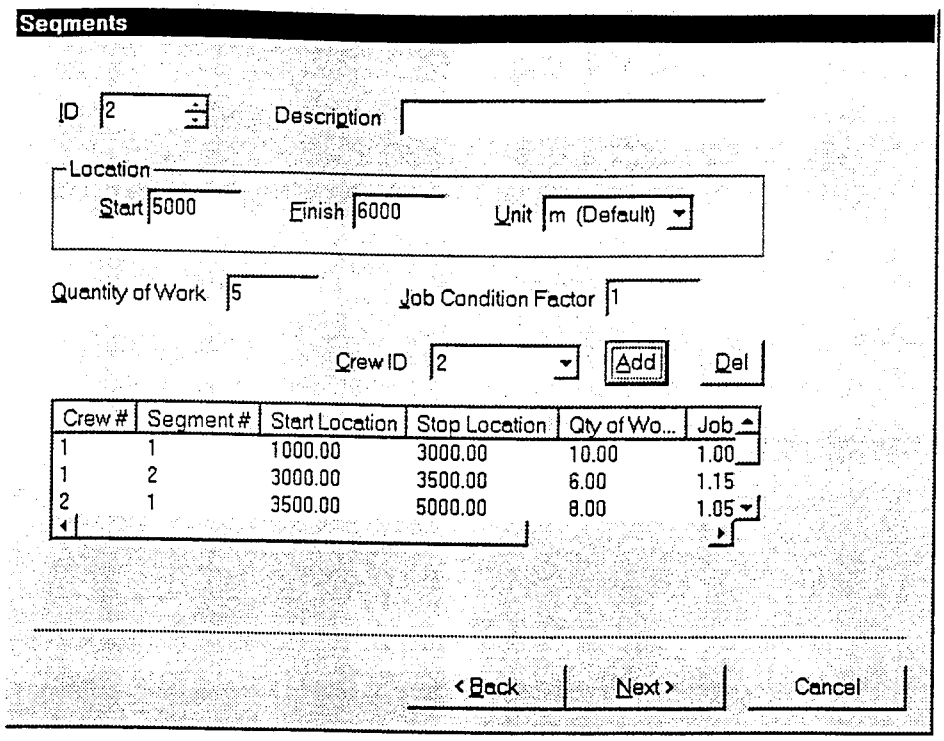

Segments
ID 2
Description
Location
Start 5000
Finish 6000
Unit m (Default)
Quantity of Work 5
Job Condition Factor 1
Crew ID 2
Add
Del
Crew # | Segment # | Start Location | Stop Location | Qty of Wo... | Job
1 | 1 | 1000.00 | 3000.00 | 10.00 | 1.00
1 | 2 | 3000.00 | 3500.00 | 6.00 | 1.15
2 | 1 | 3500.00 | 5000.00 | 8.00 | 1.05
< Back
Next >
Cancel

Figure 6c

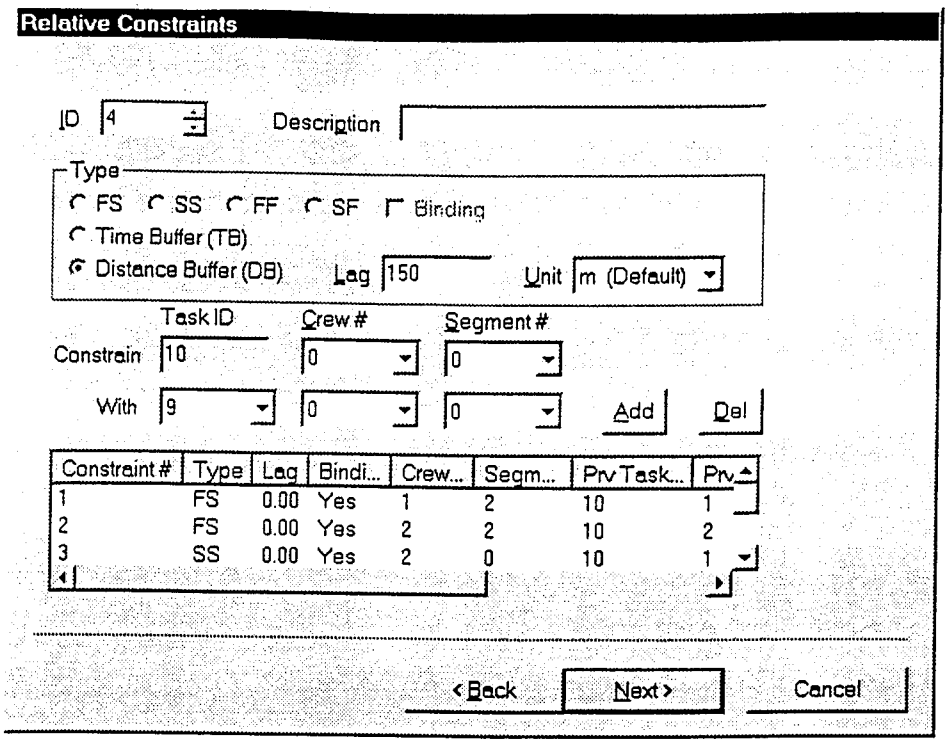

Relative Constraints
ID 4
Description
Type
FS SS FF SF Binding
Time Buffer (TB)
Distance Buffer (DB)
Lag 150
Unit m (Default)
Task ID
Crew #
Segment #
Constrain 10 0 0
With 9 0 0
Add
Del
Constraint # Type Lag Bindi... Crew... Segm... Prv Task... Prv
1 FS 0.00 Yes 1 2 10 1
2 FS 0.00 Yes 2 2 10 2
3 SS 0.00 Yes 2 0 10 1
< Back
Next >
Cancel

Figure 6d

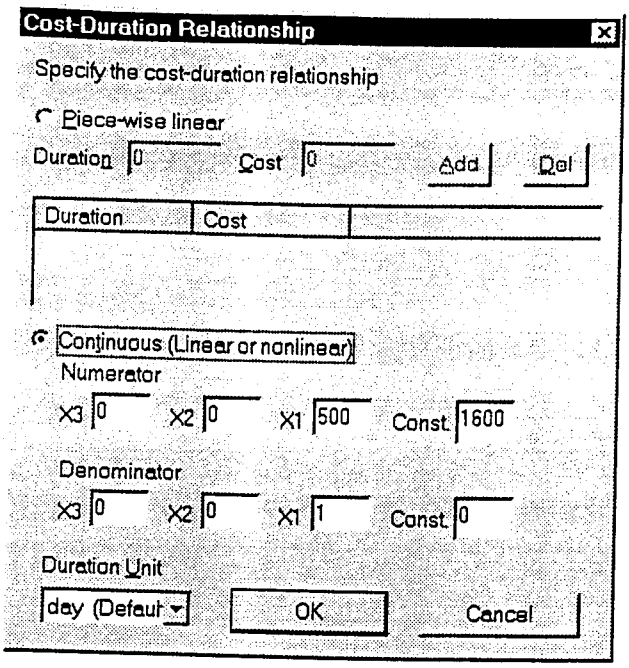

Cost-Duration Relationship
Specify the cost-duration relationship
Piece-wise linear
Duration 0
Cost 0
Add
Del
Duration
Cost
Continuous (Linear or nonlinear)
Numerator
X3 0
X2 0
X1 500
Const. 1600
Denominator
X3 0
X2 0
X1 1
Const. 0
Duration Unit
day (Defaul
OK
Cancel

Figure 7