INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

COMPUTERISED SCHEDULING AND CONTROL OF RESIDENTIAL HOUSING PROJECTS

Ramaneetharan Ramanathan

A Thesis

in

The Department of Building, Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University,

Montreal, Quebec, Canada

July, 2000

ABSTRACT

COMPUTERISED SCHEDULING AND CONTROL OF RESIDENTIAL HOUSING PROJECTS

Ramaneetharan Ramanathan

This research study presents a practical object-oriented model for scheduling and control of residential housing projects. The model is designed using an Object-Oriented modeling approach and incorporates 18 classes that are designed to facilitate the scheduling and control of residential housing projects. The model also includes two newly developed algorithms for scheduling the construction work of subcontractors in repetitive housing activities and for tracking and control of housing construction. The model considers a number of practical factors commonly encountered in scheduling and control process of these type of projects.

The scheduling algorithm complies with three major constraints namely, precedence relationships, availability period and crew work continuity and is applied in order to determine the start and finish date of the subcontractor in each housing unit. The tracking and control algorithm is designed to evaluate the cost and work performance of an on-going project at three levels: 1) entire project, 2) housing unit, and 3) subcontractor. The application of this algorithm facilitates the early detection of construction problems, if any, allowing timely corrective actions to be considered.

The developed model is implemented as a prototype software system named 'Residential Planner'. In addition to considering various practical aspects, Residential Planner can generate a number of specialized reports to address the diverse needs of the residential development firms. Residential Planner is an effective tool for scheduling and control of housing projects, and its application can lead to savings in project time and cost.

ACKNOWLEDGEMENTS

I take this opportunity to express my heartfelt appreciation and deepest gratitude to my supervisors Dr. O. Moselhi and Dr. K. El-Rayes for their perceptive guidance, helpful advice, critical insights, inspiration and encouragement throughout all stages of this research. Their suggestions to improve the contents of this thesis are greatly appreciated. I am indebted to the professional personnel from residential development firms who provided their experience and time for this research.

A special note of appreciation is extended to my parents, brothers and uncle for their encouragement, patience and psychological support. My sincere thanks to the faculty, staff, and colleagues at the Department of Building, Civil and Environmental Engineering who helped in various ways to carry out this research. A special thanks is extended to my colleagues: Dr. K. Suresh Kumar, Mohamed Marzouk, Sanjeev Kumar Sivakoti, Deep Goradia, Nikhil and Archana Vyas for their constructive criticism and helpful advice.

Last but not the least, the financial support for this research work provided by Dr. O. Moselhi's research grant is gratefully acknowledged.

TABLE OF CONTENTS

NOMENCLATURE

Act_Dur[c] :	Actual duration of finished activity c.
ACWP[r] :	Actual cost for work performed in $.
ADWP[r]	Actual duration for work performed up to report date r.
AWP[r] :	Actual work performed prior to report date r.
BCWP[r]	Budgeted cost for work performed up to report date r.
c :	Completed activity number.
C :	Total number of activities completed.
Cum_Dur :	Cumulative duration of all activities.
CV[r] :	Cost variance that represents the status of the project with respect to cost on report date r.
d :	Workday (from d=1 to PD)
Dur[j] :	Duration of housing unit j.
Early_Start[k] :	Early start date of housing unit k.
EC[j] :	Equipment cost of housing unit j in $.
Eff_Dur :	Time between the planned early start and day d for an activity in progress.
Exp_Dur :	Expected duration to finish an activity in one housing unit or all housing units.
Finish[k] :	Finish date of housing unit k.
i :	Non-repetitive activity number (i=1 to I).
I :	Total number of non-repetitive activities in a project.
Idle[k] :	Idle time of the construction crew in housing unit k.

Ind_Cost :	Project indirect cost per day in $.
j :	Housing unit number (j=1 to J).
J :	Total number of housing units in a project.
k :	Unit number that satisfies the user-specified execution order of housing units.
LC[j] :	Labor cost of housing unit j in $.
lump_price[m] :	Lump sum contract awarded to activity m in $.
m :	Repetitive activity number (m=1 to M).
M :	Total number of repetitive activities in the project.
MC[j] :	Material cost of housing unit j in $.
n :	Subcontractor number (n=1 to N).
N :	Total number of subcontractors involved in a project.
no_act :	Total number of activities in the project (i.e. both repetitive and non-repetitive).
Order[j] :	User-specified execution order of housing units.
p :	Progressing activity number (p=1 to P).
P :	Total number of activities in progress.
PD :	Planned duration of the project in days.
PDWP[r] :	Planned duration for work performed up to report date r.
Per_Comp[p] :	Percentage of work completed in progressing activity p.
Plan_Cost[d] :	Planned cumulative cost for day d.
Plan_Dur[m] :	Planned duration of repetitive activity m in days.
Plan_Work[d] :	Planned cumulative work for day d.

Prod[n] :	Required productivity / day for the subcontractor n.
Qty[j] :	Quantity of work in housing unit j.
r :	Report date.
Shift[j] :	Required shift to start and finish dates of housing unit j to comply with crew work continuity constraint.
sub_avail_start[n] :	User-specified early available date of subcontractor n at site.
SV[r] :	Schedule variance that represents the status of the project with respect to time.
Temp, Temp1:	Temporary variables used in tracking and control calculations.
Total_Cost :	Total direct cost of an activity.
Total_Qty[n] :	Total quantity of work to be performed by subcontractor n.
unit_price[m] :	Unit price contract awarded to activity m in $.

LIST OF FIGURES

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

CHAPTER 6

LIST OF TABLES

CHAPTER 3

CHAPTER 5

CHAPTER 6

CHAPTER 1

INTRODUCTION

1.1 Residential Housing Construction

The residential housing sector represents a significant percentage of the construction industry in North America. According to the latest available statistics (Statistics Canada, 1998), the total capital expenditure in residential building construction was $ 37.4 billions in the year 1997, which is almost 42% of the total capital expenditure on construction in Canada. This sector is composed of large number of small construction development firms primarily due to low capital investment requirements.

Challenges facing the residential development firms today are more complex than ever before. These challenges can be attributed to new constraints such as different life styles (aging population, increasing number of single families etc.), increased involvement of public and government agencies, environmental regulations and conditions, rising cost of housing and effects of technology. According to the latest publication by the Canadian Mortgage and Housing Corporation (CMHC, 1996), housing starts fell to 111,000 units in 1995, lowest since 1960 (see Figure 1.1). The same report indicates that there will be a decline in the household growth for the period 1991-2016. The above challenges combined with the decline in household growth would eventually lead to more competition among the large number of small-scale development firms.

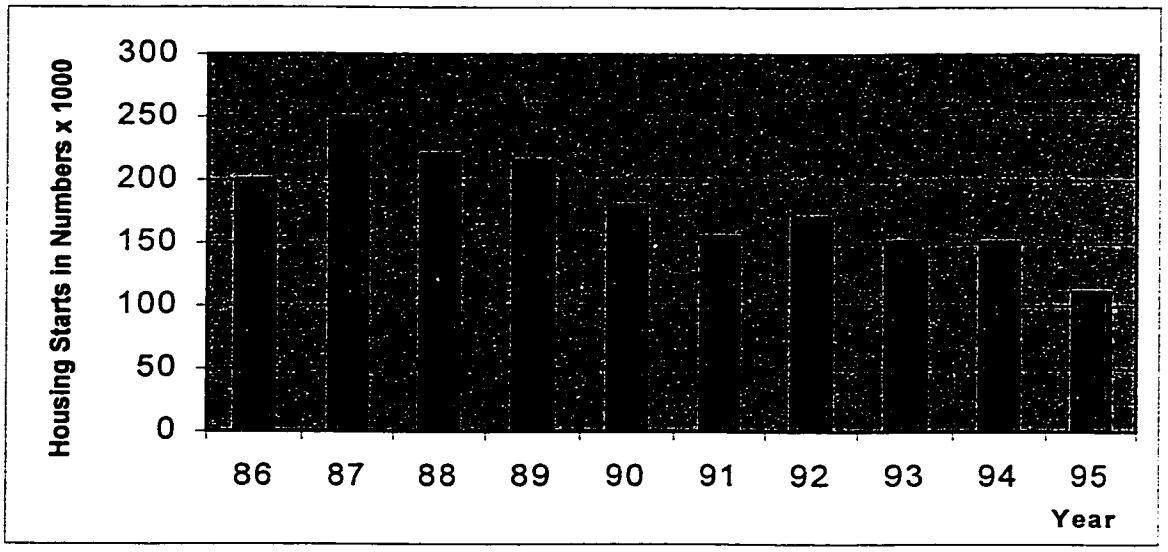


Figure 1.1 Housing Starts in Canada, 1986-1995

Hence, there is an imminent need to adopt new ideas and approaches for survival and sustenance of any residential development firm. One of the key areas that has a significant impact on the cost and duration of residential housing construction is project scheduling and control.

1.2 Challenges in Scheduling and Controlling Residential Housing Projects

The development of an effective plan for scheduling and control of residential construction involves a number of challenging tasks for residential development firms. The first challenging task in scheduling residential housing projects is that it involves both non-repetitive and repetitive activities. These two types of activities have to be scheduled using different methods to be effective. Non-repetitive activities can be scheduled using traditional network techniques. On the other hand, repetitive activities have to be scheduled by a technique that can provide resource-driven scheduling. This can be achieved by integrating

traditional network scheduling and resource-driven scheduling techniques in an effective environment.

The second challenging task is to develop a resource-driven scheduling algorithm that incorporates utilization of direct labor force and/or subcontractors. Most of the construction work in a typical residential housing project are performed by subcontractors (Russell, 1989). However, the developer often performs some of these activities by employing direct labor force. Hence the algorithm should consider this practical aspect in addition to a number of other factors commonly encountered in these projects.

The third challenging task is to develop a control methodology for residential projects. Tracking and control of these projects is challenging, primarily due to the large number of housing units and subcontractors involved in this type of construction. It will be easy to identify construction problems in a project, if the methodology can determine the time and cost performance of an individual subcontractor or a particular housing unit in addition to the whole project. This will enable early and accurate identification of troubles, if any, thereby timely actions can be initiated

The fourth challenging task is to generate a number of specialized reports to address the diverse needs of the various parties involved in the construction of these projects. These reports should include both graphical and text reports in

calendar and workdays. Due to the large number of subcontractors and housing units involved in these projects, it becomes necessary to produce a separate schedule for each subcontractor and housing unit, in addition to the project master schedule. The project master schedule should provide the necessary details with which the developer can organize, control and carry out the project. The subcontractor schedule should give their specific activities and corresponding dates to perform these activities in different houses. The housing unit schedule should assist the developer in addressing the concerns and questions from the buyer of a particular house. It will be advantageous for easy and early detection of troubles, if the tracking and control results are presented in a graphical format.

1.3 Research Objectives

The main objective of this research is to develop a model for scheduling and control of residential housing projects that addresses the challenges outlined in the earlier section. In order to develop this model, the objectives of this study are:

1) To build and extend on a recently developed model for scheduling of repetitive construction projects (El-Rayes, 1997) in order to consider the special characteristics and unique features of scheduling and control of residential housing projects.
2) To develop a resource-driven scheduling algorithm for subcontractors involved in residential housing projects.

3) To formulate an algorithm for effective tracking and control of an on-going residential housing project with respect to cost and time.
4) To develop effective methods to generate flexible and specialized reports in text and graphical format to address the needs of the developer and subcontractors.
5) To implement the suggested algorithms and methodologies in a user-friendly prototype software system.
6) To provide a database support for efficient use of the developed computer software program and store valuable historical records.

1.4 Thesis Organization

Chapter 2 presents a literature review of available scheduling techniques for repetitive activities and residential housing projects. A brief introduction to an Object-Oriented model for scheduling of repetitive construction (El-Rayes, 1997) is also presented.

Chapter 3 explains the analysis and design stages of the proposed Object-Oriented model for scheduling and tracking and control of residential housing projects. The analysis stage presents the findings of the meetings held with representatives of residential development firms and the developed object model. The design stage presents the architecture of the proposed model and describes the developed classes.

Chapter 4 presents the proposed scheduling algorithm for subcontractors, and tracking and control algorithm for residential housing projects.

Chapter 5 presents the implementation stage of the proposed model as a prototype software system, its main components, and input and output.

Chapter 6 validates the results produced by the developed model and outlines its practical application aspects using two application examples.

Chapter 7 summarizes the results of this research, its contributions and recommendations for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Project planning and scheduling is essential in the management of construction projects in order to achieve the main objectives of completing the project within a fixed time, at a previously estimated cost and to specified standards of quality. This chapter presents a review of recent literature in traditional scheduling techniques for construction projects in general, and special scheduling techniques for repetitive construction projects in particular. In addition, the chapter provides an overview of available models for scheduling residential housing projects, and presents a review of an Object-Oriented model for scheduling repetitive construction.

2.2 Traditional Scheduling Techniques

2.2.1 Bar Chart Method

Bar Chart method is a graphical representation of the project schedule that was originated and developed by Henry L. Gantt during world war-I. The length of a bar represents the duration of each activity in accordance with the time scale of the chart. The major deficiency of this method is its inability to show the inter-relationships between activities, thereby failing to identify the critical activities, which actually control the project duration (Chzanowski and Johnston, 1986; Stradal and Cacha, 1982). In spite of this drawback, these bar graph charts are

widely used in the construction industry primarily due to its graphic and easily understandable format (Nunnally, 1998).

2.2.2 Network Techniques

Network diagram is a graphical representation of activities and their relationships. By displaying the relationships between activities, these diagrams effectively eliminate the drawbacks of bar charts. Network techniques enable the identification of critical activities that control the project duration. There are two common types of network techniques widely used to represent activities and their inter-relationships. The first type is known as arrow diagram method (ADM) in which activities are represented by arrows and nodes connecting these arrows are considered events or milestones. The second type is activity on node diagram and is commonly known as precedence diagram method (PDM). In PDM, nodes represent activities and connecting arrows represent the inter-relationship among these activities. ADM can consider only one type of relationship namely finish to start where as PDM can consider four different types of relationships namely, Finish to Start, Finish to Finish, Start to Start and Start to Finish (Nunnally, 1998).

Traditional scheduling techniques focus on project duration and give little consideration for effective use of resources. However, contractors and site superintendents are more concerned about effective use of resources rather than critical paths or early project completion (Birrell, 1980; Kavanagh, 1985). The

main concern is that these techniques do not consider effective resource utilization and for this reason are widely criticized in literature (Kavanagh 1985; Birrell, 1980; Davies 1974). Moreover, these techniques produce large and complex schedules when applied to repetitive activities and the complexity increases with the increase in repetitions (Carr and Meyer, 1974). Hence it becomes practically inapplicable for projects that comprise a large number of repetitive activities such as a housing development project with 100 houses. Another important limitation of traditional techniques is its inability to maintain crew work continuity. Crew work continuity is the process by which a crew working on a particular repetitive activity moves from one unit to another, without any delay. This smooth movement should be planned in order to maximize resource utilization and minimize idle time (Ashley, 1980; Kavanagh, 1985; El-Rayes and Moselhi, 1997).

2.3 Techniques for Scheduling Repetitive Activities

Due to the limitations of the network techniques mentioned in the earlier section, a number of techniques were proposed in the literature for scheduling projects with repetitive activities. These methods can be grouped into two main categories. The first category comprises of methods, which were developed to schedule typical repetitive activities only. These methods are often referred to as 'Line Of Balance' (LOB) (Lumsden, 1968; Al Sarraj, 1990; Carr and Meyer, 1974). The second category includes methods which were developed to schedule both typical and non-typical repetitive activities, and are often referred

to as 'Linear Scheduling Method' (LSM) (Russell and Caselton, 1988; Chrazanwski and Johnston, 1986).

There are other techniques proposed in the literature for scheduling repetitive activities utilizing the principles of either LOB or LSM, with the main objective of maintaining crew work continuity. These techniques include 'Construction planning technique' (Peer, 1974; Selinger 1980), 'Vertical production method' (O' Brien, 1975; Barrie and Paulson, 1992), 'Time-Location matrix model' (Stradal and Cacha, 1982), 'Disturbance Scheduling' (Whiteman and Irwing, 1988), 'Horizontal and Vertical Logic Scheduling', (Thabet and Beliveau, 1994), 'Linear Balance Charts' (Barrie and Paulson, 1992), 'Velocity Diagrams' (Dressler 1980).

2.3.1 Line of Balance (LOB)

Line of Balance (LOB) method was developed by the U.S Navy in 1942 for planning and control of repetitive projects (Lumsden, 1968). The method was primarily designed and used in the industrial manufacturing applications. It was employed by the industrial engineers in manufacturing to optimize output by determining the required resources and speed for each stage. In industrial manufacturing, products move along a production line unlike in construction where the products are stationary and operatives move along a line. Due to this difference, LOB method was modified in 1966 from its original manufacturing industry purpose to enable its application to housing projects and was published in a report by the National Building Agency (Programming, 1966). The developed

method was simple and the schedule can be represented by plotting the number of units in Y-axis against the duration in X-axis. Repetitive activities are represented by a separate inclined bar. There are several methods proposed in the literature having the same name 'Line of Balance' (LOB) and sharing the same concept (Lumsden, 1968; Al Sarraj, 1990; Arditi and Albulak, 1986). Al Sarraj (1990) developed a formal algorithm for LOB to facilitate scheduling, resource management and project analysis and control in order to provide a mathematical alternative for the graphical LOB method. Arditi and Albulak (1986) used network technique and LOB to schedule a highway project in an attempt to compare the two methods. They concluded that LOB schedule is easy to understand and requires less time and effort.

Even though LOB method has been found to have apparent advantages such as 1) maintaining crew work continuity, 2) generating resource driven schedules, 3) providing clear and easy to produce schedules (Arditi and Albulak, 1986; Hegazy et al, 1993), it has been criticized in the literature for a number of reasons. Kavanagh (1985) indicated that LOB method was designed to model simple repetitive production process and does not readily transplant into complex construction projects. Neale and Raju (1988) attempted to refine LOB method in a spreadsheet format by introducing activities that run concurrently. They confronted complex relationships and concluded that it was practically infeasible to draw the schedule in the form of a diagram. Arditi and Albulak (1986) commented about the visual problems associated with the graphical LOB

diagram and suggested that different colors can be used to distinguish overlapping activities. Neale and Neale (1989) mentioned that while using LOB technique to monitor progress, it could show only a limited amount of information and a limited degree of complexity. Another major disadvantage of LOB method is its inability to schedule non-typical repetitive activities (Moselhi and El-Rayes, 1993).

2.3.2 Linear Scheduling Method (LSM)

Linear Scheduling Method (LSM) was developed to overcome the limitations of LOB method. LSM has all the apparent advantages of LOB method and is capable of scheduling typical and non-typical repetitive activities (Russell and Caselton, 1988; Chrzanowski and Johnston, 1986; Moselhi and El-Rayes, 1993). An important difference between LOB method and LSM is the graphical presentation of the schedule. In LOB method, an activity is represented by two parallel lines, whereas in LSM it is represented by a single line as shown in Figure 2.1.

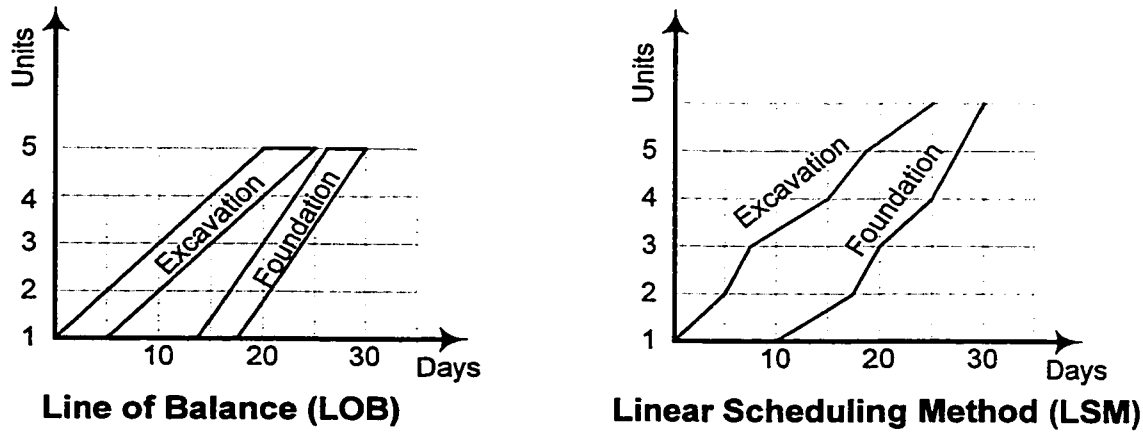


Figure 2.1 Graphical Representation of LOB and LSM

Johnston (1981) described the basic presentation format of LSM as having two axes. The horizontal axis represents the project duration and the vertical axis represents the number of repetitive units while separate diagonal lines represent repetitive activities. He suggested that LSM schedule is simple and can convey detail work schedule. Chrzanowski and Johnston (1986) employed CPM technique and LSM to schedule a highway project in an attempt to evaluate the capabilities of LSM. They concluded that LSM has several advantages such as: 1) simple and easy to understand graphical schedule and 2) availability of fairly detail information without being confronted with the numerical data found in network techniques.

2.4 Scheduling Models for Residential Housing Projects

In 1968, Lumsden published a report explaining the various aspects of LOB method and how it can be applied for scheduling and tracking and control of residential projects. The method was based on the fact that all houses have a natural rhythm at which they should be built and any deviation from this rhythm results in crew idle time. Natural rhythm of an activity is the time taken for a crew to finish the activity in one house and move to the next house. For example, if a construction activity in a single house takes 4 hours and the workweek includes 40 hours, then the natural rhythm of this activity is 10 houses per week.

The report suggests that the application of LOB method leads to an increase in construction productivity, and accordingly a reduction in duration and total cost.

The advantages of applying this method to residential projects were highlighted using two projects that included 187 and 88 low-rise houses, respectively. The application of LOB to these projects was found to reduce, in some instances, the time taken to build a house by one half and the labor requirement by one-third. LOB method, however, can be used to schedule typical repetitive activities only, and cannot consider non-typical repetitive activities, which are commonly found in residential housing projects. As such, the LOB method cannot be effectively applied in developing practical schedules for real-life housing projects.

Masoud and Diekmann (1997) proposed a simulation model to determine the optimum number of houses in which work can start simultaneously. The model determines the optimum number of housing starts in order to minimize the project cost and duration, considering the effect of learning curve. The model was developed using SLAM-II and consists of three major components: 1) model network, 2) model control statement and 3) user-inserted code. Model network represents the construction network for one housing unit. Model control statement identifies the resource availability constraints and user-inserted code considers different user specified learning rates. Similar to the LOB method, the basic assumption of this model is that all activities are typical repetitive activities. As such, the application of this model is also limited due to its inability to consider non-typical repetitive activities.

2.5 Object-Oriented Modeling for Repetitive Construction

The present scheduling and control model for residential housing projects extends and builds on a recently developed Object-Oriented Model for scheduling of repetitive construction projects named LSCHEDULER (El-Rayes, 1997). This section provides an overview of LSCHEDULER and highlights its main features and components. LSCHEDULER is capable of combining algorithms of network scheduling techniques and resource-driven scheduling techniques to schedule non-repetitive and repetitive activities, respectively. The developed algorithm for scheduling repetitive activities satisfies three main constraints: 1) logical precedence relationships, 2) crew availability and 3) crew work continuity, and considers the impact of a number of practical factors namely, 1) typical and non-typical activities, 2) crew availability period on site, 3) utilization of single and multiple crews, 4) activity interruption, 5) sequence of construction operations, 6) impact of weather and 7) effect of learning curve (El-Rayes and Moselhi, 1998). The scheduling is performed in two stages: the first considers the precedence relationships and crew availability constraints, and the second complies with crew work continuity constraint.

In order to enable the integration of repetitive and non-repetitive scheduling techniques, LSCHEDULER incorporates two types of activity objects: repetitive and non-repetitive. In order to consider precedence relationships among repetitive and non-repetitive activities, three different types of relationships were proposed in LSCHEDULER. The first type is Regular relation where the

predecessor activity as well as the successor activity is non-repetitive. The second type is the relation between two repetitive activities and is called Repetitive relation. The third type is Hetero relation, which represents either the relationship between two different types of activities (i.e. repetitive and non-repetitive), or the relationship between two specific units of two distinct repetitive activities. The model was developed using Object-Oriented Modeling (OOM) concepts which utilizes classes and relationships to solve complex problems (Martin, 1993). The model consists of 10 classes namely, *Project, Project-Data, Date, Activity, Regular-Relation, Repetitive Activity, Non-Repetitive Activity, Repetitive-Relation, Hetero-Relation* and *Crew-Formation.* The object model representing the model classes is shown in Figure 2.2 (El-Rayes, 1997).

Project class was designed to reflect the special characteristics of repetitive construction projects and to perform a number of needed functions such as 1) adding new data to the project, 2) sorting of activities and relationships and 3) initiating schedule calculations. *Project-Data* class was designed as the most generic class and includes only one string data member called name, which is used as an identifier. *Date* class was designed to 1) convert a workday schedule to a calendar date schedule, 2) determine the weekday of a calendar date, 3) add and subtract duration to and from a calendar date and 4) calculate the impact of weather for a specific period of time. *Activity* class was designed to represent the general features of construction activities (repetitive and non-repetitive). *Non-Repetitive Activity* and *Repetitive Activity* classes were designed

to represent the special characteristics of non-repetitive and repetitive construction activities, respectively.

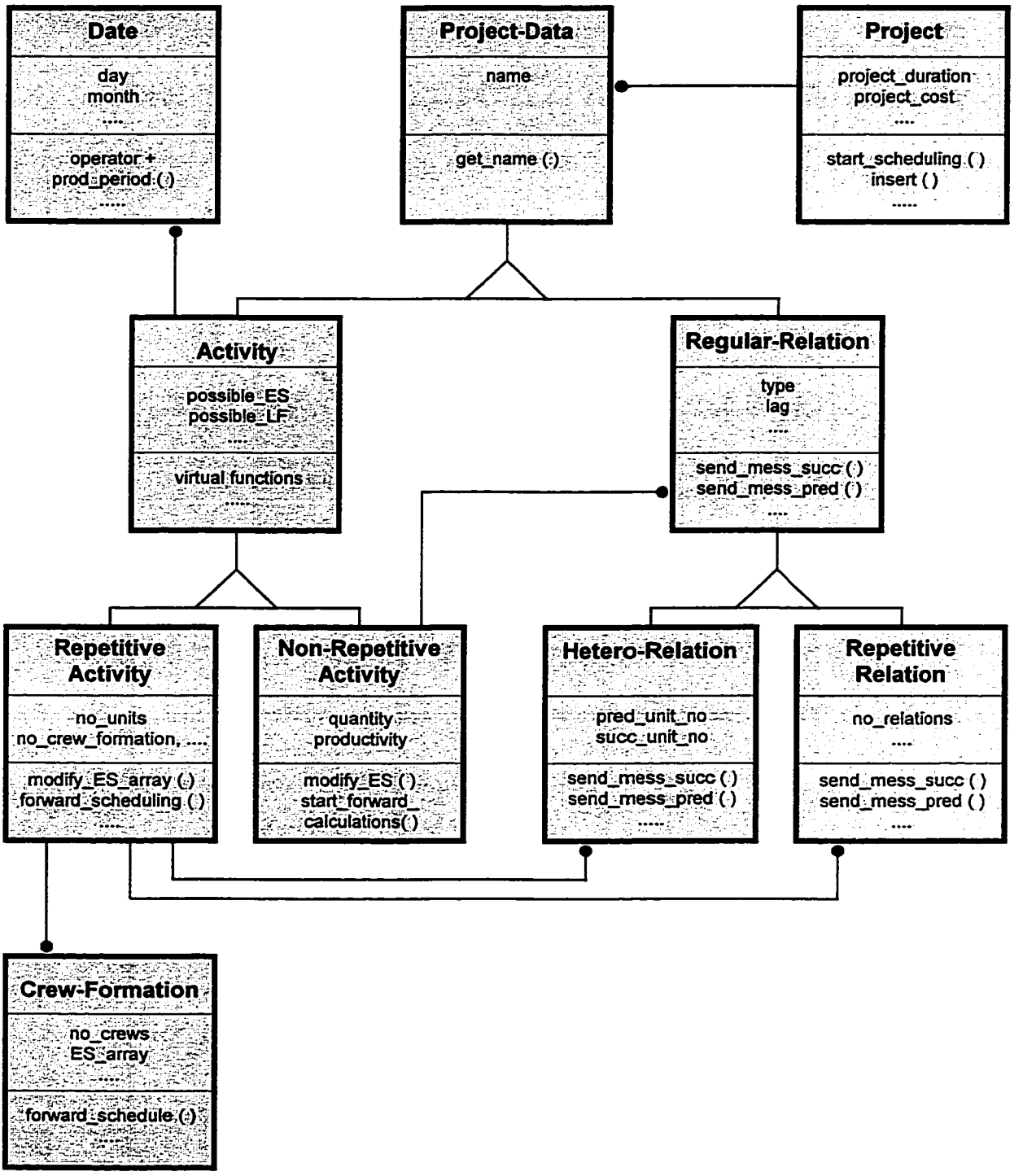


Figure 2.2 Object Model Proposed by El-Rayes (1997).

Regular-Relation class was designed to consider the general aspects of precedence relationships and also to represent the relationship among two non-repetitive activities. It is capable of considering four types of relationships namely, Finish to Start (FS), Finish to Finish (FF), Start to Start (SS) and Start to Finish (SF). *Repetitive-Relation* class was designed to represent the precedence relationship between two repetitive activities. *Hetero-Relation* class was designed to represent either the precedence relationship between two specific units of two different repetitive activities or the precedence relationship between a non-repetitive and a repetitive activity. *Crew-formation* class was designed to consider crew utilization data such as number of available crews, daily output and cost while performing scheduling calculations.

The above mentioned Object-Oriented model was developed as a generic model that can be applied to schedule all types of repetitive construction projects such as: housing projects, high-rise buildings, highway construction, pipeline networks and segmental bridge construction. Due to its generic nature, the model does not consider the special characteristics of residential housing projects, and its unique requirements. Specifically, it does not consider: 1) the impact of utilizing subcontractors on scheduling of residential projects, 2) tracking and control of this class of projects, and 3) the specific requirements of residential development firms and subcontractors in generating effective and practical representation of the developed schedule.

2.6 Summary

A review of recent literature has been presented in this chapter which includes traditional scheduling techniques, techniques for scheduling repetitive construction projects and scheduling models for residential housing projects. An overview of an Object-Oriented model for scheduling repetitive construction has also been presented. The findings of this review have been effectively used in the development of the proposed model for scheduling and tracking and control of residential housing projects, which is described in the following Chapters.

CHAPTER 3

PROPOSED MODEL

3.1 Introduction

This chapter presents the development of the proposed Object-Oriented model for scheduling and tracking and control of residential housing projects. The model is designed to comply with a number of practical factors commonly encountered in real-life residential housing projects. In order to identify these practical factors, a series of meetings were conducted with representatives of residential development firms. The findings of both these meetings and a comprehensive literature review are carefully considered in the development of the present model. The model is developed using Object-Oriented Modeling concept, which consist of three main stages: analysis, design and implementation (Rumbaugh et al, 1991). This chapter presents the analysis and design stages of the proposed model.

3.2 Object-Oriented Modeling

Object-Oriented Modeling attempts to satisfy the needs of the end user via real-world modeling capabilities (Khoshafian and Abnous, 1995). Through Object-Oriented concepts, complex problems are modeled as a set of simple objects. These objects should reflect the real-world parameters such as a concept, abstraction or any related matter, with clear boundaries and meanings for the problem at hand (Rumbaugh et al, 1991).

Object-Oriented Modeling utilizes several major concepts such as encapsulation, abstract data typing, inheritance and object identity (Khoshafian and Abnous, 1995). Using encapsulation, an object combines data, operations and functions together and hides them from other objects in order to ensure their integrity (Martin.J, 1993). A group of objects with similar properties (attributes), common behavior (operations) and relations are grouped to form a class (Rumbaugh et al, 1991). In abstract data typing, the essential characteristics of an object are identified and implemented using classes and is hidden from other objects. Inheritance allows new classes to expand and build on the basis of an existing parent or super-class. Each super-class can have several children or sub-classes. Inheritance behavior enables sub-classes with identical functions, to be inherited from a super-class, thereby avoiding duplication of functions. It also provides a natural mechanism for organizing information according to the real-world problem at hand. Object identity is the property of an object, which distinguishes each object and can contain or refer to other objects.

There are several notations available for graphical representation of the components of an Object-Oriented model in order to facilitate its development (Booch, 1994, Rambaugh et al, 1991 and Martin, 1993). A notation similar to the one proposed by Rambaugh et al (1991) is used in this thesis. The development of an Object-Oriented model, in general, consists of three main stages: analysis, design and implementation (Rumbaugh et al, 1991).

3.3 Analysis Stage

Analysis is the first stage in developing an Object-Oriented model. In this stage a real-world problem is analyzed, in order to capture and model the problem and identify its objects, functions and relationships to other objects using abstraction, inheritance and object identity (Khoshafian and Abnous, 1995). The analysis stage produces an object model which represents the static nature of the model and a dynamic model which outlines the sequence of operations performed at different stages (Rumbaugh et al, 1991). The first step in the analysis stage is to clearly understand the nature of the problem at hand. In order to understand the nature and special characteristics of scheduling and tracking and control of residential housing projects, a comprehensive literature review was performed. In addition, a series of meetings were conducted with representatives of various residential development firms to understand the nature of residential housing projects and to develop means and methods by which their needs and requirements can be addressed.

3.3.1 Findings of the Meetings

A number of meetings were conducted with representatives of residential development firms in Montreal and Toronto (see Appendix I). In these meetings the representatives have indicated that:

1) Most development firms use some form of network techniques to schedule housing projects. However, such techniques are considered ineffective and impractical due to their inability to account for the repetitive nature of these projects.

2) Most of the construction work in housing projects is performed by subcontractors, who are more concerned about work continuity for their crews than critical paths.

3) The data available to schedule subcontractors are limited primarily due to the practical nature of awarding construction contracts. In most cases these contracts are awarded with an expected duration to finish an activity in one housing unit or the whole project along with a unit price or a lump sum contract. Hence, the schedule for subcontractors has to be developed using the minimum available information.

4) The current techniques used in tracking and control of housing projects, and determining their status as a whole is considered insufficient for an early identification of potential construction problems. This is mainly attributed to the large number of subcontractors and/or housing units. In order to facilitate the determination of such problems, if any, it was suggested that the status of an individual subcontractor or a particular housing unit be determined at any given time in addition to the project status.

5) The efficiency of scheduling, tracking and control of residential housing projects can be improved by producing effective and useful reports. These reports should include a number of graphical and text reports in both workdays and calendar days. Due to the large number of subcontractors and housing units involved in a typical residential project, the representatives have indicated the need to produce a separate schedule for each subcontractor and for each housing unit in addition to the project master schedule. They

also indicated the importance of presenting the tracking and control reports in an efficient graphical format to facilitate the identification of problems, if any.

The above findings along with those of the literature review are carefully considered in the development of an object model during the analysis stage, as described in the following section.

3.3.2 Object Model

An object model represents the static structure of the model. It presents the objects, their data, functions and relationships to other objects. The purpose of an object model is to describe the objects and to capture and reflect a real-world problem. For example, a model for scheduling and tracking and control of residential housing projects should identify and incorporate objects that are essential to the scheduling and control process such as number of housing units, activities and precedence relationships. The object model is represented graphically by object diagrams containing object classes, their hierarchies and relations to other objects.

Through inheritance, classes in an object model can be arranged in a hierarchy and can be inherited by other classes. Data and functions common to more than one class are grouped together to form a super-class. These common data and functions are then inherited by sub-classes. In addition to the inherited data and functions, these sub-classes includes additional data and/or functions making

them more specialized. Each level of hierarchy represents certain level of specialization. A higher level of the hierarchy represents more generic super classes, while each lower level represents more specialized sub classes. For example, both repetitive and non-repetitive activities have some common attributes such as name and relation. These common attributes can be grouped together to form a super-class called *Activity*. This super-class can then be inherited by two of its sub-classes namely, *Repetitive* and *Non-Repetitive*, thereby avoiding redefinition of common attributes. The concept of inheritance is effectively utilized in developing the present object model for scheduling and tracking and control of residential housing projects.

In order to consider the different types of activities and relationships included in a housing project and identify the needed classes in the present object model, a similar approach to that proposed by El-Rayes (1997) is utilized as shown in Figure 3.1. In this approach two types of activity classes are developed (*Repetitive* and *Non-repetitive*), and three types of precedence relationships are designed (Regular relation, Repetitive relation and Hetero relation). Regular relation (Relation-a) is used to define a relation between two non-repetitive activities. Repetitive relation (Relation-b) is developed to describe a relation between two repetitive activities. Hetero relation can be used to represent a relation between two specific units of two different repetitive activities (Relation-c) or a relation between a non-repetitive activity and a repetitive activity (Relation-d).

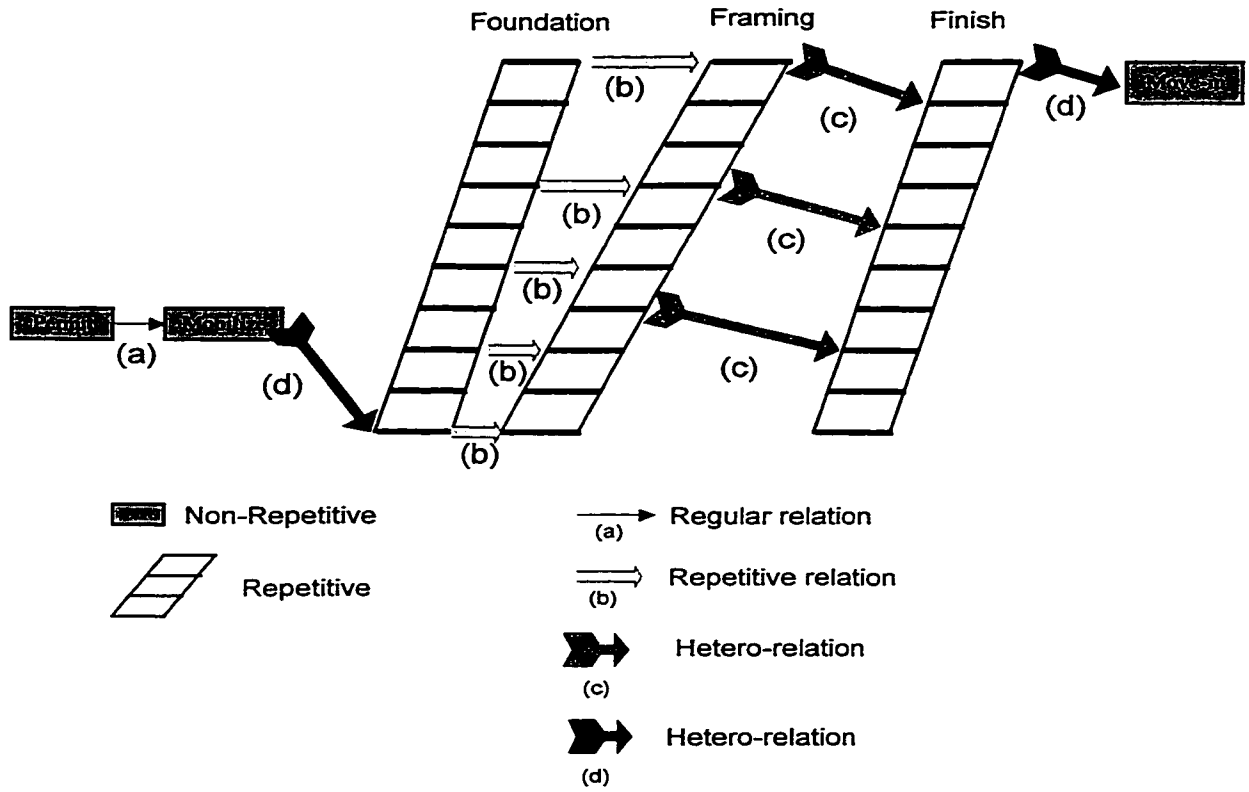


Figure 3.1 Activities and Relationships of Housing Construction

The above classes are incorporated in the present Object model for scheduling and tracking and control of residential housing construction. The present object model makes use of inheritance and object hiding in order to design a hierarchy of classes that includes 18 classes as shown in Figure 3.2. In this model, 10 classes (*Project-Data, Project, Date, Activity, Regular-Relation, Repetitive, Non-Repetitive, Hetero-Relation, Repetitive-Relation and Crew-Formation*) are similar to those proposed by El-Rayes (1997) and have been modified to suit the scheduling process for residential development projects. The eight other classes (*Data-Base, Tracking-Control, Project-Control, Unit-Control, Sub-Control, Own-Force, Sub-Contractor and View*) are newly developed and incorporated in the model as shown in Figure 3.2.

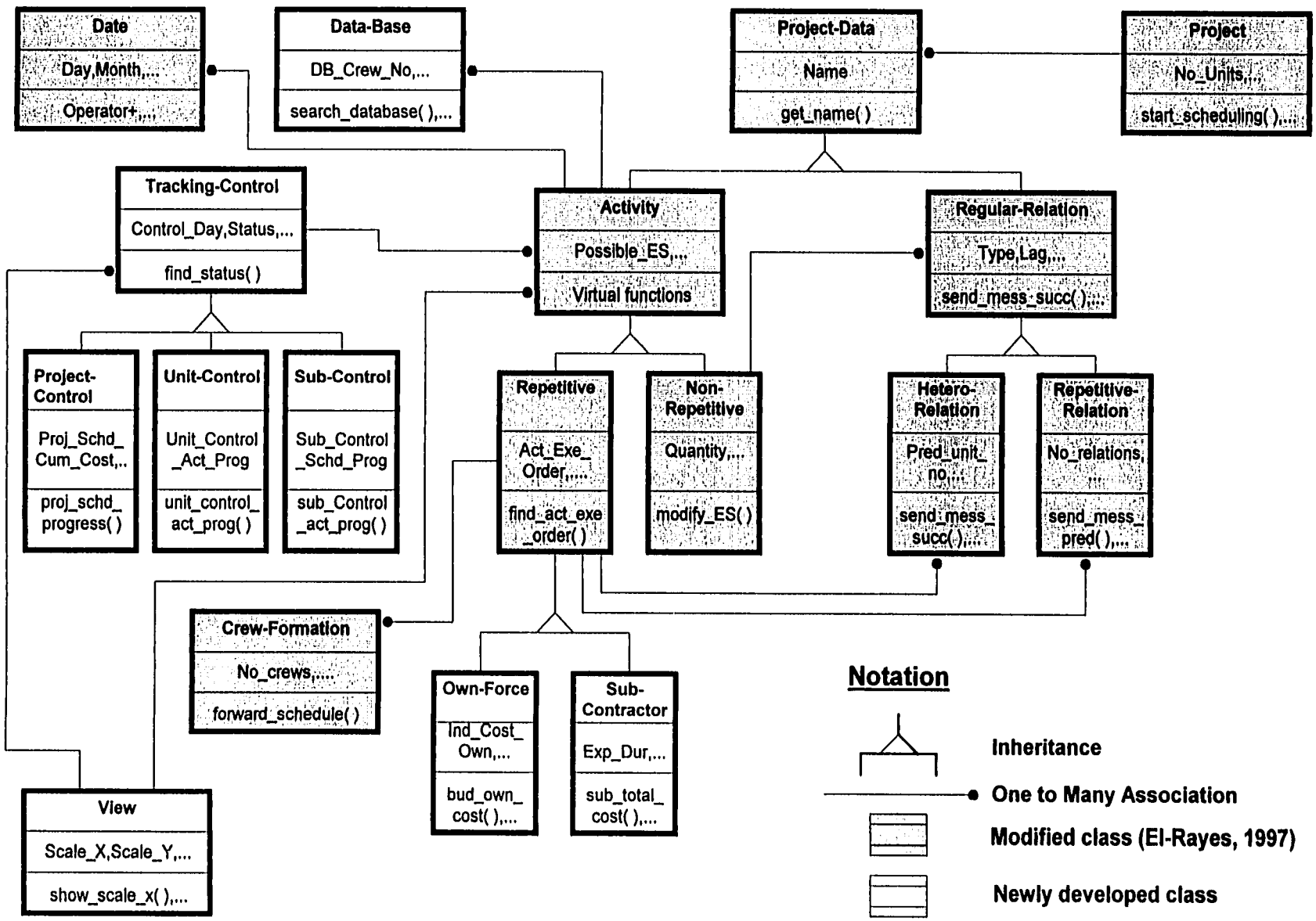


Figure 3.2 Proposed Object Model

The top most generic class, *Project-Data* contains general data such as name, number of units, measurement system. These general data are inherited by all its sub-classes in the hierarchy. The second level of the hierarchy consists of *Activity* and *Regular-Relation* classes, which are inherited from the super-class *Project-Data*. The third level of the hierarchy consists of *Repetitive* and *Non-Repetitive* classes derived from super-class *Activity*, and *Repetitive-Relation* and *Hetero-Relation* classes derived from super-class *Regular-Relation*. At the fourth level of the hierarchy, *Own-Force* and *Sub-Contractor* are derived from the super-class *Repetitive*. At the second level of hierarchy, there is an independent super-class, *Tracking-Control*, from which sub-classes *Project-Control, Unit-Control* and *Sub-Control* are derived.

In addition to the above-described hierarchy of classes, five other classes are incorporated in the present object model namely, *Project, Date, Data-Base, Crew-Formation* and *View*. *Project* class is designed to perform necessary functions at the project level such as adding a new activity or relationship and initiating scheduling calculations. *Date* class is designed to develop calendar date schedules and to convert workdays into calendar dates. *Data-Base* class is designed to support the present model with a database where historical data can be stored and retrieved for future use. *Crew-Formation* class is designed to consider various crew utilization options and *View* class is designed to generate graphical and text reports.

The *Project* class is designed to represent the characteristics of construction project as a whole. As such, it is designed to store and process various project data such as project activities, relationships and subcontractors, and therefore, it has a one to many associations with *Project-Data* class (see Figure 3.2). Similarly, the *Activity* class has one to many associations with *Date* and *Data-Base* classes from which it obtains the calendar dates and crew related information, respectively. *Tracking-Control* class has a one to many associations with *Activity* class, thereby, enabling it to obtain information such as activity name and start and finish dates. *Repetitive* class have one to many associations with *Crew-Formation, Repetitive-Relation* and *Hetero-Relation* classes, all of which will assist in scheduling of repetitive activities. *Non-Repetitive* class has one to many associations with *Regular-Relation* class in order to consider relationships. *View* class has one to many associations with *Activity* and *Tracking-Control* classes, which allows it to extract necessary information in generating effective reports.

3.3.3 Dynamic Model

As mentioned earlier, an object model describes its objects and their relationships to each other at a single moment of time. However, attributes of these objects are bound to undergo changes over a period of time (Rumbaugh et al, 1991). These changes that take place over a period of time and the sequence of operations that are performed during this period are described through a dynamic model. The two major concepts of a dynamic model are events and

states. An event is a message sent from one object to another and is something that happens at a point in time. A state of an object represents the values of its attributes at one point in time. An object can send an event to another object, which might change the state of the receiving object, or send an event to the original sender object and/or to a third object.

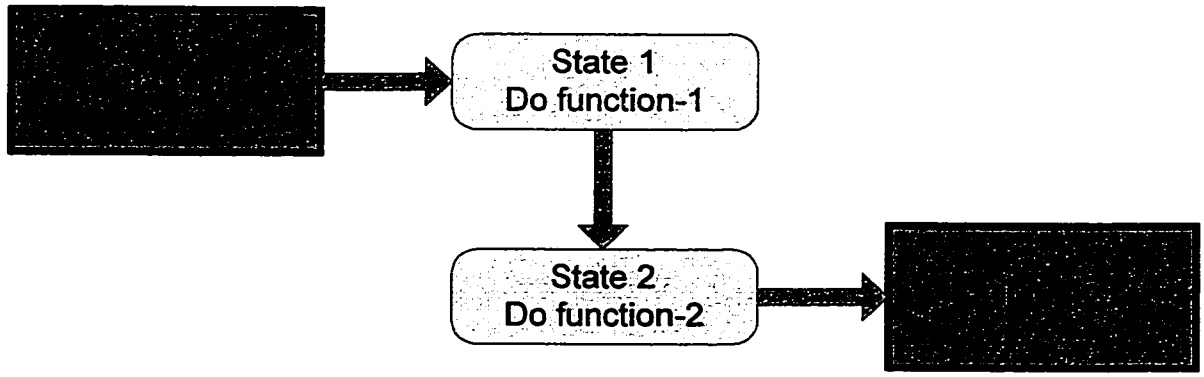


Figure 3.3 State Diagram

These events and changes in an object can be abstracted and represented by a state diagram. A state diagram is a network of events and states, which represent the important dynamic behavior of each object. These diagrams describe the various states of an object, events that trigger transition from one state to the other, and the various actions that are taken due to the state transitions. A state diagram can be graphically represented as nodes and arcs as shown in Figure 3.3. Nodes represent the states and the directed arcs are events or messages from one object to the other. A state is drawn as a rounded box and an event is drawn as an arrow from the receiving state to the target state. The dynamic model formulated for the proposed scheduling and tracking and control model includes a number of state diagrams. These state diagrams are explained in the following section outlining the design stage of the present model.

3.4 Design Stage

Design is the second stage in developing an Object-Oriented model and it builds on the analysis stage. The architecture of the proposed scheduling and tracking and control model for residential housing projects is designed as shown in Figure 3.4. The model comprises of five major components namely Input module, Scheduling module, Database Module, Control Module and Reports Module.

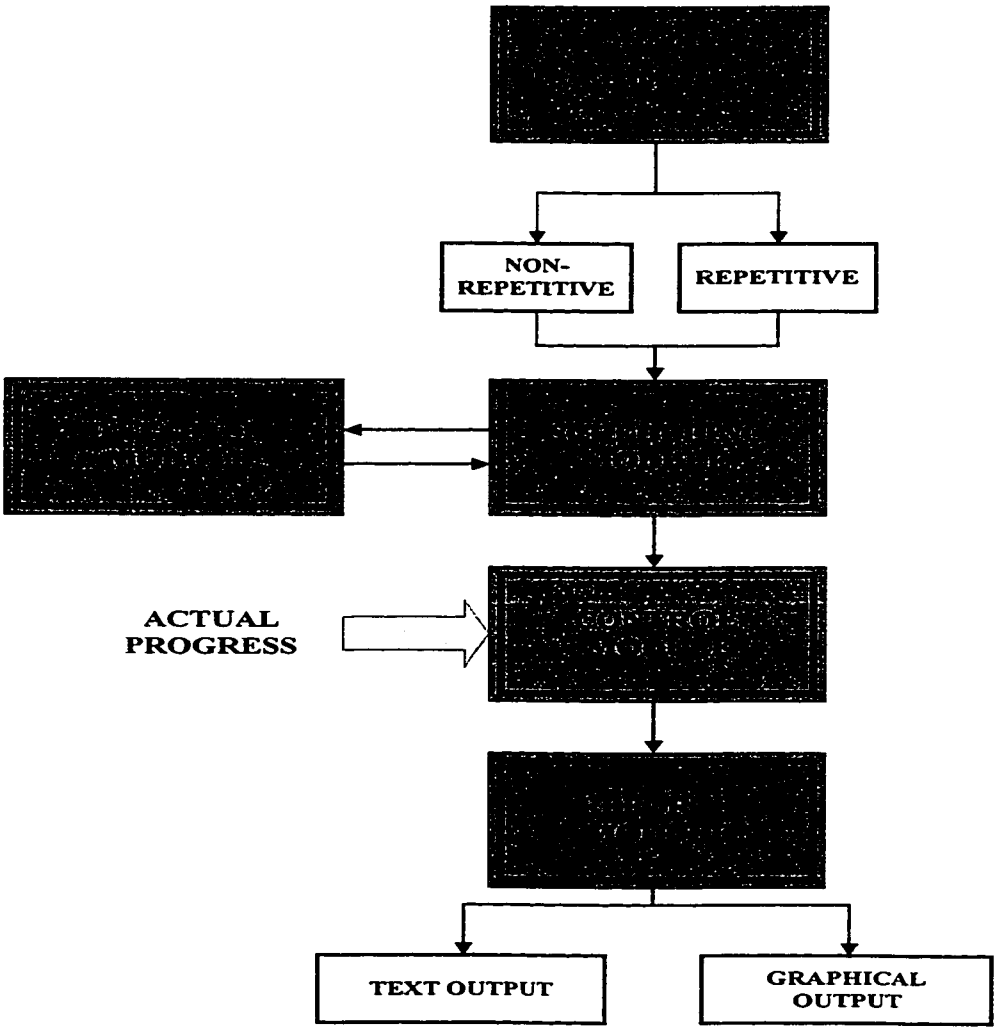


Figure 3.4 Architecture of the Proposed Model

Each module is composed of one or more classes and these classes contains data and member functions designed to perform a number of calculations such as scheduling, tracking and control and generating reports. The following section describes each module and its associated class or classes. It gives a brief discussion of each class, its data members and main member functions, and presents important state diagrams explaining various states and messages.

3.4.1 Input Module

Input Module is designed to handle the user interface aspect of the proposed model. All the user inputs are accepted and stored by this module. These inputs can vary from general project data (e.g. name of the project, location and number of units) to specific activity information (e.g. activity name, quantity and cost). This module consists of two classes namely *Project-Data* and *Project*. *Project-Data* is the most generic class in the hierarchy and *Project* class has a one to many relationships with *Project-Data*. The *Project-Data* class is similar to that developed by El-Rayes (1997) and is described in the literature review (section 2.5).

Project class has been modified and new data and functions have been added to facilitate scheduling calculations for residential construction projects, and to consider their special features and characteristics. It is designed to facilitate intake of user input data and to save these data for future use. This class reflects the typical characteristics of a residential housing project such as repetitive and

non-repetitive activities, different types of relations, activity performed by subcontractors or developers' own work force. Since this class accepts and stores all input data, it acts as a search tree for all the other classes while retrieving these data. *Project* class is designed to perform a number of needed functions such as 1) accepting new data to the project, 2) sorting and arranging activities, 3) sorting and arranging relationships among activities, 4) saving and retrieving project data and 5) initiating scheduling calculations. The main data members and member functions of *Project* class are listed in Tables 3.1 and 3.2 respectively.

Table 3.1 Data Member of *Project* Class

Data	Data Type	Description
No_Units	Integer	Number of housing units in the project.
Measurement_Unit	String	The unit of measurement to be used in the project (SI system, Imperial system or user specified).
Average_Indirect_Cost	Float	Average indirect cost of the project per day.
No_Non_Repetitives	Integer	Number of non-repetitive activities in the project.
No_Repetitives	Integer	Number of repetitive activities in the project.
No_Subs	Integer	Number of subcontractors involved in the project.
Project_Total_Cost	Float	Total project cost in $.
Project_Start	A pointer	A pointer to a date object specifying the user specified project start date.
Execution_Order	Integer	User specified integer value representing the execution order of housing units.

Table 3.2 Main Member Function of *Project* Class

Function	Description
get_no_units( )	Determines the number of housing units in the project.
get_ex_order( )	Obtains the order of execution for the housing units in the project.
start_scheduling ( )	Initiates scheduling calculations.
display_messages( )	Displays error and other messages.
get_sub_data( )	Accepts subcontractor data such as name, expected duration and availability period.

3.4.2 Scheduling Module

After the completion of data input, the general project data as well as the specific activity data are passed on to the scheduling module. This module is designed to perform numerous scheduling calculations for non-repetitive and repetitive activities. While performing scheduling, the module considers a number of practical factors commonly encountered in residential housing projects such as the type of work force utilized (i.e. developers' own force or subcontractor), precedence relationships, execution order, crew availability period at site, weather effect and learning curve.

This module comprises of ten classes. The data members and member functions of these classes are carefully designed to perform specialized functions in the scheduling process. Eight of these classes (*Activity, Regular-Relation, Repetitive, Non-Repetitive, Repetitive-Relation, Hetero-Relation, Crew-Formation and Date*) are derived on the basis of the model developed by El-Rayes (1997) and were described earlier in section 2.5. These classes, however, are modified and expanded to suit the scheduling process of residential housing projects. In

addition to these expanded classes, two new classes (*Own-Force and Sub-Contractor*) are added to consider the specialized aspects of these types of projects. The following section outlines the major modifications made in *Repetitive* class and presents a brief description of the two new classes.

Repetitive class is derived from *Activity* class and is at the third level of hierarchy (see Figure 3.2). This class is designed to consider the special characteristics of repetitive activities and to carryout its scheduling calculations. Moreover, this class is expanded to include additional data and functions to consider the specific requirements of residential housing projects such as utilizing developers' own force and/or subcontractors. Table 3.3 and 3.4 lists the newly developed data and member functions, respectively.

Table 3.3 New Data Members of *Repetitive* Class

Data	Data Type	Description
Act_Perform_By	Integer	A user specified value indicating whether a repetitive activity is performed by using own force or by subcontractor. (0 and 1 represents own force and subcontractor, respectively)
Data_Base	Integer	An integer value indicating whether the user has opted to use database support for crew information.
Act_Exe_Order	A pointer	A pointer to an array where the user specified execution order of a repetitive activity is stored. This data member is activated only when the user specifies that the execution order of an activity is different from that applied to other activities at the project level. It should be noted that this array is initially assigned the execution order of houses at the project level, by default.

Table 3.4 New Member Functions of *Repetitive* Class

Function	Description
find_act_perform( )	Determines whether the activity is performed by own force or by subcontractor.
find_act_exe_order( )	Search for and determine the execution order of a repetitive activity.

Own-Force class is designed to represent the work done in a repetitive activity using the developers' own work force. This class is derived from *Repetitive* super-class and inherits all its data members and member functions. In addition to the inherited attributes, specialized data members and functions are added to reflect the specific aspects associated with utilizing the developers' own work force. These data members and main member functions are listed in Table 3.5 and 3.6, respectively.

Table 3.5 Data Members of *Own-Force* Class

Data	Data Type	Description
Dir_Cost_Own	Float	Budgeted direct cost for each repetitive activity performed by developers' own force.
Total_Cost_Own	Float	Total budgeted cost for all the activities performed by developers' own force.

Table 3.6 Main Member Functions of *Own-Force* Class

Function	Description
bud_own_cost( )	Invokes calculations to determine the budgeted direct cost for each repetitive activity performed by developers' own force.
total_own_cost( )	Invokes calculations to determine the total budgeted cost for all the activities performed by developers' own work force.

Sub-Contractor class is designed to consider the repetitive activities performed by subcontractors. As stated earlier, subcontractors perform most of the construction work in a typical residential development project. Hence, extreme care is exercised in designing this class, which provides added flexibility and practicality to the proposed model. This class inherits general data from its super-class *Repetitive* and includes additional specialized subcontractor data and functions. These data members and main member functions are listed in Table 3.7 and 3.8, respectively.

Table 3.7 Data Members of *Sub-Contractor* Class

Data	Data Type	Description
Sub_Name	String	Name of the subcontractor.
Contract_Type	Integer	Type of contract awarded to the subcontractor. (i.e. unit price or lump sum contract).
Exp_Dur	Integer	Expected duration for the subcontracted activity in one house or all housing units.
Sub_Cost	Float	Budgeted direct cost for each subcontracted repetitive activity.
Total_Sub_Cost	Float	Total budgeted direct cost for all the activities performed by subcontractors.

Table 3.8 Main Member Functions of *Sub-Contractor* Class

Function	Description
budget_cost_sub_unit( )	Invokes the cost calculations for an individual subcontractor with a unit price contract.
sub_total_cost( )	Invokes the calculations to determine the total direct cost for all the subcontractors involved in a project.

The functions activated by this module while performing scheduling calculations differ according to the type of the activity (non-repetitive or repetitive) and whether developers' own work force or subcontractors are employed. Non-repetitive activities are scheduled using network technique method and repetitive activities performed by developers' own force are scheduled according to the resource driven scheduling algorithm proposed by El-Rayes (1997). Activities performed by subcontractor are scheduled using the proposed algorithm explained later in Chapter 4.

Functions performed in scheduling repetitive activities can be represented through a state diagram for *Repetitive* class as shown in Figure 3.5. The state diagram shows the changes in the state of the class along with the messages it receives and/or sends. The scheduling calculations for repetitive activities are initiated in order to identify the start and finish date of each housing unit. This is followed by identifying whether the repetitive activity is performed by developers' own force or subcontractors. The activity, then, examines if it has received information from all its predecessor activities. If this condition is true, forward pass calculations are initiated by sending a message to *Crew-Formation* or *Sub-Contractor* class, which in response identifies the early start and finish date for each housing unit.

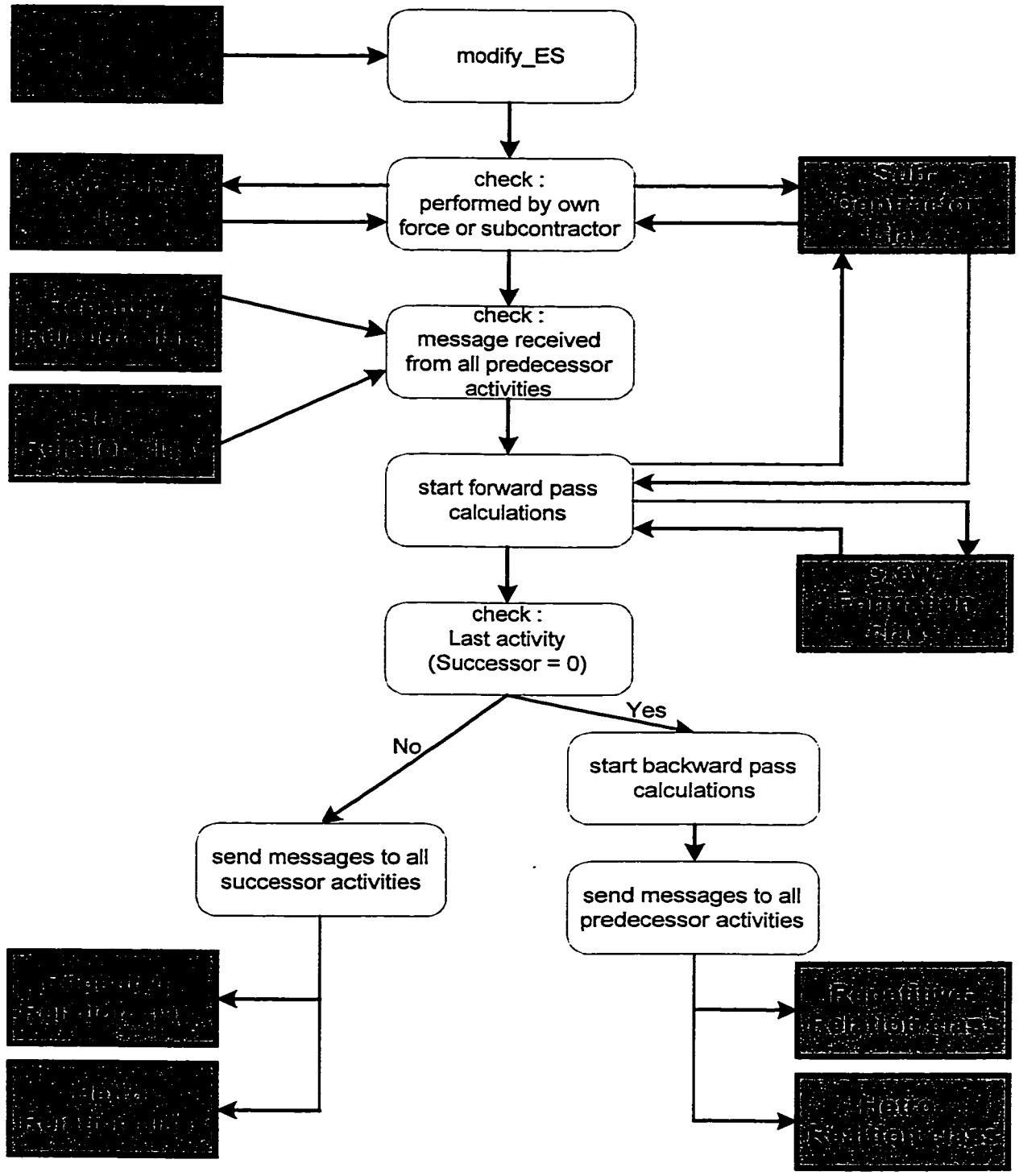


Figure 3.5 State Diagram of *Repetitive* Class

After finishing the forward pass calculations, each activity object checks whether it has any successor activity. If so, it sends messages to all of its successor

activities informing them the early start and finish dates in all housing units. If not, backward pass calculations are initiated to determine the late start and finish dates for each activity in each housing unit as shown in Figure 3.5. Backward pass calculations are similar to those of forward pass, except they start from the last activity and proceed to finish with the first activity.

3.4.3 Database Module

The proposed model includes a database support to store historical data that can be retrieved for use in future projects. The data stored in the database includes relevant crew data such as crew number, crew composition and crew cost. This module is expandable and designed to enable the addition of new data, thereby making it adaptable to growing demands. The module comprises of a number of text files, representing the common activities of residential housing construction. The data members and functions needed to perform storing, searching and retrieving data are developed and grouped together in *Data-Base* class.

Data-Base is a stand-alone class which has a one to many relationship with *Activity* class. An Activity object can retrieve data (e.g. crew daily output and crew cost) from *Data-Base* class by sending a message or request. This class performs a number of needed functions such as 1) adding new data to the database, 2) searching the database to respond to a request from an Activity object, 3) retrieving the requested data and 4) displaying all the available data.

The data members and main member functions of this class are listed in Table 3.9 and 3.10 respectively.

Table 3.9 Data Members of *Data-Base* Class

Data	Data Type	Description
DB_Crew_No	String	A specific crew number representing each crew formation. This constant variable is used as a key in the search process. (For example, an excavation crew can have a crew number such as B-12A (Means, 1999).
DB_Crew_Comp	String	Gives a detail description of each crew (For example, the above excavation crew comprises of one equipment operator and a hydraulic excavator of 1C.Y. capacity.
DB_Crew_Output	Float	Crew daily output (Unit/ Day).
DB_Mat_Cost	Float	Material cost rate ($/Unit).
DB_Lab_Cost	Float	Crew labor cost rate ($/Day).
DB_Eqp_Cost	Float	Crew equipment cost rate ($/Day).
DB_Act_Name	String	Name of the activity that is used in the search process.
DB_Select	Integer	An integer value representing the user selection from the list of available crews in the database for a particular activity.

Table 3.10 Main Member Functions of *Data-Base* Class

Function	Description
add_to_database( )	Allows new data to be added to the database.
search_database( )	Invokes a search after receiving a message from an Activity object.
display_database( )	Displays all the successful search results so that the user can select from them.
select( )	Exchanges the user-selected crew information data (output, cost etc.) with the activity object. These exchanged data are used in the scheduling calculations.

Data-Base class performs different set of functions in retrieving and storing data as shown in Figures 3.6 and 3.7. While searching and retrieving data, this class

identifies the table corresponding to the activity requested by the Activity object and displays it, so that the user can select among them.

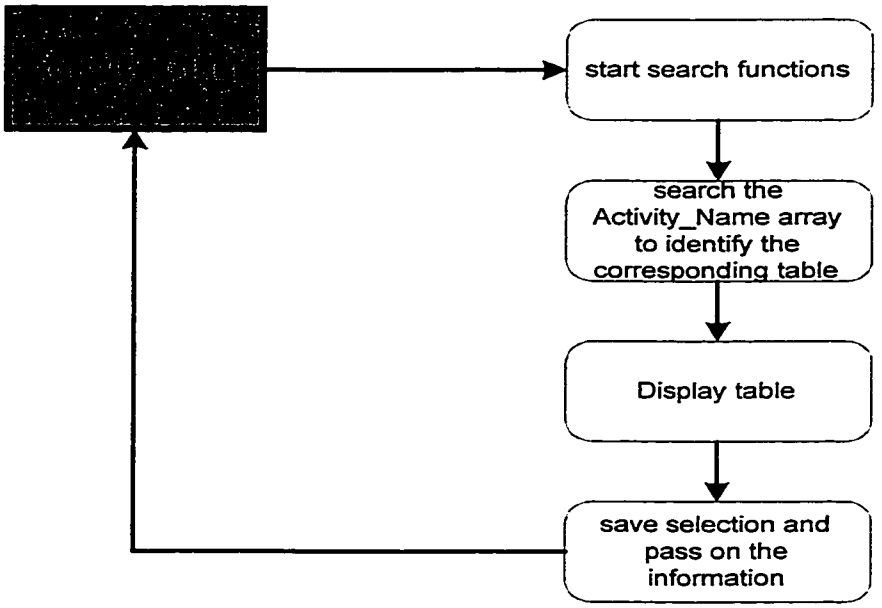


Figure 3.6 State Diagram for *Data-Base* Class to Perform Search

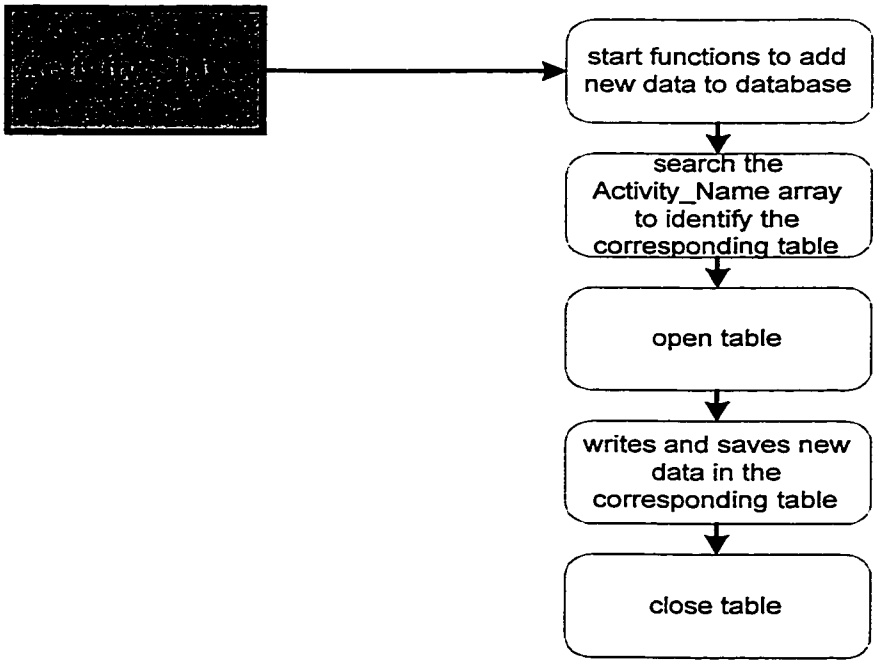


Figure 3.7 State Diagram for *Data-Base* Class to Accept New Data

While storing new data, a set of functions are performed as shown in Figure 3.7. The first function determines the database table corresponding to the activity name to which new data are to be written. Once the table is identified, it is opened and the new data are written and saved. It should be noted that the function, which writes and saves new data, skips all the already stored data and writes the data in the first available vacant line. This effectively eliminates the possibility of loosing already stored data due to overwriting. Once the new data are stored the table is closed, thereby making it available for future use.

3.4.4 Control Module

Control module is designed to consider the needed tracking and control features for residential housing projects discussed earlier in the analysis stage. The module receives the base-line schedule from the scheduling module and uses it in the tracking and control calculations. At any given period of time, the user can input the actual progress data and compare it with the planned progress. The module performs tracking and control calculations at three different levels (i.e. at the project, subcontractor and house). The cost and time performance of an on-going project, an individual subcontractor or a particular housing unit is determined using the developed algorithm explained later in Chapter 4. The control module comprises of four classes namely, *Tracking-Control, Project-Control, Unit-Control and Sub-Control.*

Tracking-Control class is at the highest level of the control module hierarchy (see Figure 3.2). It has general data and member functions designed to perform tracking and control calculations. Moreover, it has a one to many relationship with *Activity* class. This relationship enables the *Tracking-control* class to obtain the base-line schedule, activity and subcontractor details. The main purpose of this class is to act as an interface class between the base-line schedule and its subclasses. In addition, this class includes two data members and a member function, which are listed in Table 3.11 and 3.12, respectively.

Table 3.11 Data Members of *Tracking-Control* Class

Data	Data Type	Description
Control_Day	Integer	Date of reporting.
Status	Integer	An integer value representing the status of a project, subcontractor or a housing unit (1, –1 and 0 represents the project is ahead, behind and on schedule/budget, respectively).

Table 3.12 Main Member Functions of *Tracking-Control* Class

Function	Description
find_status( )	Initiates the calculations needed to determine the time and cost status of a project, subcontractor or a housing unit.

Project-Control class is derived from *Tracking-Control* class, and accordingly it inherits all the attributes of its super-class. This class is designed to determine the time and cost performance status of an on-going project. It includes a number of data members and functions that are listed in Tables 3.13, and 3.14, respectively.

Table 3.13 Data Members of *Project-Control* Class

Data	Data Type	Description
Proj_No_Act_Comp	Integer	Number of activities completed till the day of reporting.
Proj_No_Act_Prog	Integer	Number of activities started but not completed on the day of reporting.
Proj_Act_Comp_Name	String	Name of the activity that was completed.
Proj_Act_Prog_Name	String	Name of the activity in progress.
Proj_Act_Comp_Start	Integer	Start date of a completed activity.
Proj_Act_Comp_Finish	Integer	Finish date of a completed activity.
Proj_Act_Prog_Per	Float	Percentage of work completed in an activity in progress.
Proj_Schd_Cum_Prog	A pointer	A pointer to an array of size [Project_Duration], which includes the planned cumulative work of a project.
Proj_Act_Cum_Prog	A pointer	A pointer to an array of size [Control_Day], which includes the actual cumulative work of the project.
Proj_Schd_Cum_Cost	A pointer	A pointer to an array of size [Project_Duration], which includes the planned cumulative cost of the project.

Table 3.14 Main Member functions of *Project-Control* Class

Function	Description
proj_schd_progress( )	Invokes the calculations to determine the planned cumulative work. These calculations are carried out using the developed algorithm explained later in Chapter 4 (section 4.3.1).
proj_act_progress( )	Invokes the calculations to determine the actual cumulative work.

The functions performed to determine the time and cost performance of an on-going project can be represented through a state diagram for *Project-Control* class as shown in Figure 3.8. This state diagram shows the sequence of the functions that are activated and the messages received and/or sent.

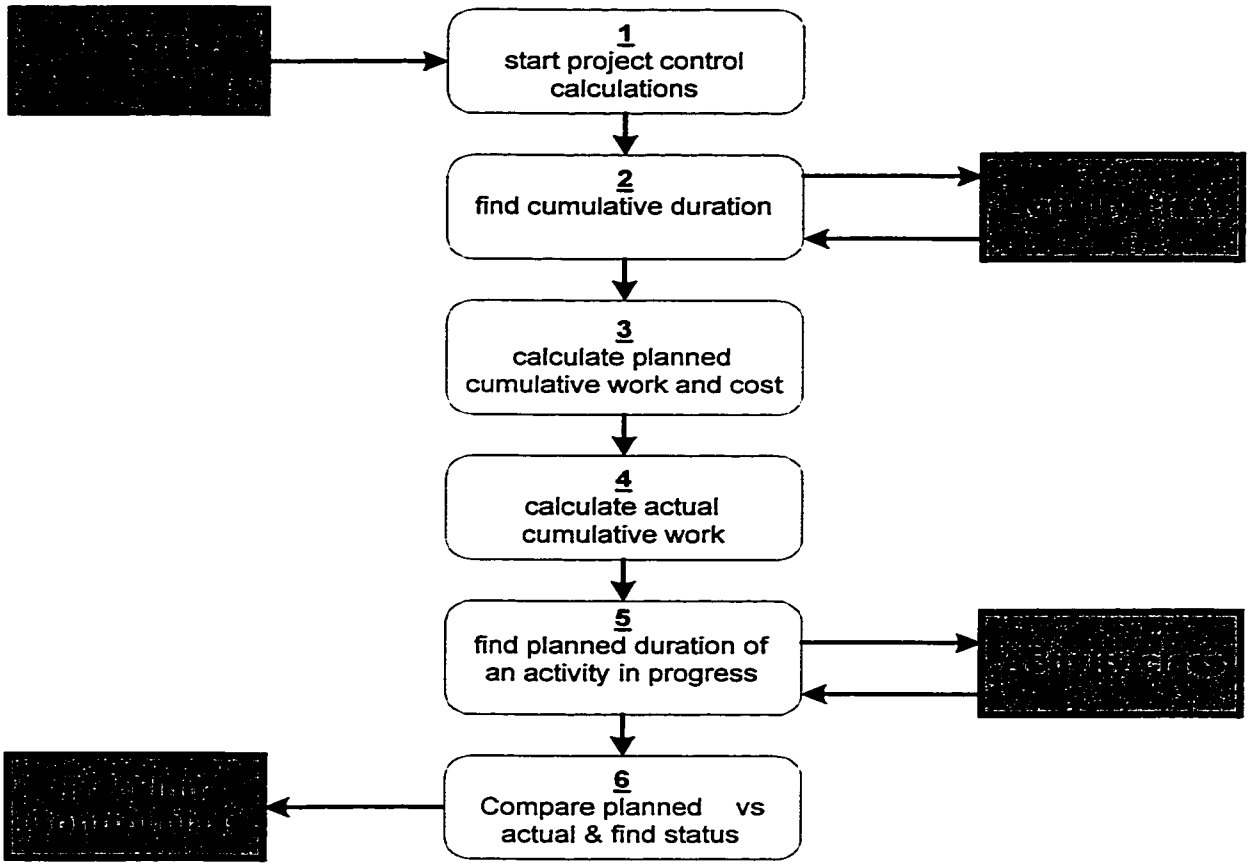


Figure 3.8 State Diagram of *Project-Control* Class

The first step in project tracking and control calculations is invoked by a message or request from *Tracking-Control* class, which in-turn initiates control calculations. The second step determines the cumulative duration of all activities (repetitive and non-repetitive), by sending and receiving messages from *Activity* class. The third step calculates the planned cumulative work and cost according to the base-line schedule and the fourth step determines the actual cumulative work according to the actual progress data input. The calculations in the third and fourth steps are performed using the proposed tracking and control algorithm explained later in Chapter 4 (section 4.31). The fifth step obtains the planned duration of an activity in progress by sending a message to the *Activity* class and

accordingly calculates the percentage of work completed. The final step compares the actual and planned performance to determine the time and cost status of the project, and sends a message to the *Tracking-Control* class.

Unit-Control class is derived form *Tracking-Control* super-class and is designed to determine the progress status of a single house in a housing development project. In addition to the generalized data inherited from its super-class, *Unit-Control* class includes additional specialized data members and functions as listed in Table 3.15 and 3.16, respectively. The sequence of functions performed by *Unit-Control* class is similar to that of *Project-Control* class (see Figure 3.8), except for that the data corresponding to a particular housing unit are considered instead of the whole project.

Table 3.15 Data Members of *Unit-control* Class

Data	Data Type	Description
Unit_Control_No	Integer	House number for which the status is to be determined. (e.g. house # 5).
Unit_Control_Comp	Integer	Number of activities completed in this particular housing unit.
Unit_Control_Prog	Integer	Number of activities in progress in this housing unit.
Unit_Control_Comp_Name	String	Name of completed activities.
Unit_Control_Prog_Name	String	Name of activities in progress.
Unit_Control_Comp_Start	Integer	Start date of completed activities.
Unit_Control_Comp_Finish	Integer	Finish date of completed activities.
Unit_Control_Prog_Per	Float	Percentage of work completed in activities in progress.
Unit_Control_Schd_Prog	A pointer	A pointer to an array of size [Unit_Duaration], which includes the planned cumulative work of a housing unit.
Unit_Control_Act_Prog	A pointer	A pointer to an array of size [Control_Day], which includes the actual cumulative work.

Table 3.16 Main Member Functions of *Unit-Control* Class

Function	Description
unit_control_schd_prog( )	Invokes the planned cumulative work calculations for a unit.
unit_control_act_prog( )	Invokes the actual cumulative work calculation for a unit.

Sub-Control class is inherited from *Tracking-Control* super-class and is designed to determine the time and cost performance status of an individual subcontractor. In addition to the inherited data, new data and member functions are included in this class in order to consider the special aspects of tracking and control of subcontractors. The data members and member functions of this class are listed in Table 3.17 and 3.18, respectively.

Table 3.17 Data Members of *Sub-Control* Class

Data	Data Type	Description
Sub_Control_Name	String	Name of the subcontractor whose performance is to be determined.
Sub_Control_Activity	String	Name of the activity performed by the subcontractor.
Sub_Control_Units_Comp	Integer	Number of houses completed by the subcontractor.
Sub_Control_Units_Prog	Integer	Number of houses started by the subcontractor but not finished.
Sub_Control_Comp_Start	Integer	Start date of completed houses.
Sub_Control_Comp_Finish	Integer	Finish date of completed houses.
Sub_Control_Prog_Per	Float	Percentage of work completed.
Sub_Duaration	Float	Planned duration of the activity performed by the subcontractor.
Sub_Control_Schd_Prog	A pointer	A pointer to an array of size [Sub_Duaration], which includes the planned cumulative work of the subcontracted activity.
Sub_Control_Act_Prog	A pointer	A pointer to an array of size [Control_Day], which includes the actual cumulative work of the subcontracted activity.

Table 3.18 Main Member Functions of *Sub-Control* class

Function	Description
sub_control_schd_prog( )	Invokes planned cumulative work calculations.
sub_Control_act_prog( )	Invokes actual cumulative work calculations.

The functions performed and the messages exchanged by *Sub-Control* class in order to determine the status of a subcontractor can be represented by a state diagram as shown in Figure 3.9. The first step in sub-control calculations is invoked by a message from Tracking-Control class. This message performs two tasks: 1) provides the *Sub-Control* class with the name of the subcontractor whose performance is to be evaluated and 2) initiates tracking and control calculations. The second and third steps determine the activity performed by this particular subcontractor and the planned cumulative duration of this activity, respectively. This is achieved by sending and receiving messages to and from *Activity* class. The fourth step calculates the planned cumulative work and cost performance according to the developed base-line schedule and the fifth step determines the actual work progress according to the actual progress input data. The sixth step, exchanges messages with Activity class, whenever needed, to obtain the planned duration of the subcontracted activity in housing units in which work has started but not finished, in order to calculate the percentage of work completed. The final step determines the time and cost status by analyzing the planned and actual performance and informs the *Tracking-Control* class.

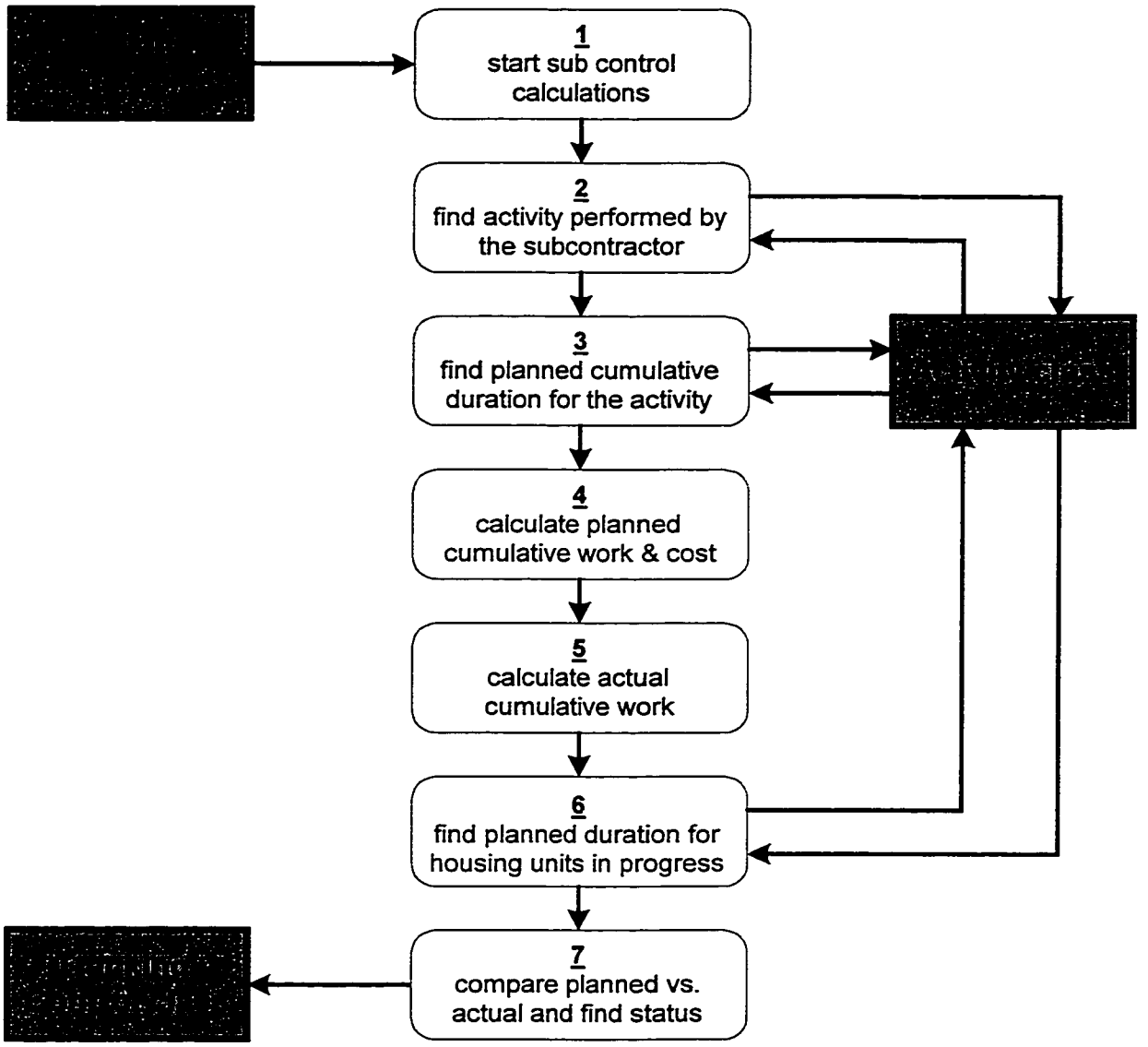


Figure 3.9 State Diagram for *Sub-Control* Class

3.4.5 Reports Module

The reports module is the last of the five modules comprising the proposed scheduling and tracking and control model for residential housing projects. This module is designed to provide two types of outputs (i.e. text and graphical output) as shown in Figure 3.4. The needs and requirements of residential development

firms discussed earlier in the analysis stage are carefully considered in the design of this module. The module is capable of producing both graphical and text outputs, specifically designed for residential housing projects. Functions and data members, which perform various calculations and display reports are developed and grouped together in *View* class. *View* class is designed as a stand-alone class, which has one to many relationships with *Activity* class and *Tracking-Control* class (see Figure 3.2). These relationships enable the *View* class to extract the entire schedule and control information needed to generate and display various reports. Data members and main member functions of the *View* class are listed in Table 3.19 and 3.20, respectively.

Table 3.19 Data Members of *View* Class

Data	Data Type	Description
Scale_X	Float	Value by which the X-axis is equally divided. This value varies according to the item represented by the X-axis and its quantity. (e.g. if the project duration is 60 days, then the X-axis will be equally divided into 60 sections when displaying the graphical project schedule.)
Scale_Y	Float	Value by which Y-axis is equally divided.
Cstring	String	A string to represent characters used in displaying reports.
Font	Integer	An integer value representing the size of font used to produce the reports.
Developer	String	Name of the project development firm.
Scheduler	String	Name of the person preparing the schedule.
Location	String	Name of location of the project.

Table 3.20 Main Member Functions of *View* class

Function	Description
show_scale_x( )	Calculates and divides the X-axis to equal sections.
show_scale_y( )	Calculates and divides the Y-axis to equal sections.
show_font( )	Displays character variables in reports.
graphical_output_nonrep( )	Displays the schedule for all non-repetitive activities in the project in a graphical format.
graphical_output_rep( )	Displays the schedule for all repetitive activities in the project in a graphical format.
project_schedule_workday( )	Produces the project schedule report in workdays.
project_schedule_calendar( )	Generates project schedule report in calendar dates.
graphical_output_subcontract( )	Produces the schedule for a particular subcontractor in a graphical format.
unit_schedule_workday( )	Produces the schedule for each housing unit in workdays.
graphical_output_progress_ curve( )	Generates a graph displaying the planned and actual work progress curves for the project.
graphical_output_sub_progress _curve( )	Generates a graph displaying the planned and actual work progress curves for a subcontracted activity.
graphical_output_unit_progress _curve( )	Generates a graph displaying the planned and actual work progress curves for a particular housing unit.
graphical_output_cost_curve( )	Generates the planned cumulative cost curve for the project.
control_result( )	Displays the tracking and control results.
project_summary( )	Displays the summary of the project.

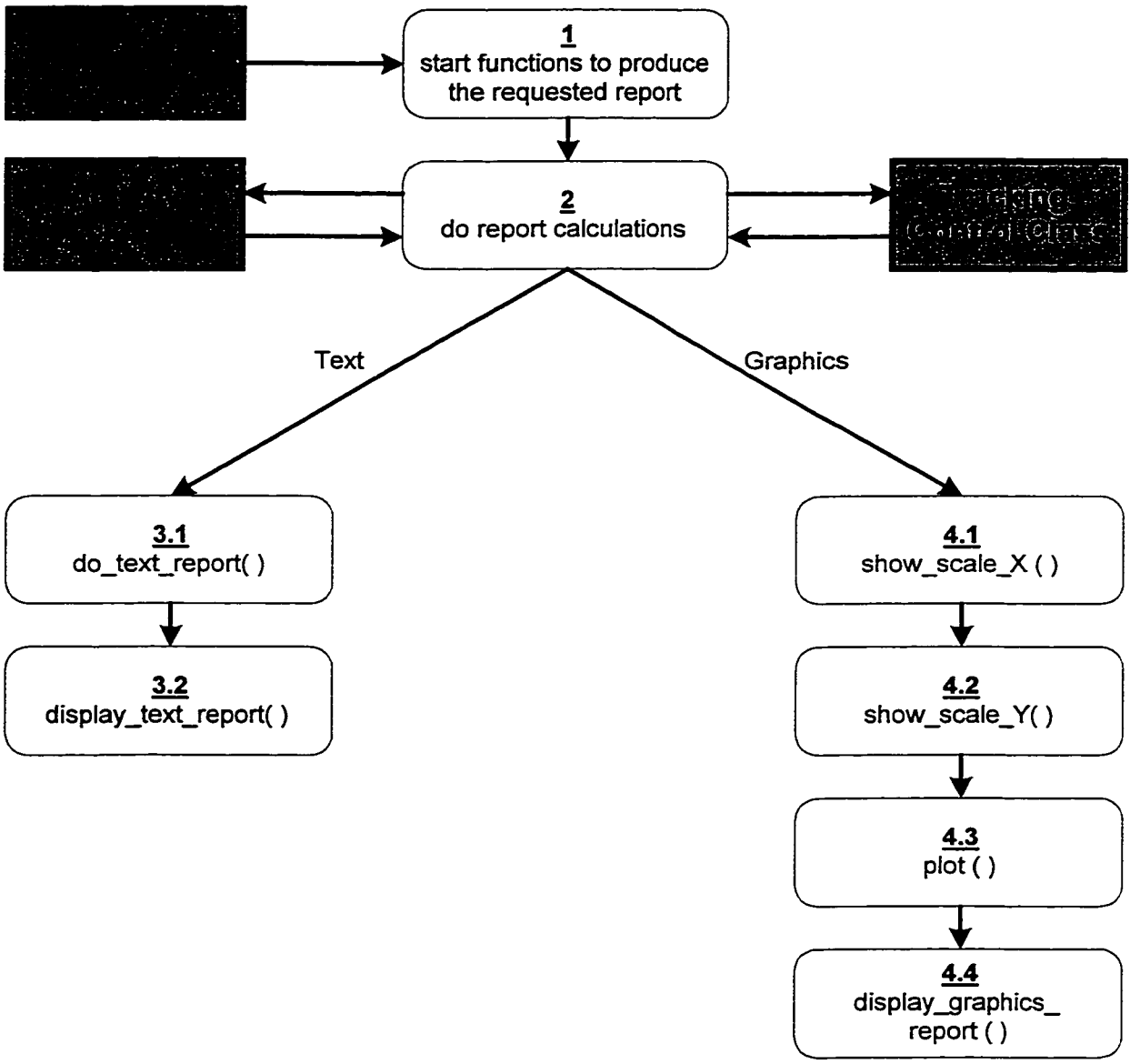


Figure 3.10 State Diagram for *View* Class

A state diagram for *View* class can be used to represent the sequence of functions performed in generating a report as shown in Figure 3.10. The first step in this sequence is activated by a message from *Project* class, which informs the *View* class of the type of report to be produced. The second step performs the needed calculations to generate the report by exchanging messages with *Activity*

and/*or Tracking-Control* class. The third step in this sequence is activated in order to produce and display the text format report. The fourth step is invoked when the produced report is of the graphical format as shown in Figure 3.10. The functions performed in the fourth step include 1) dividing the X-axis equally according to the item it represents (e.g. project duration), 2) dividing the Y-axis (e.g. number of housing units), 3) plotting the curves and/or lines needed for the report, 4) displaying the generated report.

3.5 Summary

An Object model for scheduling and tracking and control of residential housing projects has been presented in this chapter. The proposed model consists of five modules namely, Input module, Scheduling module, Control module, Database module and Reports module. The model is developed using Object-Oriented modeling concepts. In order to identify the practical factors affecting the scheduling and control of residential housing projects, a number of interviews were conducted with the representatives of residential development firms. The findings of these meetings are outlined in the analysis stage and were carefully considered in the model development. An overview of the model classes, their data members and member functions along with a number of important state diagrams have also been presented. The following Chapter presents the design of a scheduling algorithm for subcontractors and a tracking and control algorithm that are incorporated and used in the present model.

CHAPTER 4

PROPOSED SCHEDULING AND CONTROL ALGORITHMS

4.1 Introduction

This chapter presents the development of algorithms for scheduling subcontractors and for tracking and control of residential housing projects. The development of these two algorithms builds on the analysis and design stages, and considers a number of practical factors commonly encountered in residential housing projects highlighted in previous Chapters. The scheduling calculations for subcontractors are performed by the *Crew-Formation* and *Sub-Contractor* classes. Tracking and control calculations are performed by *Tracking-Control* class and its sub-classes. These calculations are performed by exchanging messages among various classes.

4.2 Scheduling Algorithm for Subcontractors

As stated earlier, most of the construction tasks in a residential housing project are performed by subcontractors. In many instances, the data available to schedule repetitive activities performed by subcontractors are limited. More often, construction contracts are awarded to subcontractors with an expected duration to finish a housing unit or all housing units along with a unit price or lump sum contract. Hence, in most cases the schedule for subcontractors have to be developed only with the expected duration that will take to finish construction work in a house or all houses.

The present scheduling algorithm for subcontractors expands on the recently developed algorithm for resource-driven scheduling of repetitive activities (El-Rayes and Moselhi, 1998) in order to consider the special features of scheduling subcontractors. The present algorithm considers three major constraints namely, 1) precedence relationships, 2) subcontractor availability period and 3) crew work continuity. Precedence relationship constraint represents the job logic among activities. Subcontractor availability constraint depends on the availability period of a subcontractor on site and its impact on the developed schedule. Crew work continuity can be provided if the schedule allows a construction crew to finish an activity in one house and be able to move to the next without delay. This smooth movement of crews is of utmost importance to subcontractors since they are more concerned about continuity of work for their crews than critical paths or early completion. Application of crew work continuity provides an effective resource utilization that leads to minimum idle time and/or reduction in hiring and firing.

A number of practical factors commonly encountered in scheduling repetitive activities performed by subcontractor are also considered in the present algorithm. These factors include: 1) expected duration of a housing unit or all housing units, 2) typical and non-typical repetitive activities, 3) sequence of construction operations and 4) type of contract (i.e. unit price or lump sum). The algorithm is applied in two stages (see Figures 4.1 and 4.2) to determine the start and finish dates of each housing unit. Stage 1 considers the precedence

relationships and subcontractor availability constraints and stage 2 considers crew work continuity constraint.

Input data required for the present algorithm are: 1) number of housing units, 2) earliest available date of the subcontractor, 3) estimated construction duration either in a single housing unit (e.g. 2 workdays to finish carpeting in a house) or all housing units (e.g. 30 workdays to finish carpeting in 20 houses) and 4) execution order of the housing units. It should be noted that the algorithm is designed to provide practicality and flexibility in its requirement for data input. It allows the user to input the estimated duration needed by the subcontractor to complete the construction work in either: 1) a single housing unit or 2) all housing units. For the first option, the algorithm utilizes the provided construction duration for a single housing unit as such in the scheduling calculations. For the second option, however, the algorithm calculates the duration for each housing unit based on the provided duration for all housing units. This is achieved by determining the required productivity (i.e. Total quantity of work in all housing units / expected duration) of the subcontractor, and accordingly calculating the duration.

4.2.1 Stage 1

For each repetitive activity performed by subcontractor n, the scheduling calculations starts by initializing the possible start array (PS[n]), with the user-specified early available date as follows:

PS[n]=sub_avail_start[n] (4.1)

Where,

PS[n] :– next possible start of subcontractor n,

sub_avail_start[n] :– user-specified early available date of subcontractor n at site.

The algorithm allows the user to specify the earliest available date of the sub-contractor on site. However, it should be noted that the algorithm provides a default value of zero, indicating that the sub-contractor can be at site whenever needed, unless otherwise specified. Having initialized the possible start array, the algorithm proceeds with scheduling calculations as shown in Figure 4.1. For each housing unit (j), the algorithm:

1) Determines whether the provided construction duration is for a housing unit or for the entire housing project and accordingly proceeds with the calculations. If the provided duration is for a housing unit, then this duration is used as a direct input in the calculations and skips to step 5. Otherwise, calculations to determine the duration of each housing unit is performed (steps 2 to 4).

2) Calculates the total quantity of work to be performed by the subcontractor. This is determined by adding the quantity of work in each housing unit (j=1 to j=J) in loop 1, where, J is the total number of housing units, as follows:

Total_Qty[n]=Total_Qty[n]+Qty[j] (4.2)

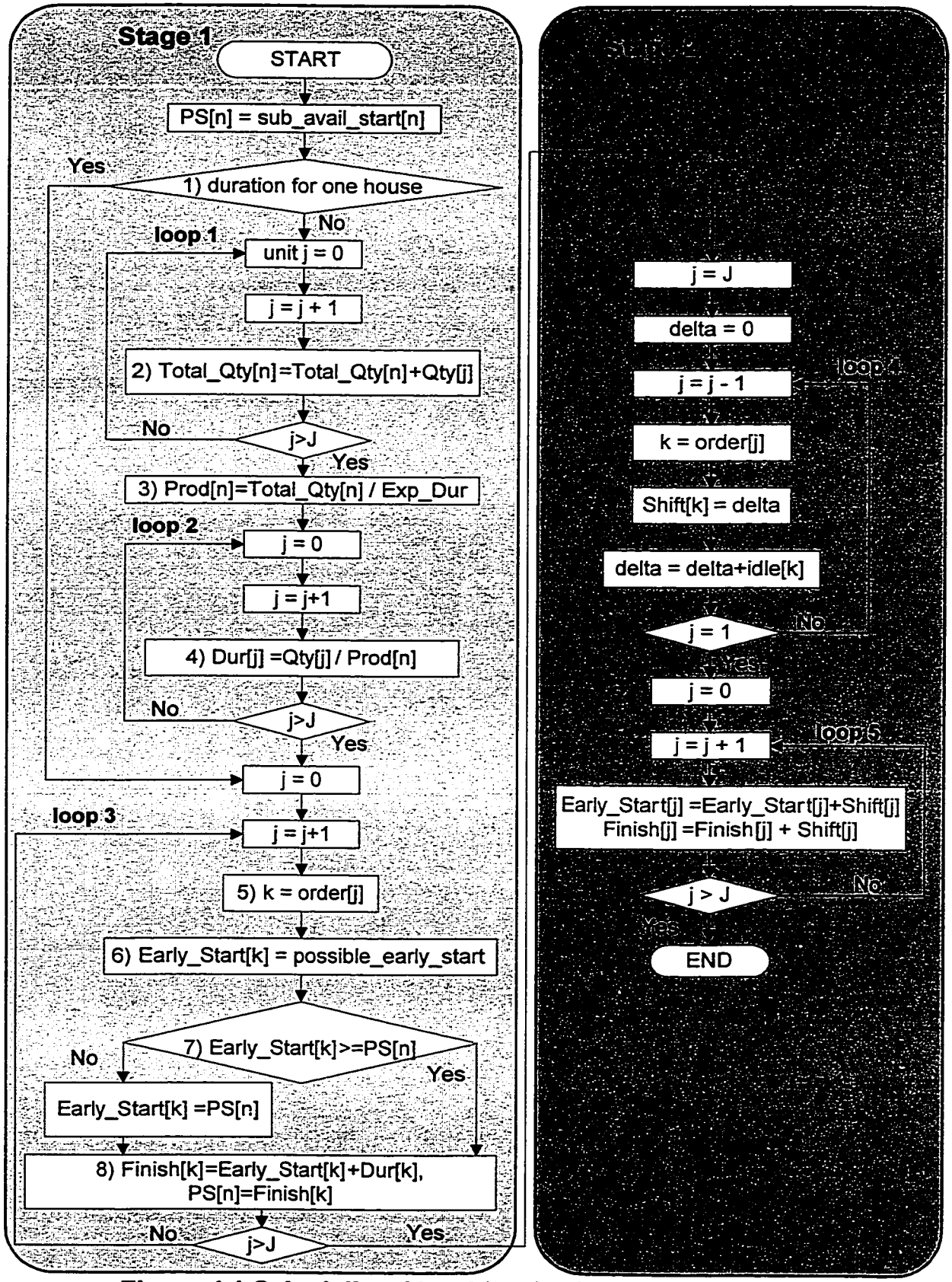


Figure 4.1 Scheduling Algorithm for Subcontractors

Where,

Total_Qty[n] :- total quantity of work to be performed by subcontractor n,

Qty[j] :– quantity of work in each housing unit j.

3) Determines the productivity of the subcontractor per day, as follows:

$$Prod[n] = Total\_Qty[n] / Exp\_Dur \quad (4.3)$$

Where,

Prod[n] :– daily output of the subcontractor n,

Exp_Dur :– user-specified expected duration to finish the activity in all housing units.

4) Calculates the duration of each housing unit (j=1 to j=J) using the required productivity determined in step 3 in loop 2 as follows:

$$Dur[j] = Qty / Prod[n] \quad (4.4)$$

Where,

Dur[j] :– Duration of housing unit j.

5) Identifies the housing unit in conformity with the user-specified execution order, as follows:

$$k = order[j] \quad (4.5)$$

Where,

Order[j] :– user-specified execution order,

k :– unit number that satisfies the user-specified execution order.

The algorithm provides the user with three options to specify the execution order of housing units. These are 1) ascending order (order[j] = 1,2,3,4...J), 2) descending order (order[j] = J, J-1, J-2,...,3,2,1) and 3) user-specified order (order[j] = 3,1,4,6,J.... 5) where, J is the number of housing units. The algorithm is designed to provide ascending order as default value, unless otherwise specified.

6) Calculates the possible early start date that satisfies the logical precedence relationships and assigns to 'Early_Start' variable as follows:

$$Early\_Start[k] = possible\_early\_start \tag{4.6}$$

Where,

Early_Start[k] :– early start date of housing unit k,

possible_early_start :– earliest possible start of housing unit k due to precedence relationships.

7) Examines whether the assigned early start date complies with the subcontractor availability period. This is determined by checking whether the assigned early start of housing unit k is greater than or equal to possible early start date (PS[n]) of subcontractor n. If this condition is false, the early start date of unit k is assigned the user-specified available date for the subcontractor (PS[n]). Otherwise skips to step 8.

8) Calculates the finish dates of each housing unit and assigns it to the possible start array as follows:

$$Finish[k] = Early\_Start[k] + Dur[k] \quad (4.7)$$

$$PS[n] = Finish[k] \quad (4.8)$$

Where,

Finish[k] :– finish date of housing unit k,

Dur[k] :– Duration of housing unit k.

9) Repeats steps 1 to 8 in order determine the start and finish dates of all units (j=j+1 to J) in loop 3 as shown Figure 4.1. As such, the developed schedule complies with precedence relationships and subcontractor availability period.

4.2.2 Stage 2

As stated earlier, the subcontractor schedule developed in stage 1 does not guarantee work continuity for the construction crews. This crew work continuity criteria is considered in stage 2 of the proposed algorithm. As shown in Figure 4.2 there are idle time in several units (1,5,7,9 and 10). In order to eliminate crew idle time and to maintain crew work continuity, the developed schedule is shifted. The shift needed is calculated for all housing units assigned to each subcontractor n in loop 4 and 5 as shown in Figure 4.1. This required shift is determined by adding the idle times in all units.

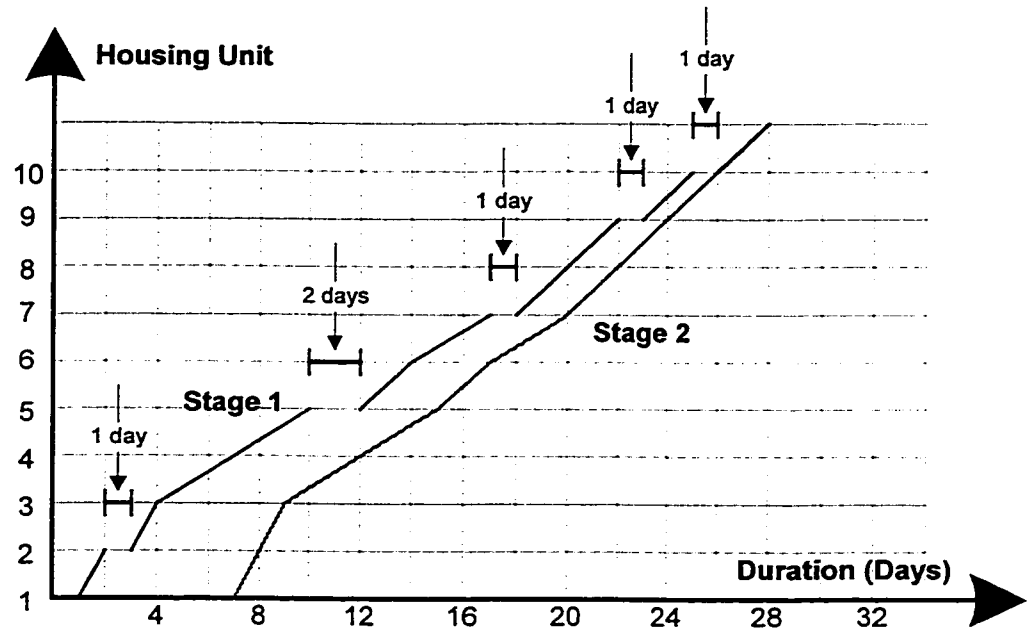


Figure 4.2 Stages 1 and 2

The calculated required shift is used to shift the scheduled start and finish dates of stage 1 in loop 5 as follows:

$$Early\_Start[j]=Early\_Start[j]+Shift[j] \quad (4.9)$$

$$Finish[j]=Finish[j]+Shift[j] \quad (4.10)$$

Where,

Shift[j] :– Required shift to the start and finish dates of unit j to comply with crew work continuity constraint

4.3 Tracking and Control Algorithm

The purpose of the developed tracking and control algorithm is to determine the time and cost status of an on-going project at three different levels: 1) project, 2) housing unit and 3) subcontractor. Tracking and control calculations are performed by the control module, which comprises of *Tracking-Control* super

class and three of its sub-classes namely, *Project-Control, Unit-Control and Sub-Control* (see Figure 3.2). Depending upon the user request of whether to determine the status of the project, a particular housing unit, or an individual subcontractor, Tracking-Control object sends a message to one of the corresponding sub-classes invoking the necessary calculations.

The proposed algorithm is applied in three stages to determine the time and cost performance of a project as a whole, a particular housing unit, or an individual subcontractor. The calculations performed in these three stages are shown in Figure 4.3. The first stage determines the planned cumulative work and cost for each workday according to the developed base line schedule. The second stage calculates the cumulative actual work performed (AWP) up to report date (r) based on the user-specified actual progress data. The third stage first determines the planned duration for work performed (PDWP) and the budgeted cost for work performed (BCWP) as shown in Figure 4.3. This stage then compares the actual and planned duration for work performed (i.e. ADWP and PDWP) in order to calculate the schedule variance (SV). The SV value can be 0, <0 or >0, representing on, behind or ahead of schedule, respectively. The cost status is determined by comparing the budgeted cost for work performed (BCWP) to the actual cost for work performed (ACWP) to calculate the cost variance (CV). The value of CV can be 0, >0 or <0 representing on, under or over budget, respectively.

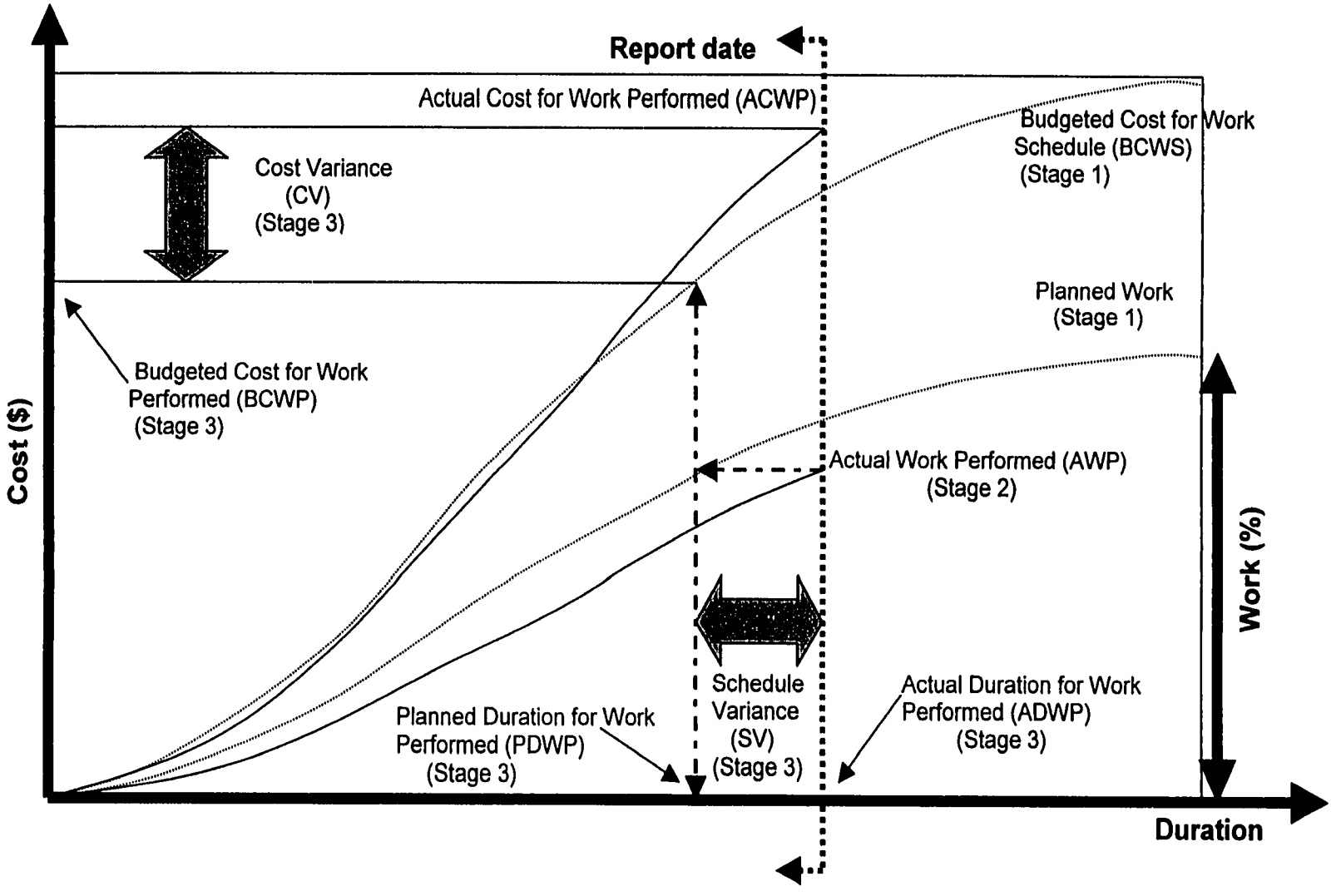


Figure 4.3 Tracking and Control Calculations

4.3.1 Project Control

Project-Control class is designed to perform the above-described calculations in order to determine the time and cost status of an on-going residential housing project. The input data needed for these calculations include: 1) number of non-repetitive activities in the project, 2) number of repetitive activities in the project, 3) planned start and finish dates of activities, 4) type of work force utilized (i.e. developers' own force or subcontractors, 5) type of contract awarded to subcontractor, if any (i.e. unit price or lump-sum), 6) total indirect cost of the project per day, 7) report date, 8) actual progress data such as number of activities completed and their actual start and finish dates, number of activities in progress and percentage completed in these activities, and actual construction cost spent up to the report date.

The algorithm is applied in three stages as shown in Figures 4.4, 4.5 and 4.6. The purpose of the first stage is to determine the planned cumulative work and cost for each workday according to the developed base-line schedule. In order to achieve this, the first stage of the algorithm performs the following 11 steps which are designed to:

1.1) Initialize the planned progress and cost arrays with zero values which will be replaced by the calculated planned cumulative work and cost in the subsequent steps for each work day (d=1 to PD), as follows:

$$Plan\_Work[PD] = 0.0 \tag{4.11}$$

Plan_Cost [PD] = 0.0 (4.12)

Where,

PD :– planned project duration,

Plan_Work :– planned cumulative work percentage array,

Plan_Cost :– planned cumulative cost array

1.2) Determine the total cumulative duration of both repetitive and non-repetitive activities in loop 1 as follows:

Cum_Dur = Cum_Dur + Plan_Dur[no_act] (4.13)

Where,

Cum_Dur :– total cumulative duration of all activities,

Plan_Dur[no_act] :– Duration of a particular activity.

1.3) Execute loop 2 to determine the planned cumulative work and cost for each workday starting with d=1 to PD. In this step, the algorithm starts by initializing the variables representing the planned cumulative work and cost for workday d to zero at the beginning of loop 2 as follows:

Plan_Work[d] = 0.0

Plan_Cost[d] = 0.0

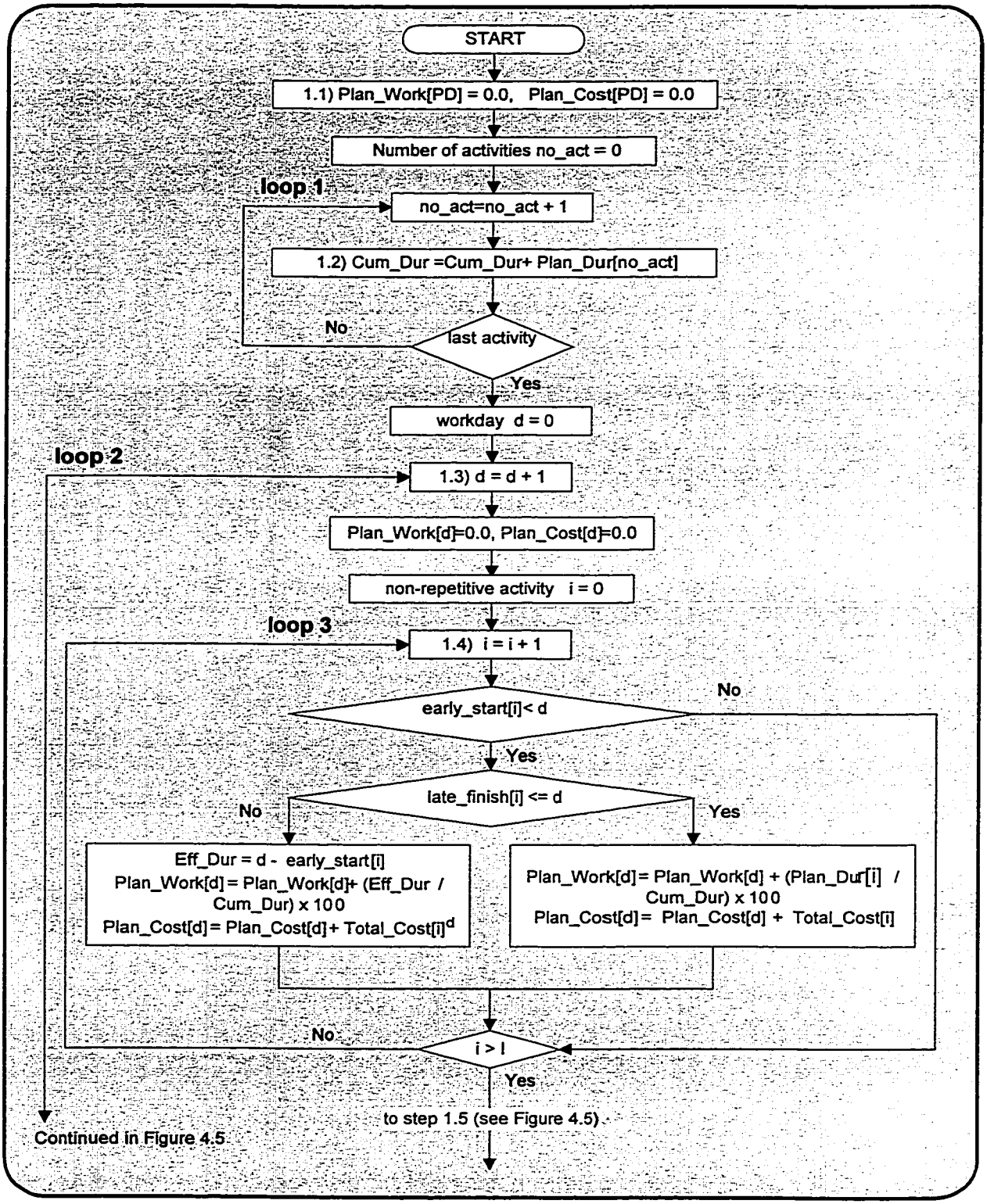


Figure 4.4 Project Control Algorithm (Stage 1a)

1.4) Determine the progress status of non-repetitive activities first. In this step, the algorithm starts by examining whether non-repetitive activity i has started prior to day d. If this condition is false, the next non-repetitive activity (i=i+1) is considered. If this condition is true, the algorithm determines whether the activity i is planned to be finished prior to day d. If this condition is true, planned cumulative work and cost for non-repetitive activity i is calculated as follows:

$$Plan\_Work[d] = Plan\_Work[d] + (Plan\_Dur[i] / Cum\_Dur) \times 100 \quad (4.14)$$

$$Plan\_Cost[d] = Plan\_Cost[d] + (Total\_Cost[i]) \quad (4.15)$$

Where,

Plan_Dur[i] :– planned duration of non-repetitive activity i,

Total_Cost[i] :– total direct cost of non-repetitive activity i.

If activity i has started but not finished by day d, the planned cumulative work and cost are calculated by determining the percentage of work planned to be completed by day d as follows:

$$Eff\_Dur = d - early\_start[i] \quad (4.16)$$

$$Plan\_Work[d] = Plan\_Work[d] + ((Eff\_Dur / Cum\_Dur) \times 100) \quad (4.17)$$

$$Plan\_Cost[d] = Plan\_Cost[d] + (Total\_Cost[i]^{d}) \quad (4.18)$$

Where,

Eff_Dur :– time between the planned early start of activity i and day d,

$Total\_Cost[i]^{d}$:– total direct cost of non-repetitive activity i till day d.

It should be noted that the total direct cost of a non-repetitive activity (i), which has started but not finished prior to day d ($Total\_Cost[i]^d$) is calculated according to the time interval between the planned early start of the activity i and day d (Eff_Dur), and the planned quantity of work to be finished during this time interval. Repeat Step 1.4 for all non-repetitive activities (i=i+1 to I, where I is the number of non-repetitive activities in the project) in nested loop 3 to determine the planned cumulative work and cost for day d.

1.5) Consider repetitive activities after completing the analysis of all non-repetitive activities in loop 4 as shown in Figure 4.5. For each repetitive activity m, determine whether the activity is planned to be started by day d. If this condition is false, next repetitive activity m=m+1 is considered. Otherwise, determine whether it is planned to be finished in all housing units (j = 1 to J, where, J is the total number of housing units) prior to day d. If this condition is true, the planned cumulative work is calculated as follows:

$$Plan\_Work[d] = Plan\_Work[d] + (Plan\_Dur[m] / Cum\_Dur) \times 100 \quad (4.19)$$

Where,

Plan_Dur[m] :– planned duration of repetitive activity m.

1.6) Calculate the planned cumulative cost, which varies depending upon whether the activity is performed by own force or subcontractors as shown in Figure 4.5. If activity m is performed by subcontractors move to step 1.7. If it is planned to be performed by own force, the planned total cost is calculated

by summing up the material, labor and equipment costs in all housing units (j) as shown in loop 5 as follows:

$$Total\_Cost[m] = Total\_Cost[m] + (MC[j]+LC[j]+EC[j]) \quad (4.20)$$

Where,

Total_Cost[m] :– total direct cost of repetitive activity m,

MC[j] :– material cost of housing unit j,

LC[j] :– labor cost of housing unit j,

EC[j] :- equipment cost of housing unit j.

1.7) Determine the planned cost of activity m in the case of employing a subcontractor by first examining the type of contract awarded. Most of the contracts awarded to subcontractors in a residential housing project are either lump sum or unit price. The proposed algorithm considers both of these contract types, embracing flexibility and practicality. The algorithm examines whether the contract type is unit price or lump sum. If it is lump sum move to step 1.8. If the contract is unit price, the planned total cost of activity m is calculated in loop 6 as follows:

$$Total\_Qty = Total\_Qty + Qty[j] \quad (4.21)$$

$$Total\_Cost_m = Total\_Qty \times unit\_price[m] \quad (4.22)$$

Where,

Total_Qty :– total quantity of work to be performed in activity m,

Qty[j] :– quantity of work in each repetitive unit j,

unit_price[m] :– user-specified unit price for repetitive activity m.

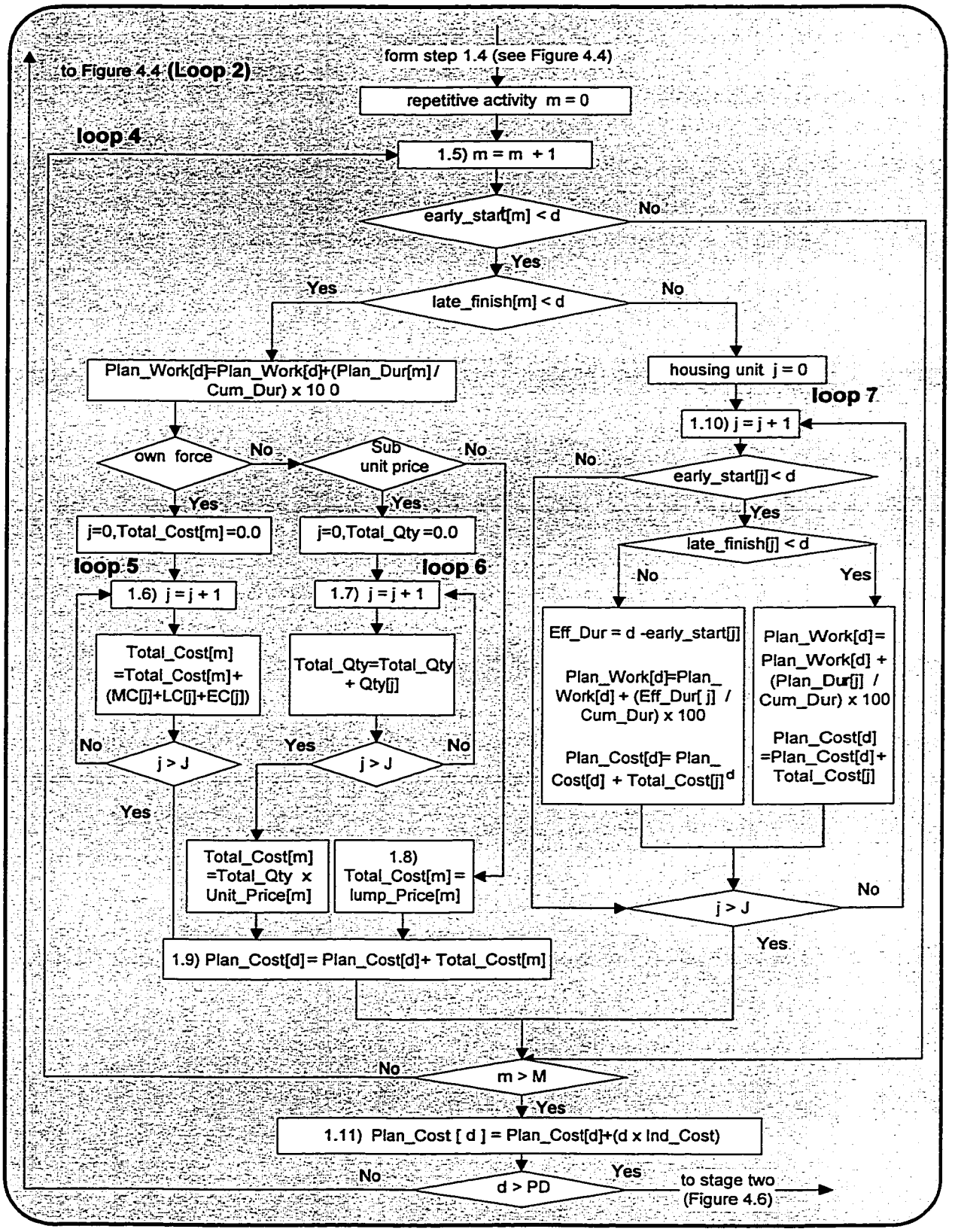


Figure 4.5 Project Control Algorithm (Stage 1b)

1.8) Assign the user-specified lump sum contract amount awarded to the repetitive activity m, in case of a lump sum contract as follows:

$$Total\_Cost[m] = lump\_price[m] \quad (4.23)$$

Where,

lump_price[m] :– user-specified lump-sum price for activity m.

1.9) Calculate the planned cumulative cost of all repetitive activities up to day d by adding the total cost for each repetitive activity (m=1 to M) as follows:

$$Plan\_cost[d] = Plan\_Cost[d] + Total\_Cost[m] \quad (4.24)$$

1.10) Determine the work progress and cost of repetitive activities that have stated but not completed prior to day d. The algorithm identifies repetitive housing units which are finished and in progress and accordingly calculate the planned cumulative work and cost in loop 7. The sequence of operation is similar to that of non-repetitive activity (step 1.4) except that each housing unit (j) is considered in place of a non-repetitive (i) activity as shown in Figure 4.5. Repeat steps 1.5 to 1.10 in order to consider all repetitive activities (m=m+1 to M, where M is the total number of repetitive activities in the project) in loop 4.

1.11) Modify the planned cumulative cost (Plan_Cost[d]) for day d to consider the project indirect cost, as follows:

$$Plan\_Cost[d] = Plan\_Cost[d] + (d \times Ind\_Cost) \quad (4.25)$$

Where,

Ind_Cost :– User-specified average indirect cost per day.

Repeat Steps 1.3 to 1.11 for all workdays (d =d+1 to PD) in loop 2, and calculate the corresponding planned cumulative work and cost and move to the second stage.

The purpose of the second stage is to calculate the cumulative actual work performed (ACW[r]) at the project level up to report date (r) according to the user input as shown in Figure 4.6. In order to achieve this, the second stage performs the following 3 steps which are designed to:

2.1) Initialize the variable representing the actual work performed (AWP[r]) to zero, where r is the report date.

2.2) Calculate the cumulative actual work performed for the activities, which are completed prior to report date r, in loop 8 as follows:

$$AWP[r] = AWP[r] + (Act\_Dur[c] / Cum\_Dur) \times 100 \quad (4.26)$$

Where,

Act_Dur[c] :– actual duration of completed activity c.

Repeat step 2.2 for all completed activities (c= c+1 to C, where C is the total number of activities completed by report date r).

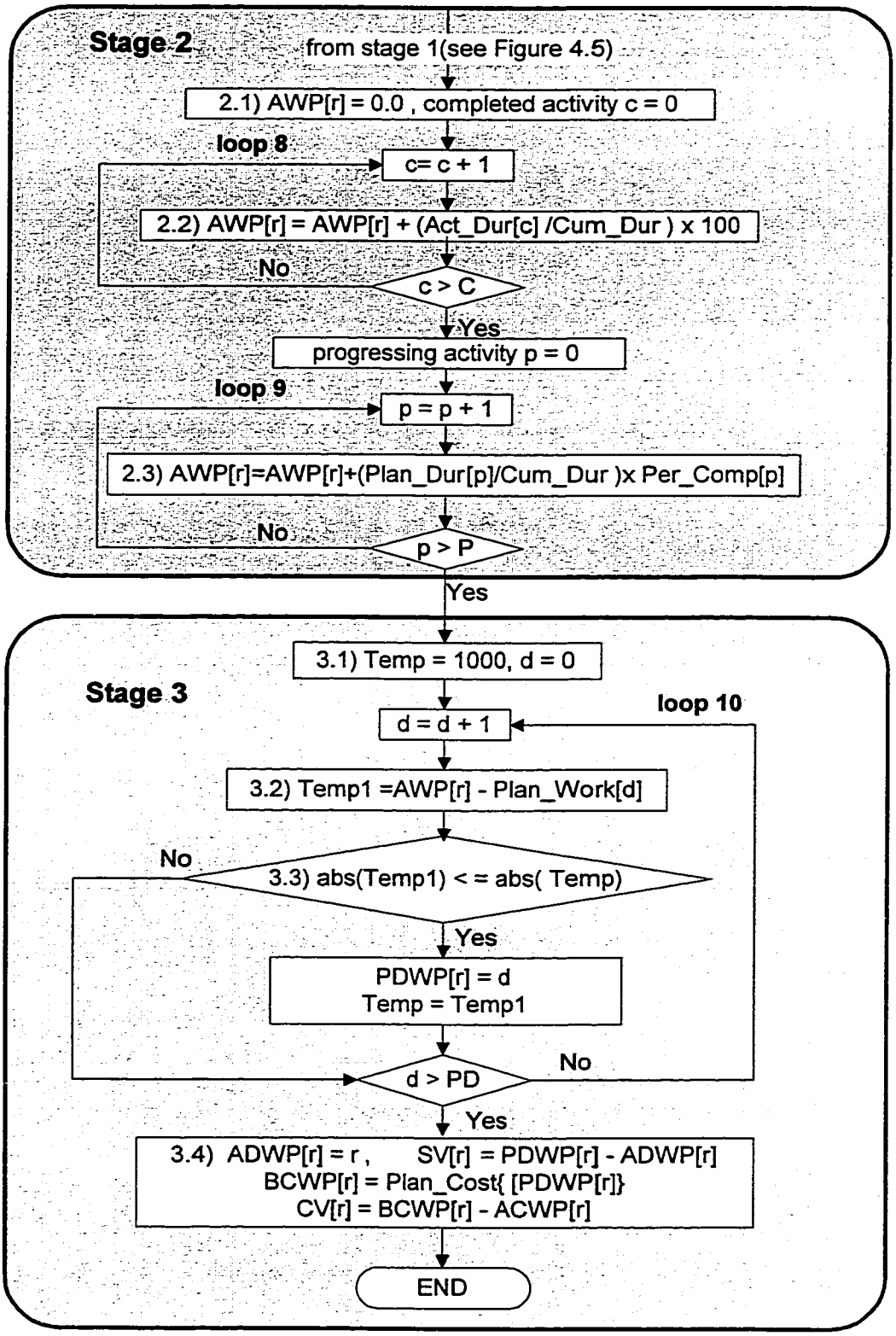


Figure 4.6 Project Control Algorithm (Stage 2 and 3)

2.3) Calculate the cumulative actual work performed for the activities that are in progress on report date r in loop 9 as shown in Figure 4.6, as follows:

$$AWP[r] = AWP[r] + (Plan\_Dur[p] / Cum\_Dur) \times Per\_Comp[p] \quad (4.27)$$

Where,

Plan_Dur[p] :– planned duration of progressing activity p,

Per_Comp[p] :– user-specified percentage of work completed in activity p as of report date r.

Repeat Step 2.3 for all the activities (p=p+1 to P, where P is the total number of activities in progress as of report date r) which are in progress and move to third stage.

The objective of the third stage is to determine the planned duration for work performed (PDWP) and the budgeted cost for work performed (BCWP), in order to calculate the schedule variance (SV) and cost variance (CV). The steps in this stage are shown in Figure 4.6 and are designed to:

3.1) Initialize a temporary variable (Temp) to a large number (e.g. 1000). This large number will be replaced by the least difference between the planned and actual cumulative work in the subsequent calculations.

3.2) Identify the day d that has a planned cumulative work value (Plan_Work[d]) equal or closest to the calculated cumulative actual work performed value (AWP[r]). This is achieved by identifying the difference

between the actual and planned cumulative work values for day d=1 to PD, and assigns the resultant value to another temporary variable (Temp1) as follows:

$$Temp1 = AWP[r] - Plan\_Work[d] \quad (4.28)$$

3.3) Determine whether the absolute value of the second temporary variable (Temp1) calculated in step 3.2 is less than or equal to the first temporary variable (Temp). If this condition is true, the corresponding value of day d is assigned to a variable representing the planned duration for work performed (PDWP[r]), and the value of the second temporary variable (Temp1) is assigned to the first temporary variable (Temp), thereby replacing it with a smaller value.

Repeat steps 3.2 and 3.3 for all workdays (d=d+1 to PD) in loop 10. This process will identify the day d, which represents the planned duration for work performed (PDWP[r]) .

3.4) Determine the schedule variance (SV) and cost variance (CV) for the project on report date r. The schedule variance is calculated as the difference between the planned duration for work performed (PDWP[r]), identified in steps 3.1 to 3.3, and the actual duration for work performed (ADWP[r]) on report date r, as follows:

$$ADWP[r]=r \quad (4.29)$$

$$SV[r] = PDWP[r] - ADWP[r] \quad (4.30)$$

Where,

SV[r] :– schedule variance that represents the status of the project with respect to time on reporting date r.

A schedule variance value of zero represents that the project is as per schedule. Negative and positive values represent the project is behind and ahead of schedule, respectively.

The cost variance (CV) is calculated as the difference between the budgeted cost for work performed (BCWP[r]) and the actual cost of work performed (ACWP[r]), as follows:

$$BCWP[r]=Plan\_Cost\{PDWP[r]\} \quad (4.31)$$

$$CV[r] = BCWP[r] - ACWP[r] \quad (4.32)$$

Where,

CV[r] :– cost variance that represents the status of the project with respect to cost on report date r.

ACWP[r] :– actual cost for work performed based on the user-specified actual cost till report date r.

A cost variance value (CV) of zero represents that the project cost is as per budgeted. Negative and positive values represent the project is above and below budgeted cost, respectively.

4.3.2 Housing Unit Control

The purpose of the unit control algorithm is to determine the time and cost status of a particular housing unit. The algorithm performs the necessary calculations to determine the status of the housing unit as requested by the user. The input needed for these calculations include: 1) number of the housing unit (e.g. house # 10) for which the performance is to be determined, 2) number of activities to be performed in this particular housing unit, 3) planned start and finish dates of all activities, 4) type of work force utilized (i.e. own force or subcontractors), 5) type of contract awarded to the subcontractor (i.e. unit price or lump sum), 6) reporting date and 7) actual progress data.

The algorithm is implemented in three stages. The first stage calculates the planned cumulative work and cost. The second stage determines the actual cumulative work and the third stage identifies the time and cost performance. Unit control algorithm is similar to the one developed for project control explained in the earlier section. However, the data corresponding to a housing unit such as duration, number of activities, start and finish dates of varies activities in a particular unit are used in the calculations instead of the overall project data.

4.3.3 Sub Control

The purpose of the sub control algorithm is to evaluate the time and cost performance of an individual subcontractor. The input needed for this algorithm include: 1) name of the subcontractor for whom the performance is to be

evaluated, 2) number of housing units assigned for the subcontractor, 3) scheduled start and finish dates of the activity in each unit, 4) type of contract awarded, 5) reporting date, 6) actual progress data. These input data are used in the calculations to determine whether the subcontractor is ahead or behind schedule and over or under paid. The algorithm is applied in three stages and is similar to the project control algorithm explained in section 4.3.1. However, the data corresponding to a subcontractor are used in the tracking and control calculations, instead of the overall project data.

4.4 Summary

Two algorithms, one for scheduling subcontractors and the other for tracking and control of residential housing projects have been presented in this chapter. A number of practical factors commonly encountered in these types of projects are considered in the development of both algorithms. The scheduling algorithm for subcontractors is applied in two stages. The first stage complies with precedence relationships and subcontractor availability constraints and the second complies with the crew work continuity constraint. The proposed tracking and control algorithm is implemented in three stages. The first and second stage calculates the planned and actual cumulative work and cost, respectively. The third stage analyzes the results of the previous two stages and evaluates the time and cost performance. The algorithm can be applied at three levels (i.e. project, housing unit or subcontractor) of an on-going project, enabling easy and early detection of construction problems, if any.

CHAPTER 5

IMPLEMENTATION OF THE PROPOSED MODEL: RESIDENTIAL PLANNER

5.1 Introduction

The implementation stage of the proposed scheduling and tracking and control model for residential housing projects is presented in this chapter. This stage is the third and final stage of the model development (Rambaugh et al, 1991). The model is implemented as a Windows application using Visual C++ 6.0 and Microsoft Foundation Class (MFC), and is named 'Residential Planner'. It runs on Windows 2000 and NT and supports user-friendly interface. Residential Planner consists of two main components: 1) Model and 2) Graphical User Interface (GUI) as shown in Figure 5.1. These two parts are grouped together to form a workspace, where all the source files (.h, .cpp) are combined to produce target files (.exe, .dll).

5.2 Model

The proposed scheduling and tracking and control model consists of 18 classes. These classes are implemented using C++ programming language in two types of files: header (.h) and code (.cpp). In C++, declaration and definition of a class are often included in header and code files, respectively. The implementation of the proposed model includes 6 header files (calendar.h, project.h, database.h, control.h, view.h and data.h) and 11 Code files (calendar.cpp, project1.cpp,

project2.cpp, project3.cpp, database.cpp, control.cpp, view.cpp, data1.cpp, data2.cpp, data3.cpp and crew.cpp) as shown in Figure 5.1.

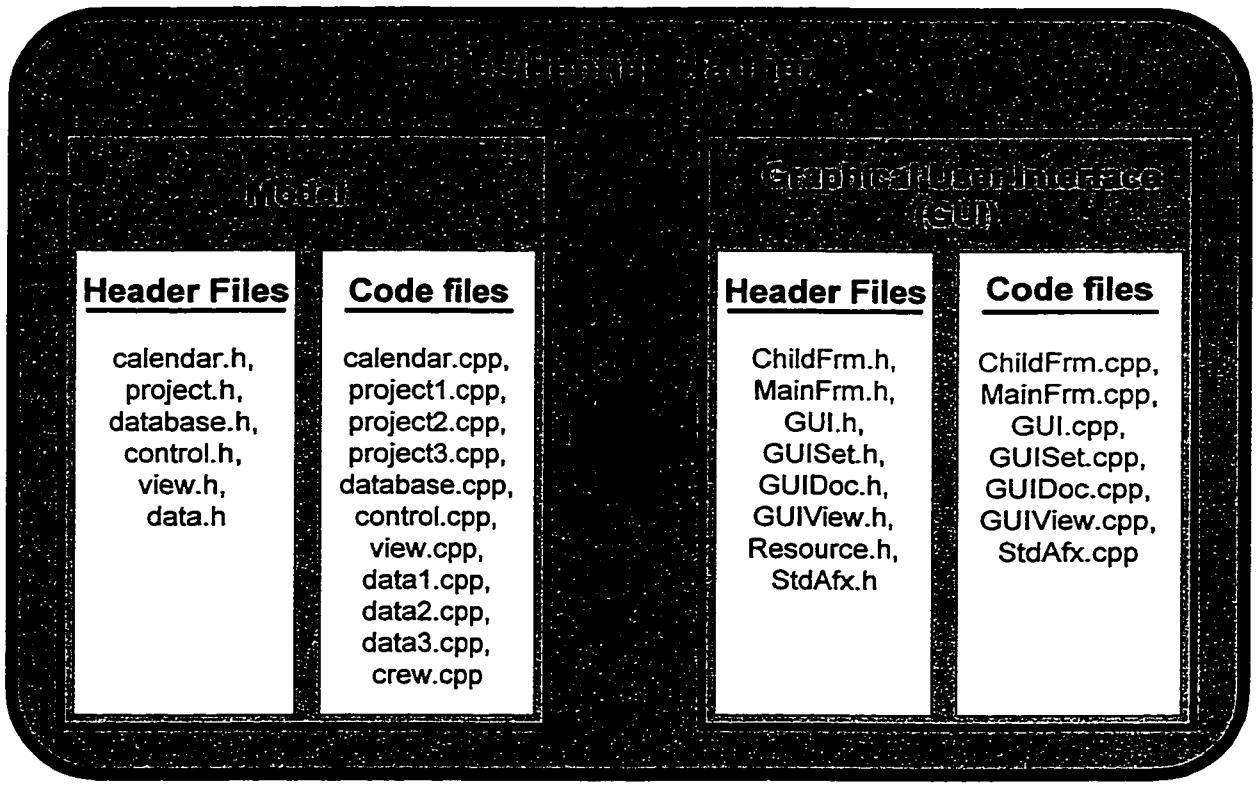


Figure 5.1 Residential Planner

The declaration and definitions of Date class are included in calendar.h and calendar.cpp, respectively. Declaration of Project class is included in project.h and its definitions are included in project1.cpp, project2.cpp and project3.cpp. Declaration and definition of Data-Base class is included in database.h and database.cpp, respectively. Declarations of classes Tracking-Control, Project-Control, Unit-Control and Sub-Control are included in control.h and their corresponding definitions are included in control.cpp. Declaration and definitions of View class are included in view.h and view.cpp, respectively. Declarations of

the remaining 10 classes are included in data.h and their definitions are included in four separate code files (data1.cpp, data2.cpp, data3.cpp and crew.cpp).

5.3 Graphical User Interface (GUI)

Visual C++ uses a tool called AppWizard to create user-friendly interfaces. AppWizard produces a platform with several classes, objects and functions to facilitate building windows application (Gregory, 1999). Graphical User Interface (GUI) is developed using AppWizard and has eight header files (ChildFrm.h, MainFrm.h, GUI.h, GUISet.h, GUIDoc.h, GUIView.h, Resource.h and StdAfx.h) and seven code files (ChildFrm.cpp, MainFrm.cpp, GUI.cpp, GUISet.cpp, GUIDoc.cpp, GUIView.cpp and StdAfx.cpp) as shown in Figure 5.1. GUI is developed as a multiple document interface (MDI) application, which enables the user to have more than one document open at a time. It has menus, toolbar, status bar and dialog boxes which forms an integral part of GUI.

5.3.1 Menus, Toolbar and Status bar

Residential Planner has a menu bar at the top of the screen from which a user can select a specific function as shown in Figure 5.2. It also contains a toolbar below the menu bar and a status bar at the bottom of the screen. The menu bar consists of twelve menus: File, Project, Activity, Relation, Record, View, Tracking & Control, Text Report, Graphical Report, Edit, Window and Help. AppWizard is used to develop File, View, Edit, Window and Help menu items with the platform and they perform similar functions to those commonly found in most Windows

applications. However, the rest of the menu items are designed to perform specific functions as shown in Figures 5.3 to 5.9. Tables 5.1 to 5.7 summarize these menu items, their functions and associated dialog boxes, if any.

Figure 5.2 Menu Bar

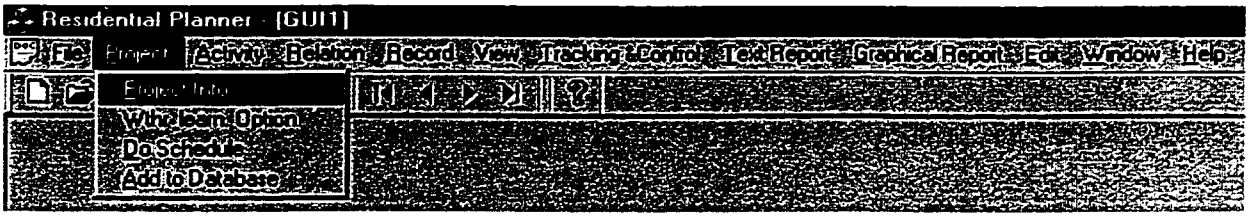


Figure 5.3 Project Menu

Table 5.1 Project Menu Functions

Menu item	Associated dialog box	Function
Project Info.	Figure 5.10	Accepts general project information.
Wthr_learn. Option	Figure 5.11	Specifies whether to consider the impact of weather and/or learning curve in scheduling calculations or not.
Do Schedule	—	Starts scheduling calculations.
Add to Database	Figure 5.12	Allows adding new information to database.

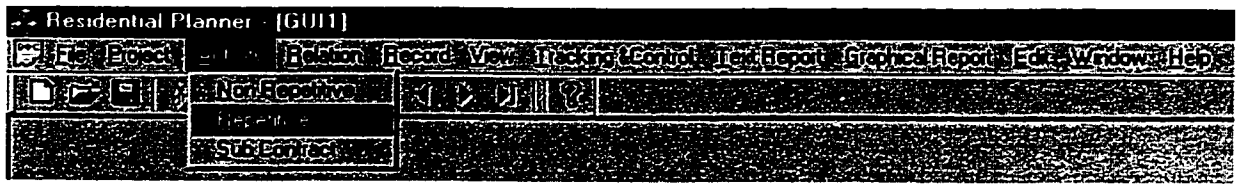


Figure 5.4 Activity Menu

Table 5.2 Activity Menu Functions

Menu item	Associated dialog box	Function
Non Repetitive	Figure 5.13	Accepts data of a non-repetitive activity.
Repetitive	Figure 5.14	Allows repetitive activity data input.
Sub Contract	Figure 5.15	Allows data input for a subcontractor.

Figure 5.5 Relation Menu

Table 5.3 Relation Menu Functions

Menu item	Associated dialog box	Function
Input	Figure 5.20 and 5.21	Allows new precedence relationship input to the project. Also specifies the type, lag, predecessor and successor activities.

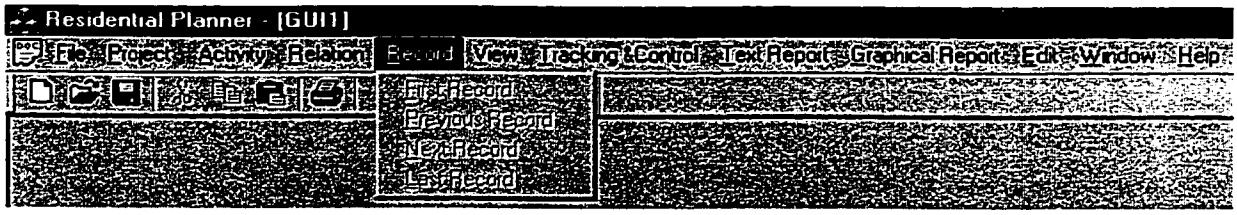


Figure 5.6 Record Menu

Table 5.4 Record Menu Functions

Menu item	Associated dialog box	Function
First Record	----	Displays the first record in a corresponding database table.
Previous Record	----	Displays the previous record.
Next Record	----	Displays the next record in the database.
Last record	----	Displays the last available record in the database table.

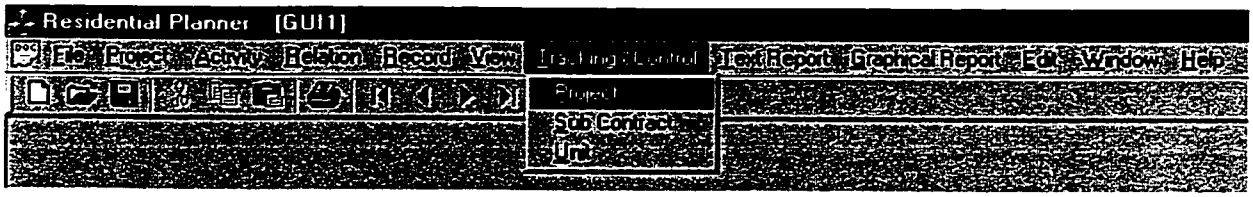


Figure 5.7 Tracking & Control Menu

Table 5.5 Tracking & Control Menu Functions

Menu item	Associated dialog box	Function
Project	Figure 5.22	Accepts actual progress data at the project level.
Sub Contract	Figure 5.25	Allows input of actual progress data of a subcontractor.
Unit	Figure 5.26	Accepts actual progress data of a specific unit.

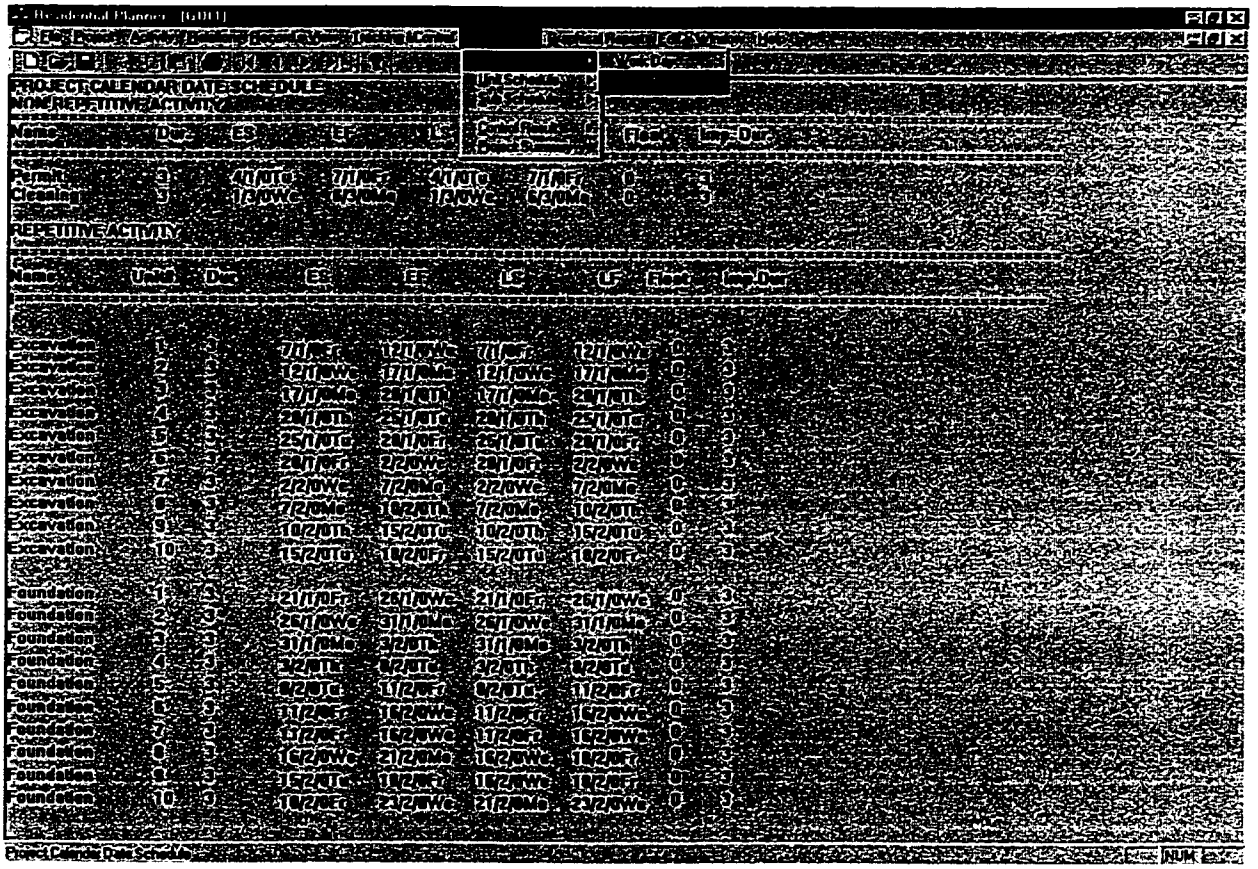


Figure 5.8 Text Report Menu

Table 5.6 Text Report Menu Functions

Menu item	Associated dialog box	Function
Project Schedule	----	Displays the project schedule in workday or calendar date.
Unit Schedule	----	Displays the developed schedule for a particular housing unit in workday or calendar date.
Sub Schedule	----	Displays the schedule for a subcontractor in workday or calendar date.
Control Result	----	Displays the status of a project, subcontractor or a housing unit.
Project Summary	----	Displays the summary of the project.

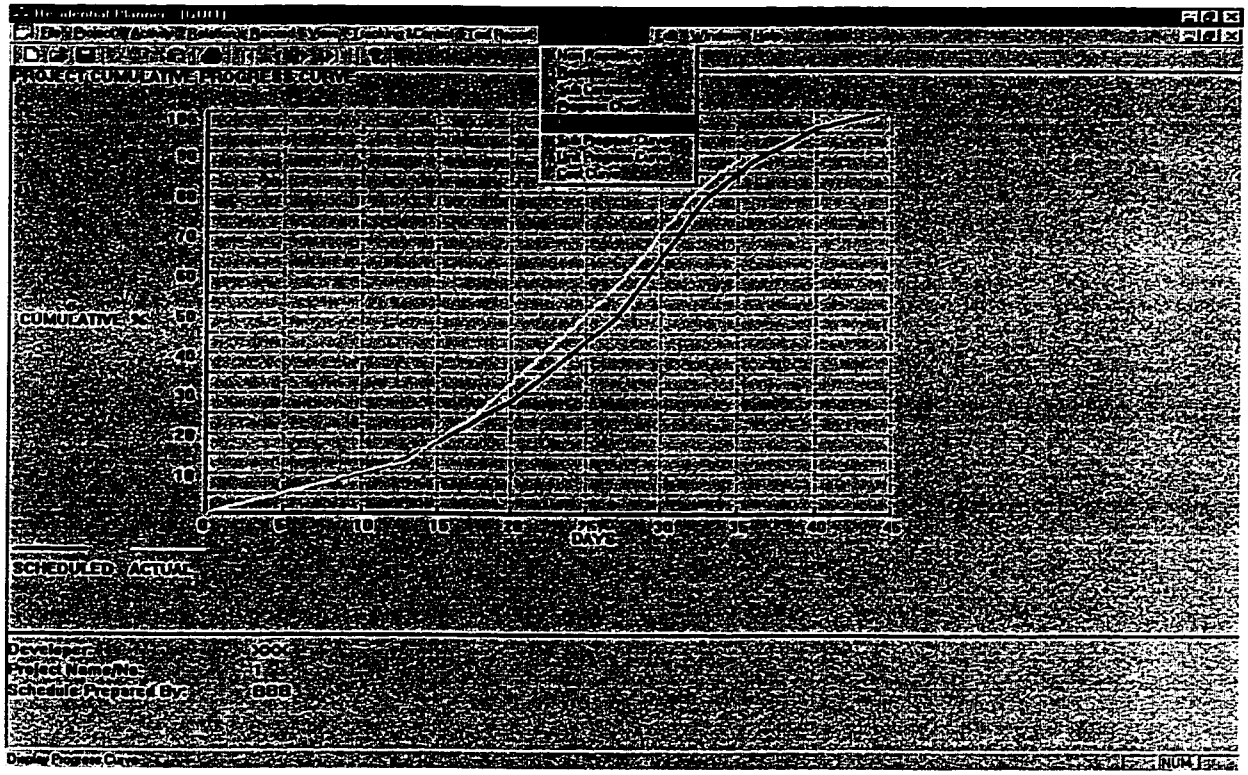


Figure 5.9 Graphical Report menu

Table 5.7 Graphical Report Menu Functions

Menu item	Associated dialog box	Function
Non Repetitive	----	Displays the schedule for all non-repetitive activities in a graphical format.
Repetitive	----	Displays the schedule for all repetitive activities in a graphical format.
Sub Contractor	----	Displays the graphical schedule for a sub contractor.
Progress Chart	----	Displays the planned and actual progress charts.
Progress Curve	----	Displays the planned and actual cumulative work curves for the project.
Sub Progress Curve	----	Displays the planned and actual cumulative work curves for a subcontractor.
Unit Progress Curve	----	Displays the planned and actual cumulative work curves for a particular unit.
Cost Curve	----	Displays the planned cumulative cost curve for the project.

5.3.2 Dialog Boxes

Residential planner incorporates a number of dialog boxes to facilitate user-input. Several important dialog boxes of residential planner are shown in Figures 5.10 to 5.26.

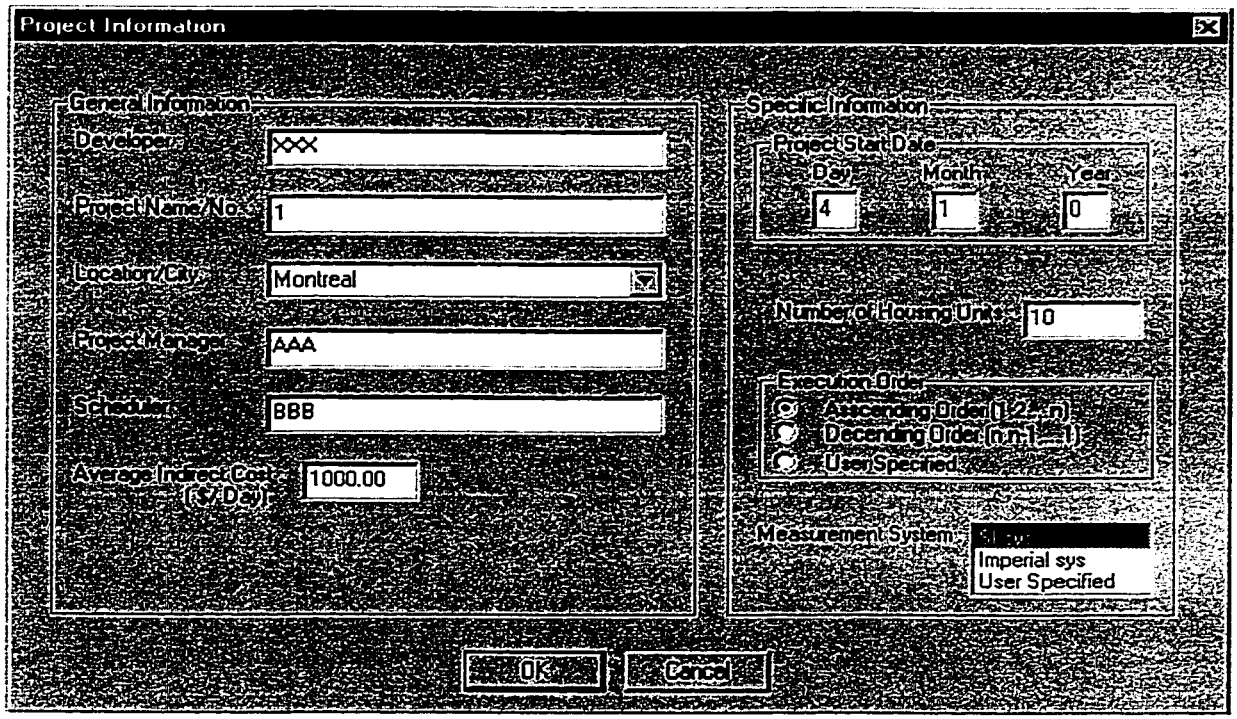


Figure 5.10 Project Information Dialog Box

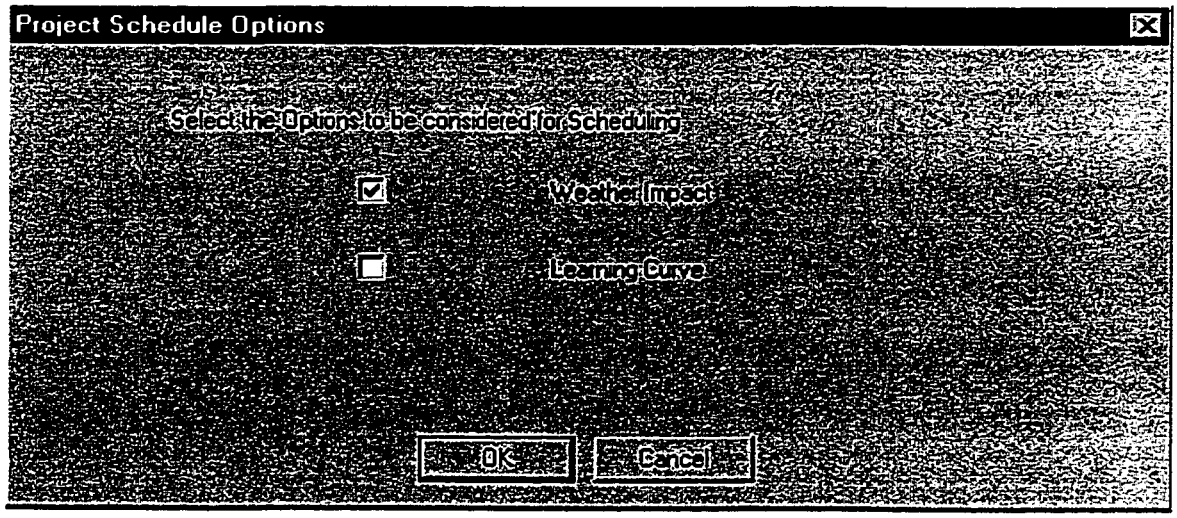


Figure 5.11 Weather and Learning Curve Dialog Box

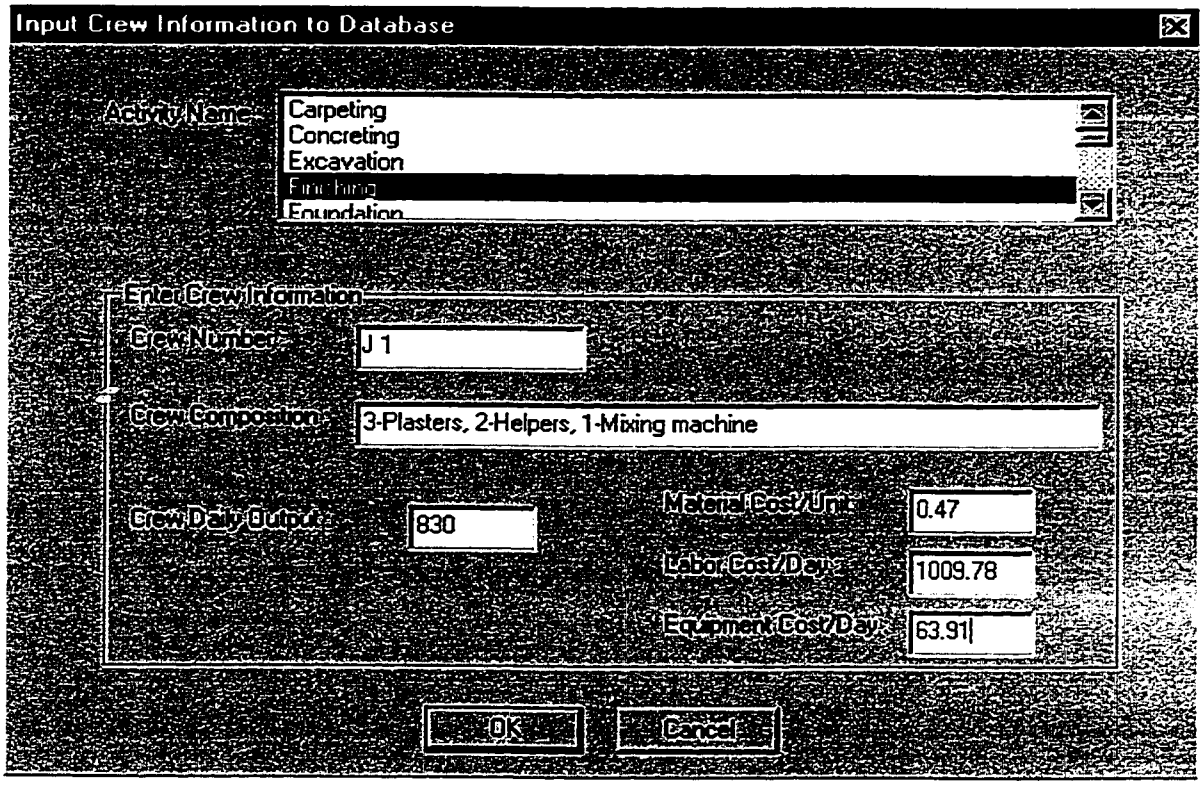


Figure 5.12 Inserting New Information to Database Dialog Box

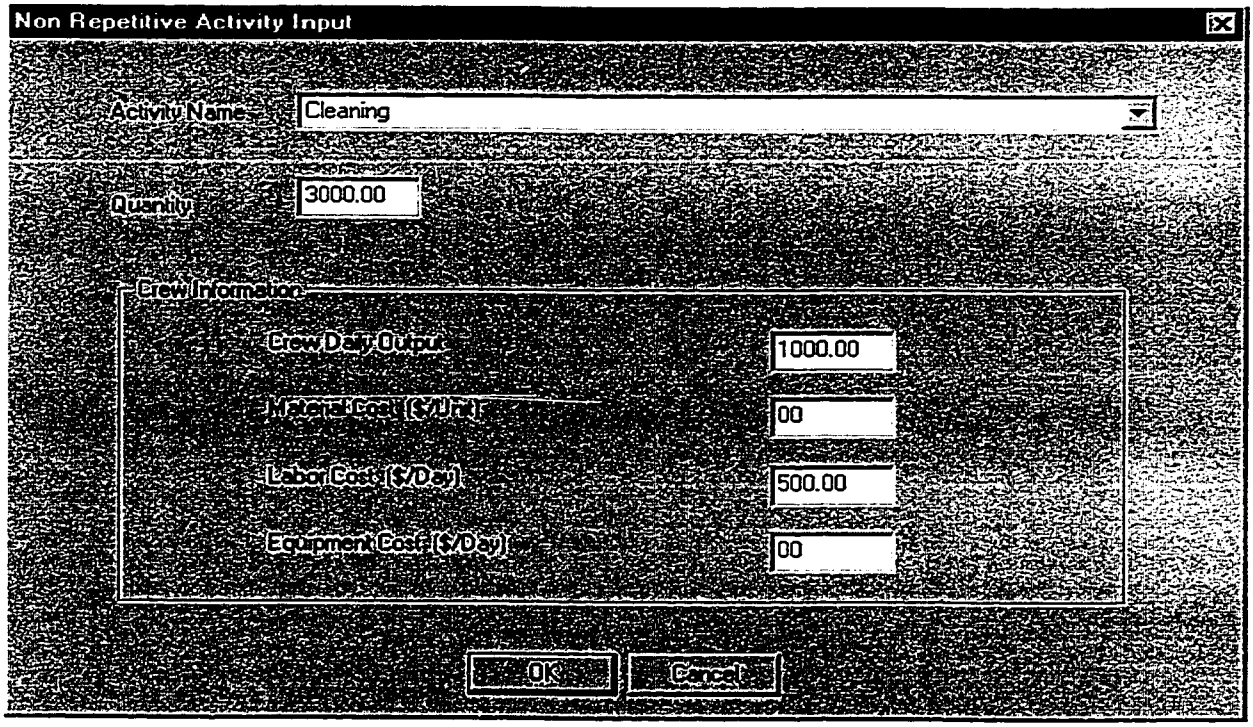


Figure 5.13 Non-repetitive Activity Input Dialog Box

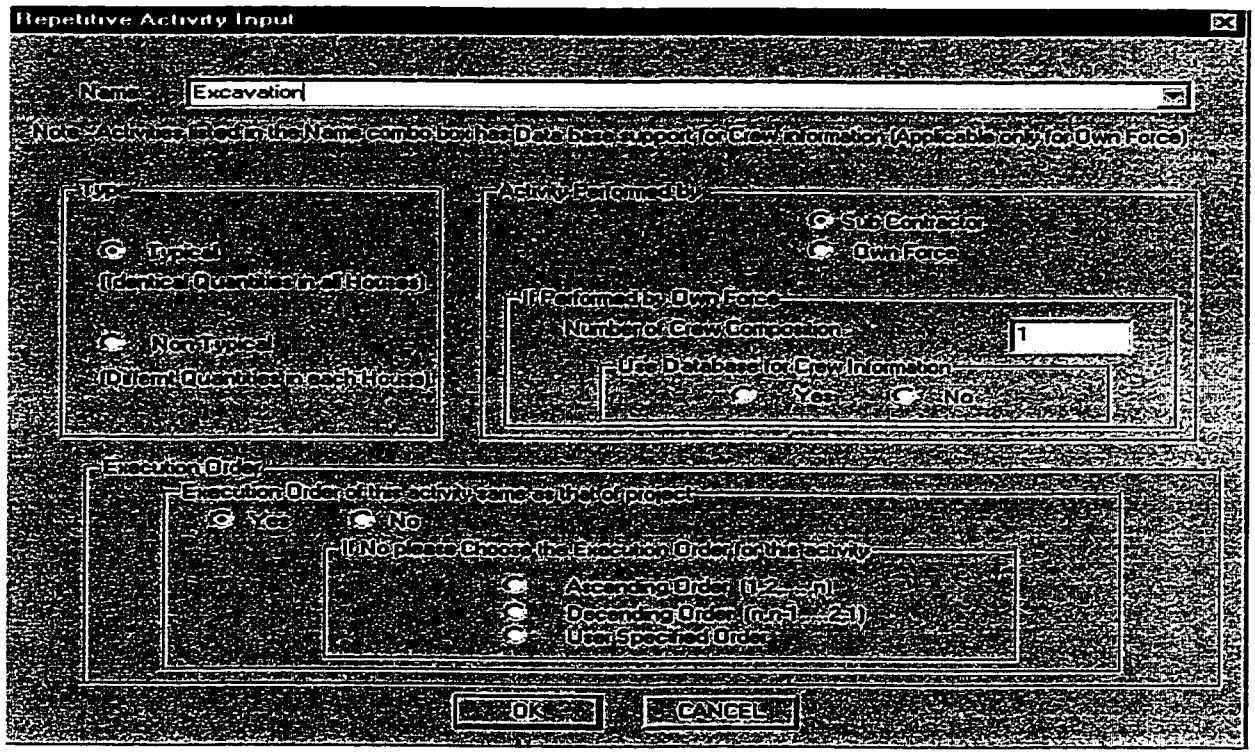


Figure 5.14 Repetitive Activity Input Dialog Box (Subcontractor)

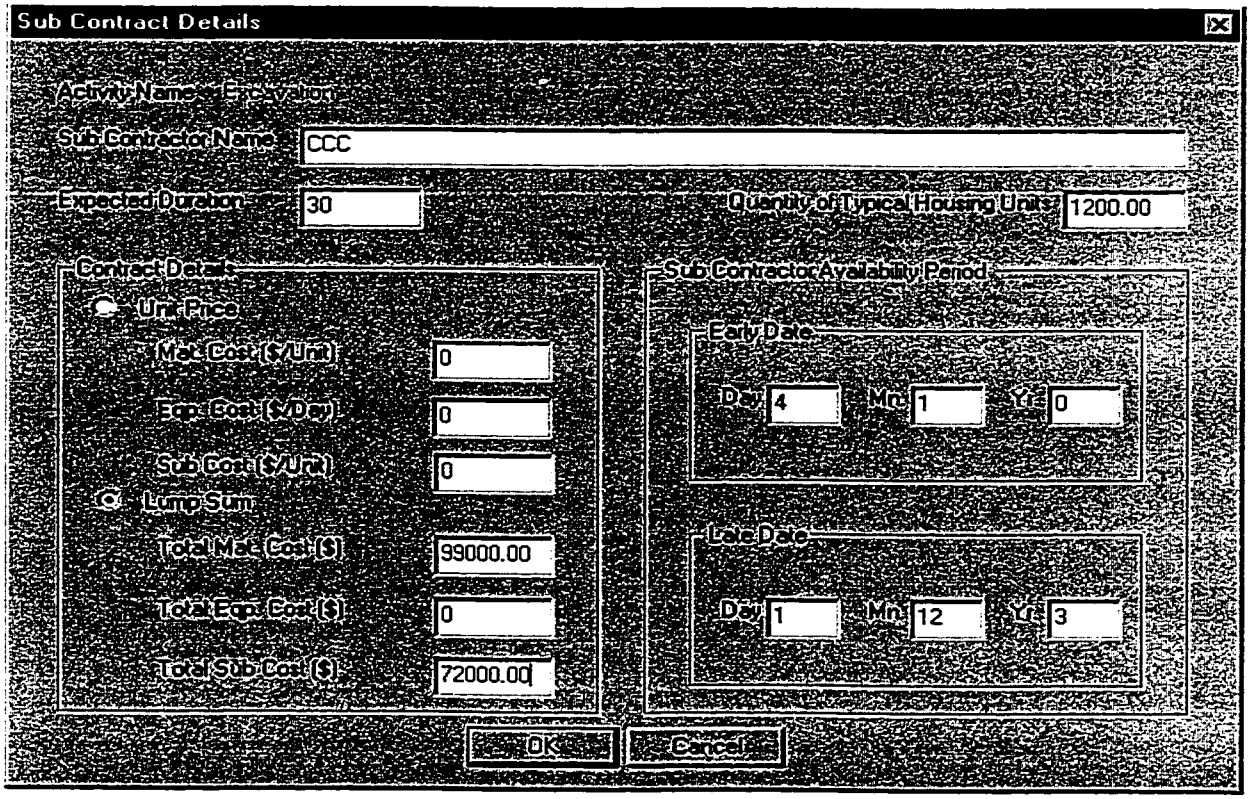


Figure 5.15 Subcontractor Input Dialog Box

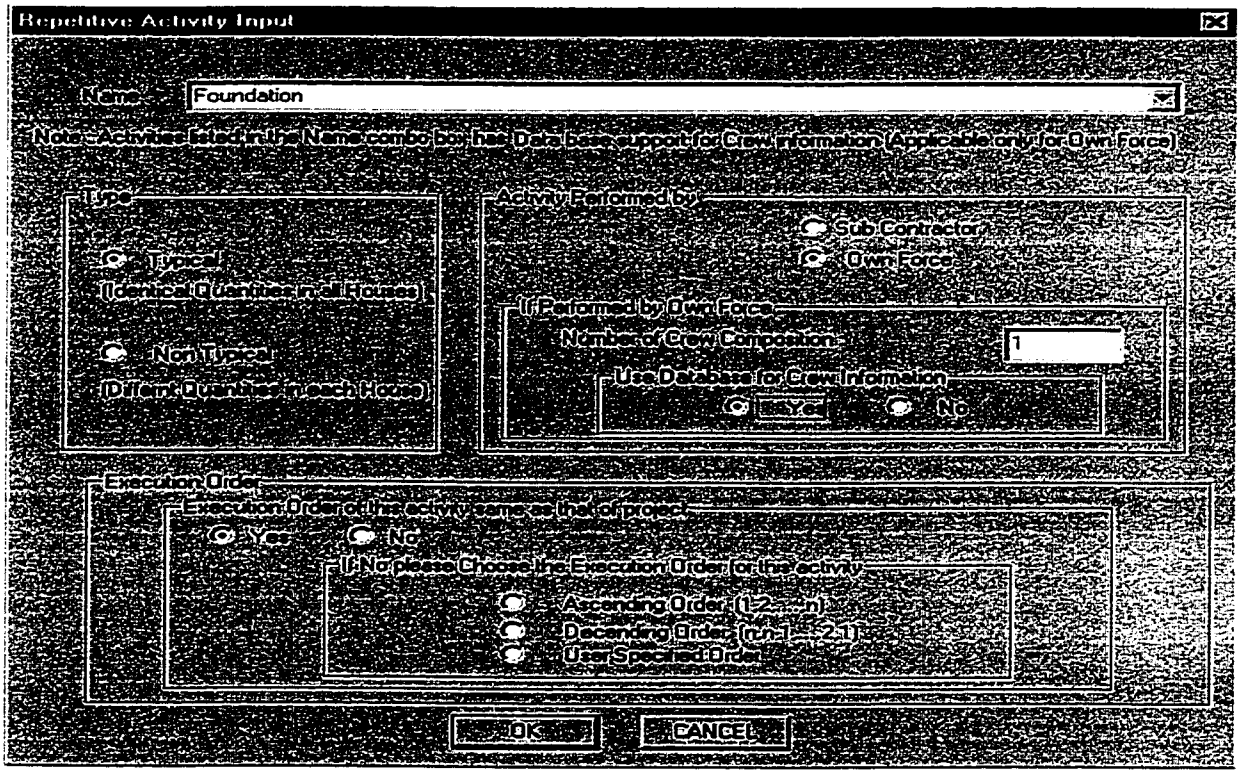


Figure 5.16 Repetitive Activity Input Dialog Box (Own work force)

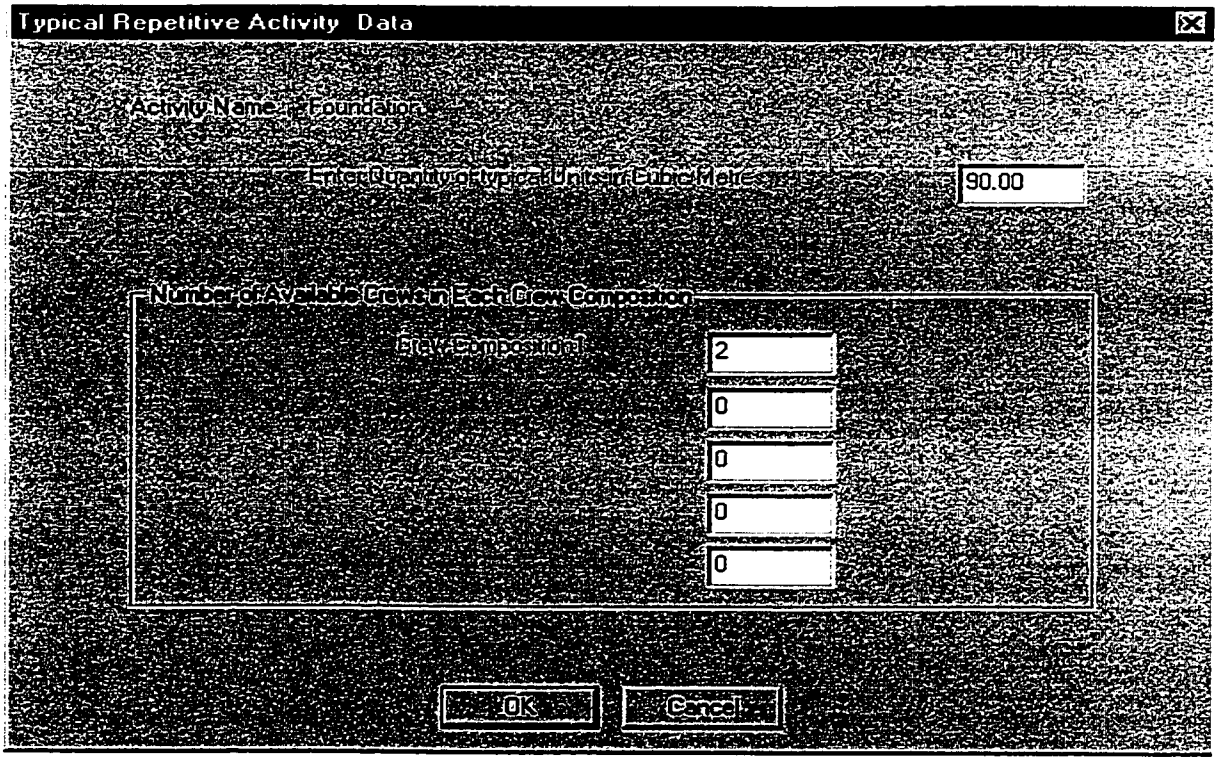


Figure 5.17 Typical Repetitive Activity Input Dialog Box

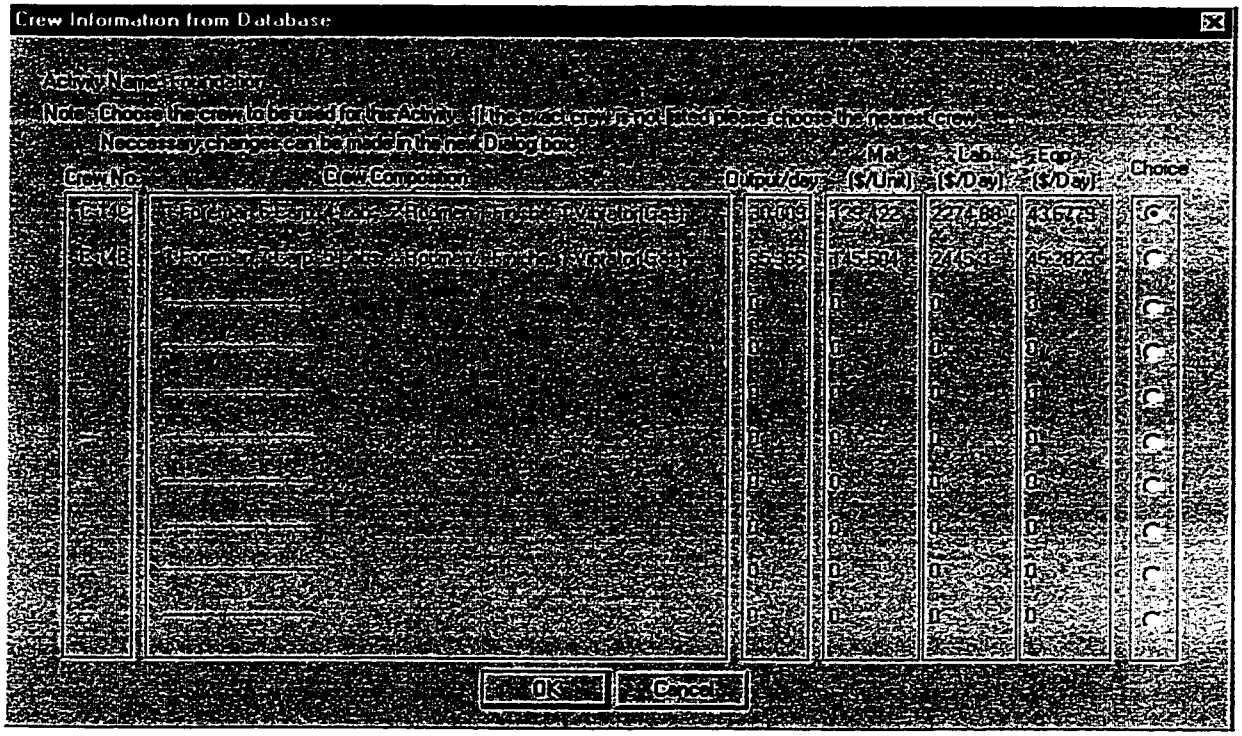


Figure 5.18 Crew Information from Database Dialog box

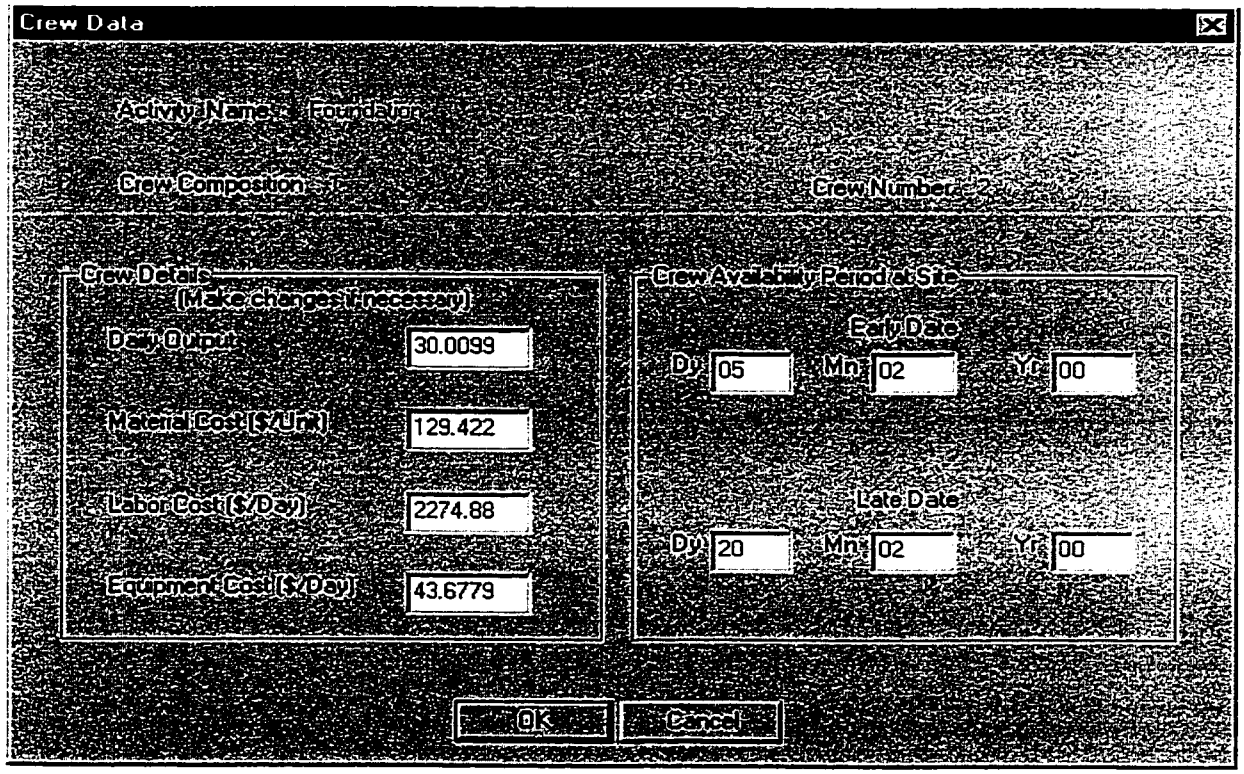


Figure 5.19 Crew Data Dialog Box

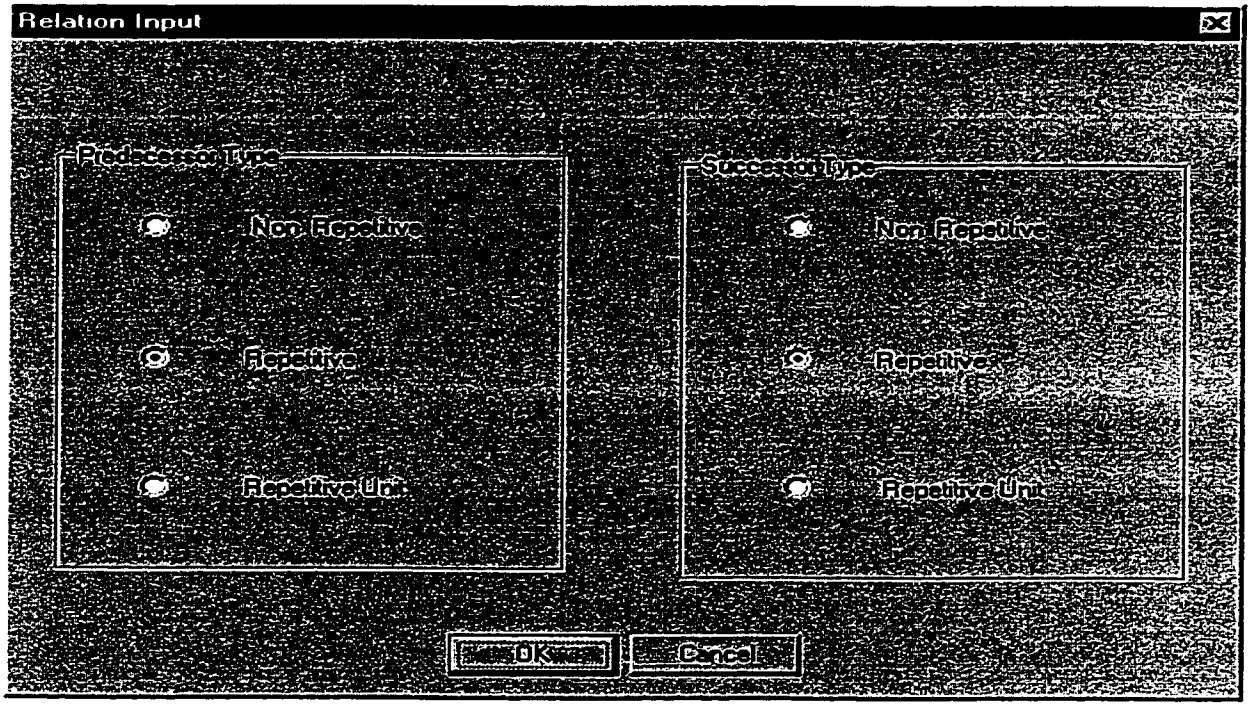


Figure 5.20 Relation Type Input Dialog Box

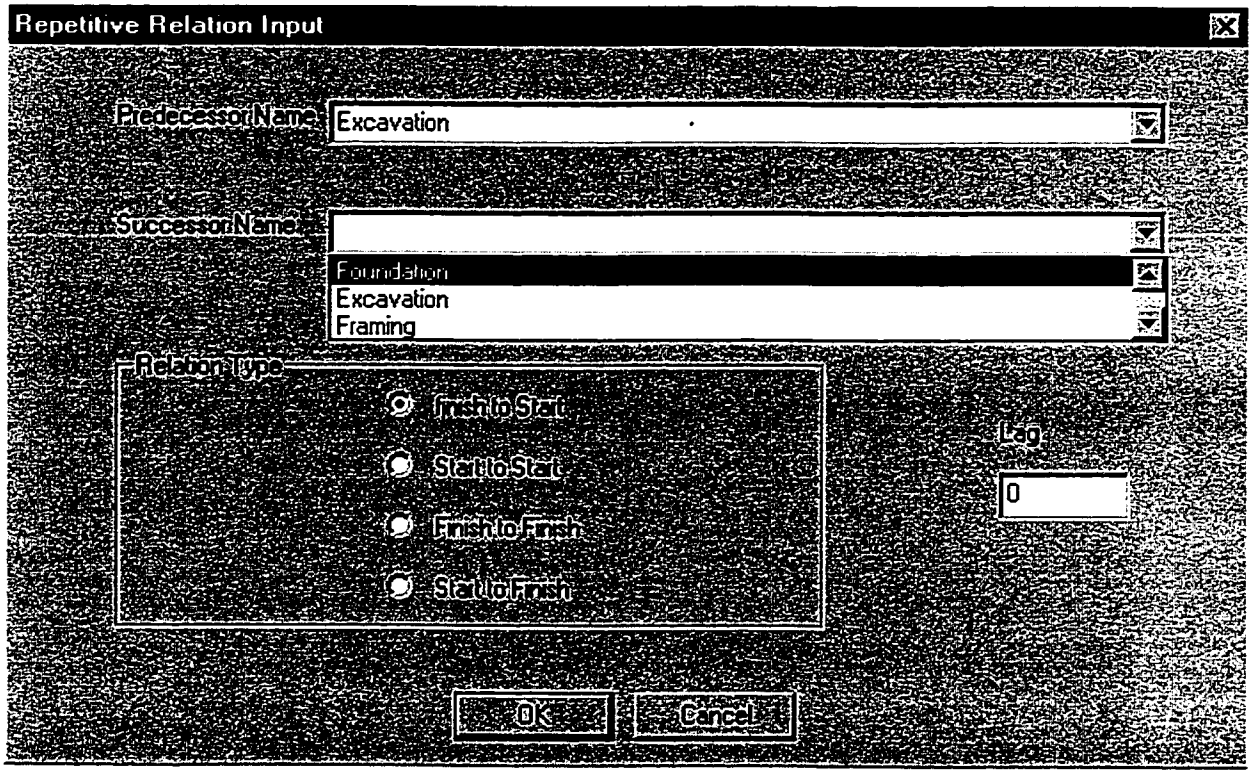


Figure 5.21 Repetitive Relation Input Dialog Box

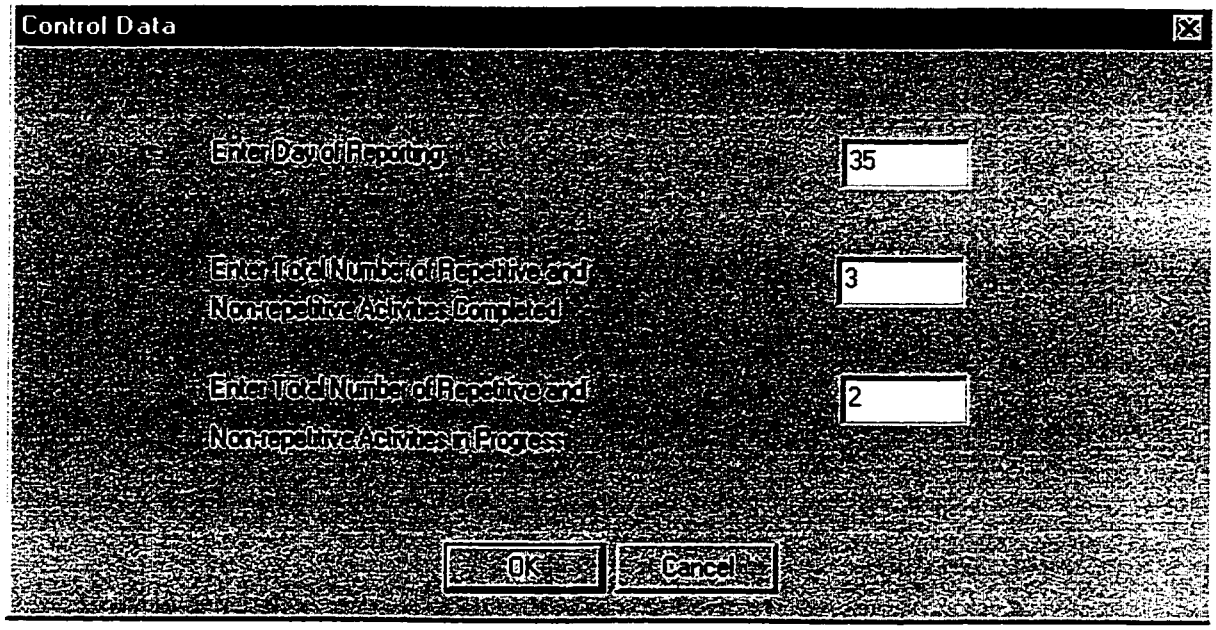


Figure 5.22 Project Control Dialog Box

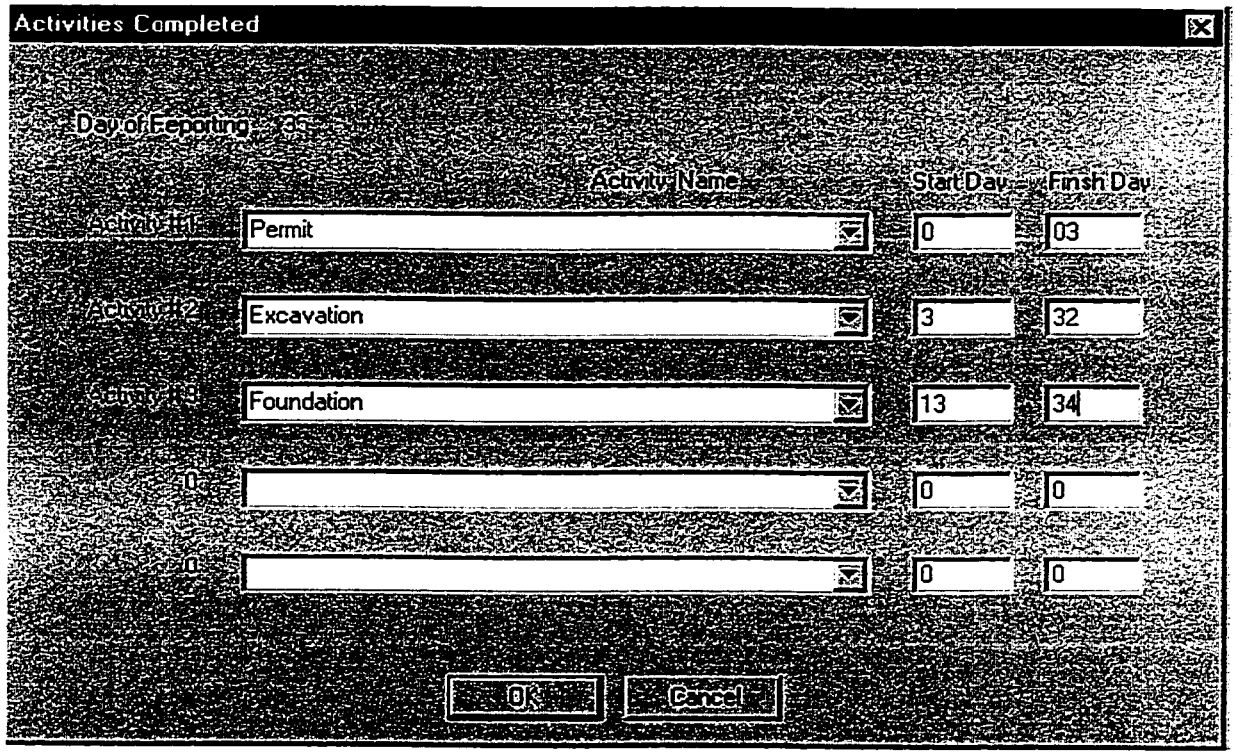


Figure 5.23 Progress Input Dialog Box (Activities completed)

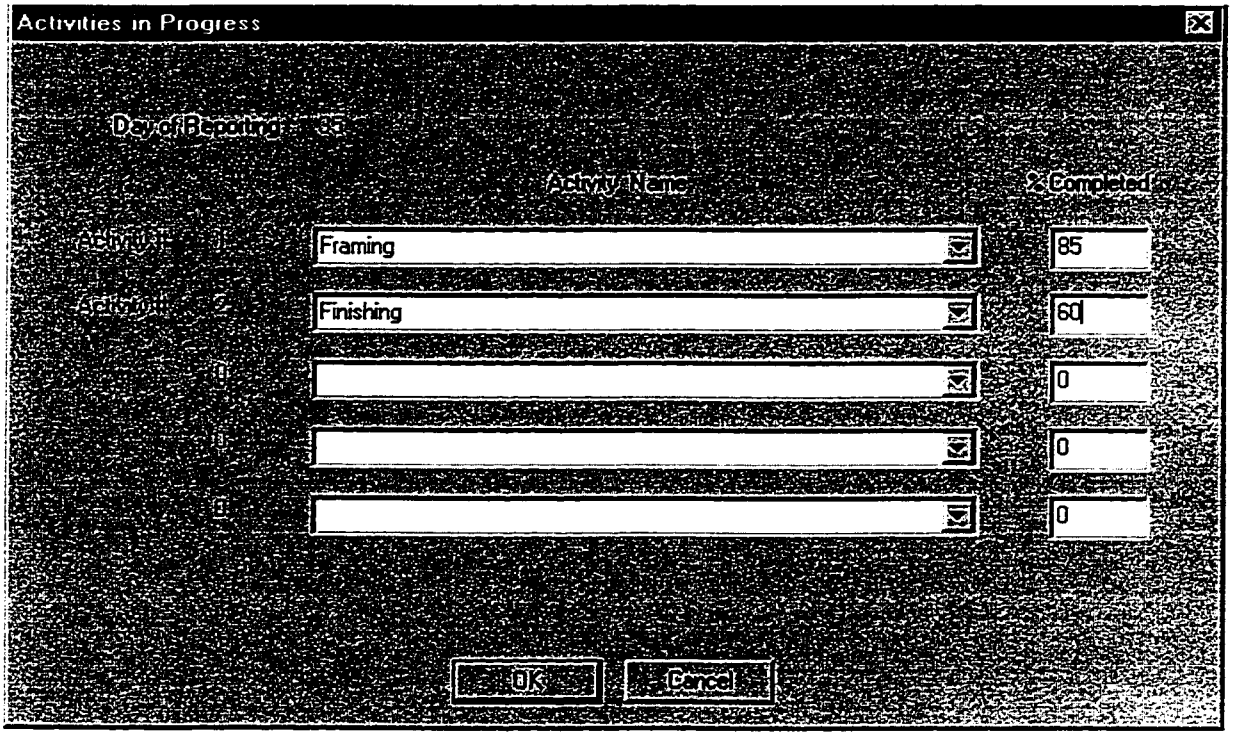


Figure 5.24 Progress Input Dialog Box (Activities in Progress)

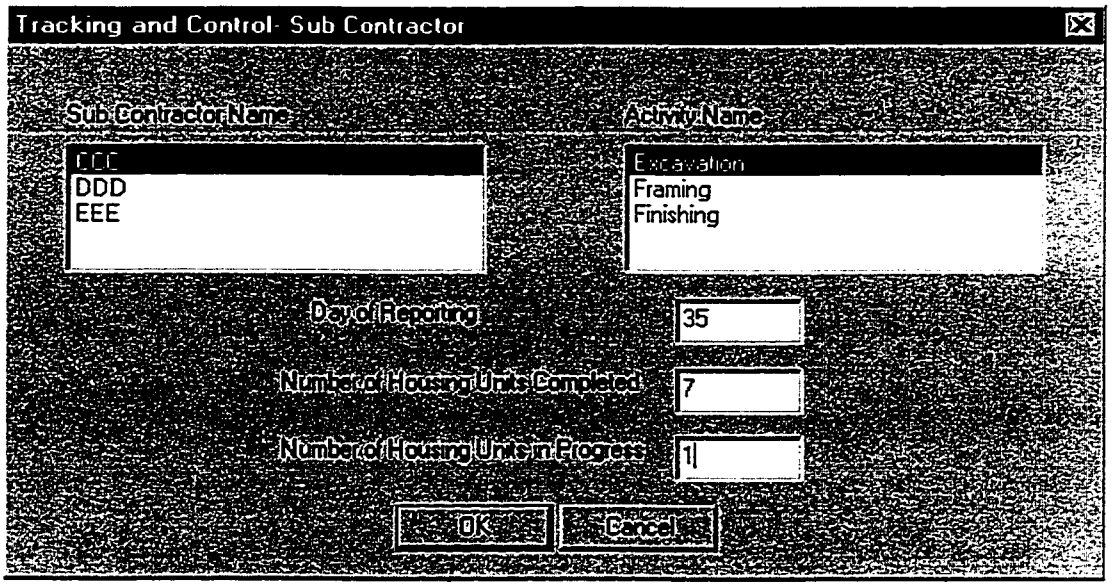


Figure 5.25 Subcontractor Control Dialog Box

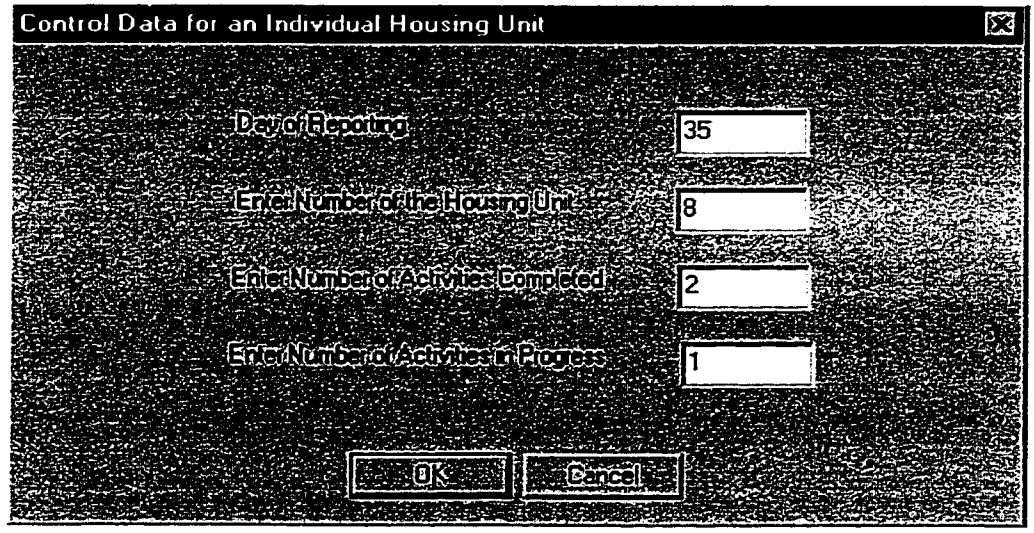


Figure 5.26 Unit control dialog box

5.4 Input and output

Residential Planner is capable of scheduling and tracking and controlling of residential housing projects and requires specific input data to perform these calculations as shown in Figure 5.27. To perform project scheduling, five levels of data are required: 1) project, 2) non-repetitive, 3) repetitive, 4) subcontractor, and 5) relation. Similarly, for tracking and control calculations, different set of input data have to be provided depending on the user request to determine the status of the project as a whole, a particular housing unit or an individual subcontractor. Residential Planner can generate text reports at three distinct levels namely project, housing unit and subcontractor. These reports are supported with a variety of graphical reports in order to address the diverse needs of the developer and to assist in managing subcontractors and administering all contracts effectively.

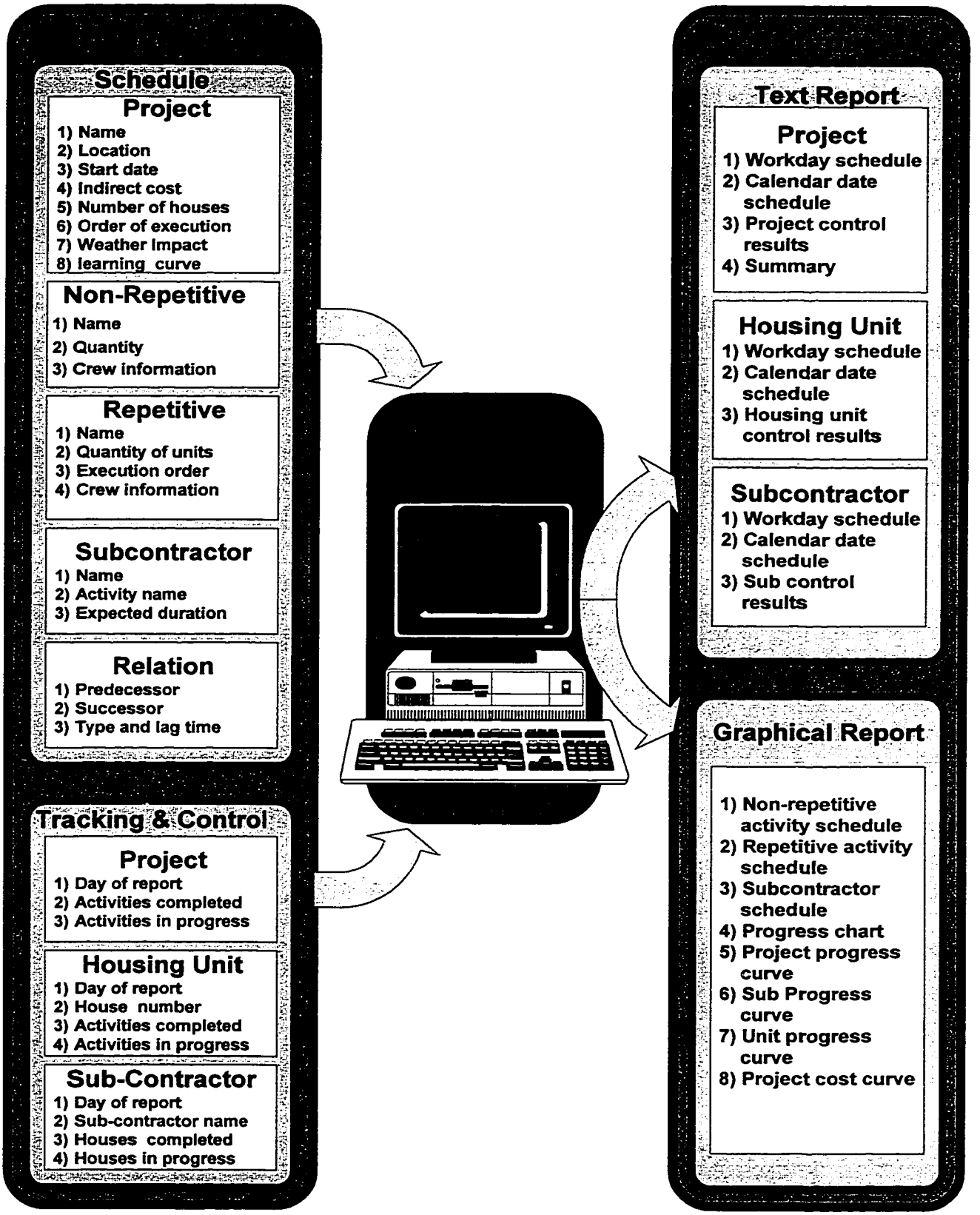


Figure 5.27 Residential Planner Input and Output

5.5 Summary

The implementation stage of the proposed Object-Oriented model for scheduling and tracking and control of residential housing projects has been presented in this chapter. The model is implemented as a Windows application using Visual C++ 6.0 and Microsoft Foundation Class (MFC), and is named 'Residential Planner'. Residential Planner has two major components namely, Model and Graphical User Interface (GUI). The model includes the various classes developed to perform the scheduling and control calculations. The GUI includes the user-interface aspects of Residential Planner and incorporates menus, toolbar, status bar and dialog boxes.

CHAPTER 6

APPLICATION EXAMPLES

6.1 Introduction

This chapter presents the application aspects of the present scheduling and tracking and control model for residential housing projects. The model considers a number of practical aspects often found in these projects. It can be applied to schedule developers' own work force and/or subcontractors, and can evaluate the cost and time performance of an on-going project at the project, unit and subcontractor levels. Two application examples are analyzed in order to demonstrate the various practical aspects of the model. The first example validates the results and the reports generated by the model and the second example illustrates the capabilities of the present model.

6.2 First Example

An application example project from the literature (Lumsden, 1968) is analyzed in order to validate the results generated by the developed model. The project involves the construction of 10 typical houses. The construction works involves four main typical repetitive activities: 1) *substructure*, 2) *superstructure*, 3) *finishes* and 4) *decoration*. The precedence relationships among these activities are assumed to be finish to start with no lag time. Multiple crews are employed to perform work in the activities of *substructure, superstructure and finishes* (2, 3 and 5, respectively), whereas a single crew is used for *decoration* activity. The

original schedule from the literature is shown in Figure 6.1, and has a total project duration of 47.5 days. It should be noted that in the original example both crews working in the substructure activity had to wait a day before commencing work in the next assigned housing unit. Similarly, the crews employed in superstructure activity had a waiting time of 1.5 days before starting work in their next assigned housing unit. To enable a comparison between the schedule developed by the present model and that of Lumsdens' (1968), identical number of crews (2, 3, 5 and 1 for *substructure, superstructure, finishes and decoration*, respectively) are used in the present example. Accordingly, quantity of work for each activity and the productivity of crews used are obtained from Means Residential Cost Data (1999). Table 6.1 lists the input data for the present model.

Table 6.1 Input Data

	Substructure	Superstructure	Finishes	Decoration
Quantity of work	1840 m^3	120 m^3	1040 m^2	100 m
Number of crews	2	3	5	1
Crew ID	B-12A	C-6	J-1	F-2
Crew output / day	460 m^3	20 m^3	80 m^2	50 m
Cost / Unit ($)	1332	2664	2331	555

The schedule developed by the present model is as shown in Figure 6.2 and listed in Table 6.2. In addition to complying with precedence relationships, the developed schedule guarantees work continuity for all crews, including the crews utilized in activities *substructure* and *superstructure*. For example, crew 1 of *substructure* activity promptly moves to housing unit 3 after completing work in housing unit 1, without any idle time. It should be noted that the developed project schedule has a total duration of 46 days, a reduction of 1.5 days due to the elimination of crew idle time and improved resource utilization.

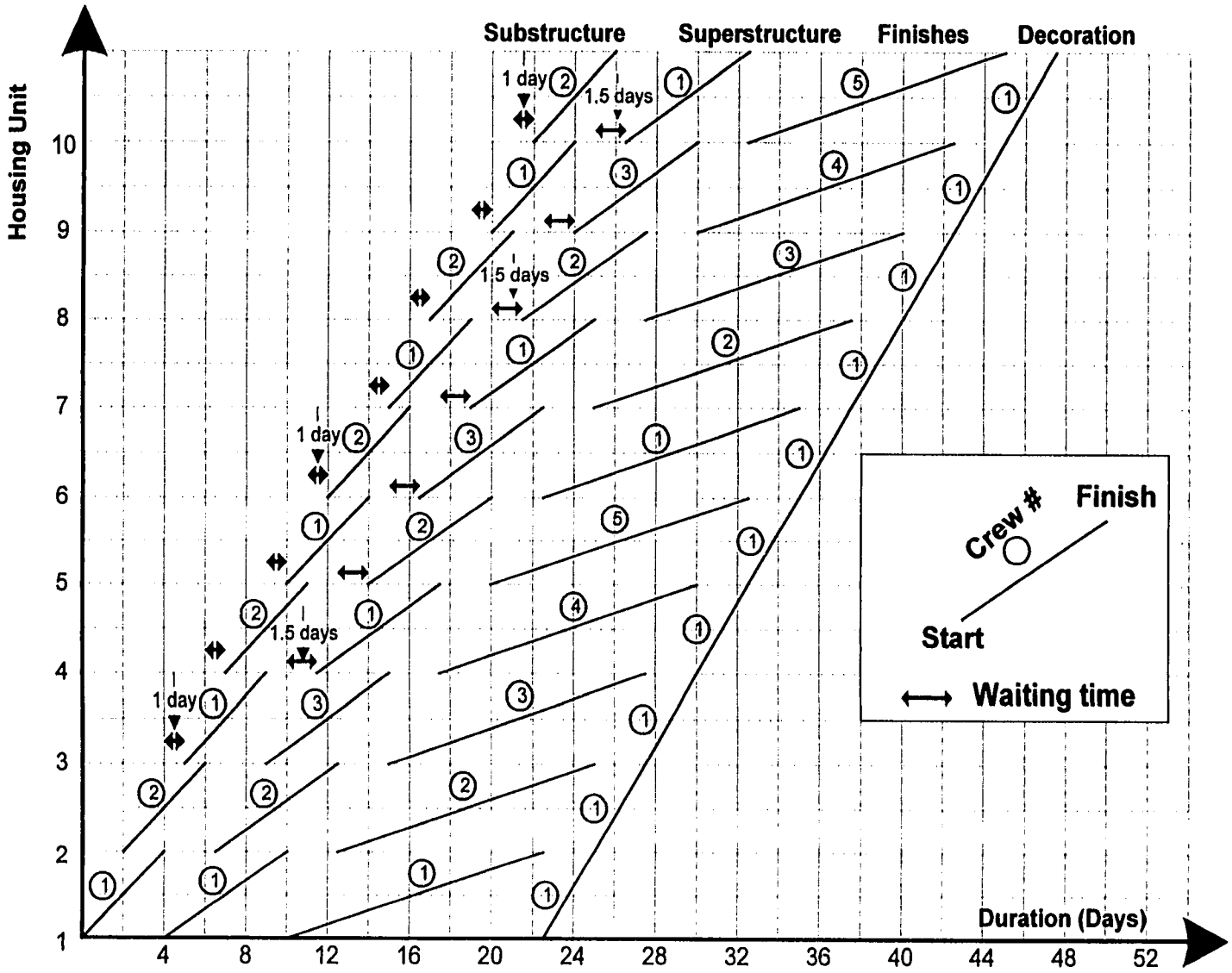


Figure 6.1 Project Schedule from Literature (Lumsden, 1968)

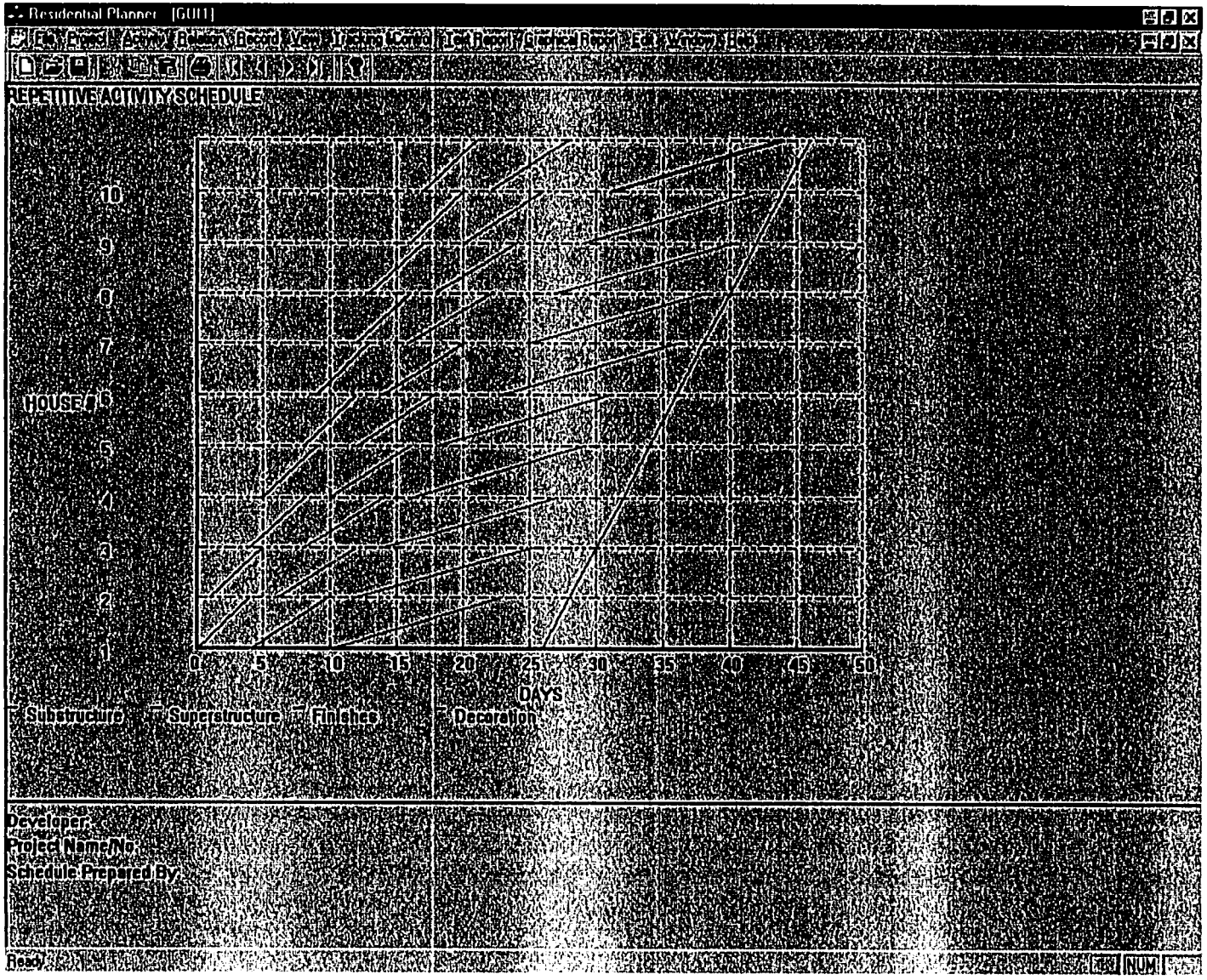


Figure 6.2 Project Schedule Developed by the Model

Table 6.2 Project Schedule

Housing Unit #	Substructure				Superstructure				Finishes				Decoration			
	Dur	start	Finish	Crew #	Dur	start	Finish	Crew #	Dur	start	Finish	Crew #	Dur	start	Finish	Crew #
1	4	0	4	1	6	4	10	1	13	10	23	1	2	26	28	1
2	4	1	5	2	6	6	12	2	13	12	25	2	2	28	30	1
3	4	4	8	1	6	8	14	3	13	14	27	3	2	30	32	1
4	4	5	9	2	6	10	16	1	13	16	29	4	2	32	34	1
5	4	8	12	1	6	12	18	2	13	18	31	5	2	34	36	1
6	4	9	13	2	6	14	20	3	13	23	36	1	2	36	38	1
7	4	12	16	1	6	16	22	1	13	25	38	2	2	38	40	1
8	4	13	17	2	6	18	24	2	13	27	40	3	2	40	42	1
9	4	16	20	1	6	20	26	3	13	29	42	4	2	42	44	1
10	4	17	21	2	6	22	28	1	13	31	44	5	2	44	46	1

In order to validate the tracking and control results, the cost data relating to each activity obtained from the original example is used (see Table 6.1). The cumulative cash flow generated according to the original cost data (Lumsden, 1968) at different time intervals is listed in Table 6.3. The cumulative cash flow curve generated by the present model is shown in Figure 6.3, which closely resembles the cash flow profile of Table 6.3. The total project cost in the original solution and the present model is found to be $ 68,820.00 and $ 68,460.00, respectively. This difference is attributed due to the better resource utilization. It should be noted that the project indirect cost is not considered in both solutions.

Table 6.3 Project Cumulative Cash Flow in $ (Lumsden, 1968)

Activity	Project Time (Workday)				
	10	20	30	40	47.5
Substructure	6600	12600	13320	13320	13320
Superstructure	5800	18648	26640	26640	26640
Finishes	-----	5828	16317	20979	23310
Decoration	------	-----	1665	3885	5550
Total	12400	37076	57942	64824	68820

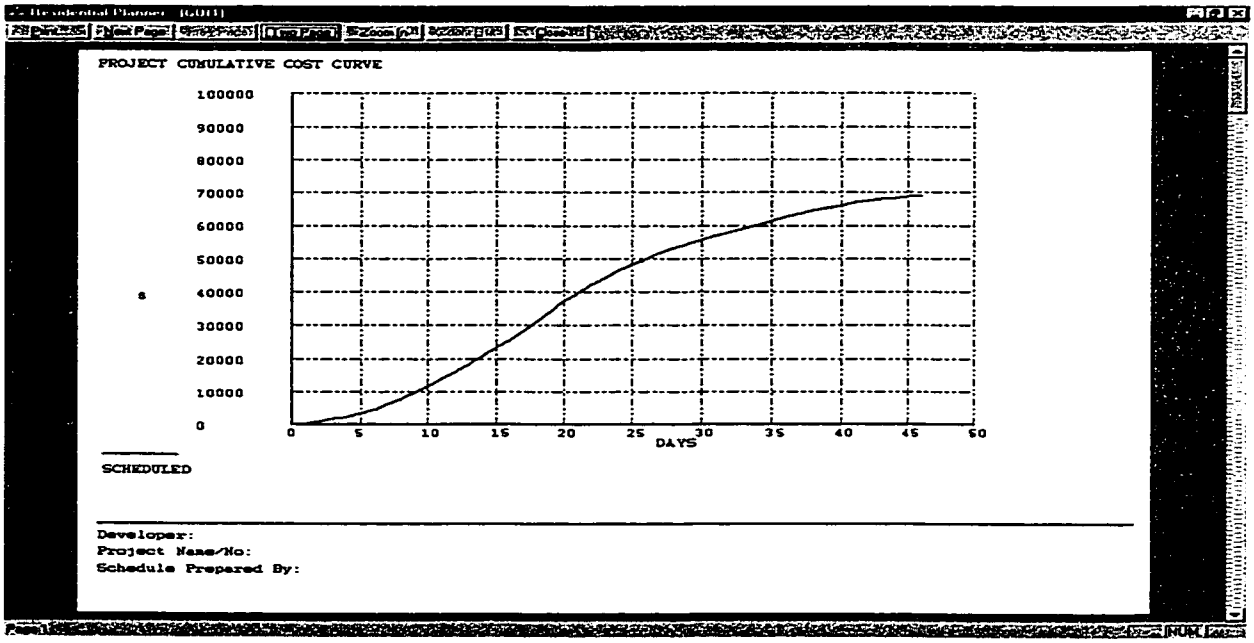


Figure 6.3 Cumulative Cash Flow Curve Generated by the Model

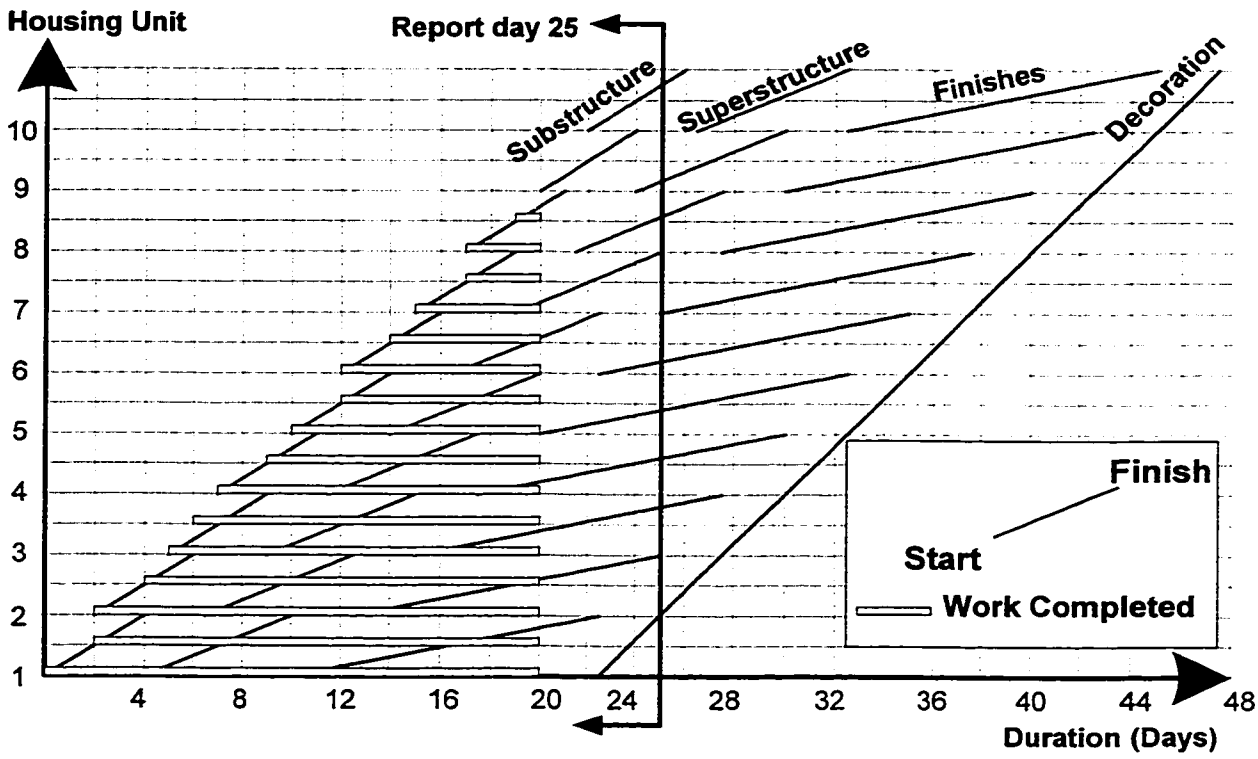


Figure 6.4 Actual Progress (Lumsden, 1968)

The performance status of the project with respect to time and cost is analyzed on day 25. The percentage of work completed in *substructure, superstructure, finishes and decoration* are 75%, 60%, 20% and 0%, respectively. The actual cost of work performed (ACWP) up to the report day 25 is assumed to be $ 34,200.00. The actual progress is shown in Figure 6.4, which indicates that the work progress is behind schedule. The planned cumulative work progress and cost for the entire project duration (day 1 to 46) is calculated according to the first stage of the tracking and control algorithm explained in Chapter 4 and is listed in Table 6.4. The second stage of the developed tracking and control algorithm is applied to calculate the actual work performed (AWP = 36.46%). The planned duration for work performed (PDWP) is identified to be 18 days by the third stage of the algorithm (see highlighted column in Table 6.4). Hence, the project is

behind schedule by 7 days (SV = PDWP – ADWP = 18 – 25 = -7). The budgeted cost of work performed (BCWP) is also calculated in the third stage of the algorithm (BCWP = $ 31,239.00) indicating that project is over budget (i.e. CV = BCWP – ACWP =31,239.00 - 34,200.00 = - 2961.00). The resulting planned and actual work progress curves generated by the model is shown graphically in Figure 6.5, which indicates that work progress was ahead at the beginning but has fallen behind schedule on report day.

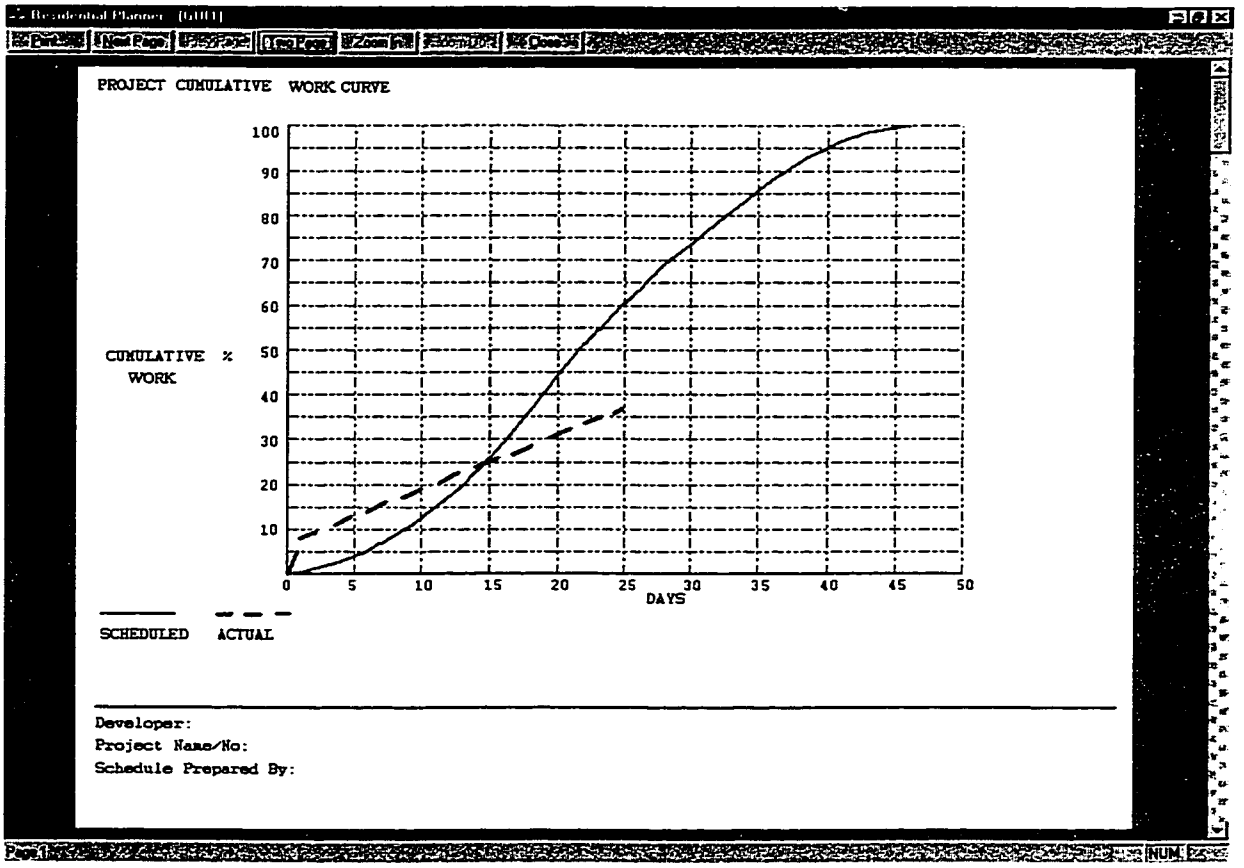


Figure 6.5 Performance Results Generated by the Model

Table 6.4 Planned Cumulative Work Percentage and Cost for Each Workday

Workday	1	2	3	4	5	6	7	8	9	10
Work (%)	0.407	1.201	2.00	2.808	4.00	5.20	6.80	8.40	10.40	12.40
Cost ($)	333	999	1665	2331	3441	4551	6105	7659	9657	11,655

Workday	11	12	13	14	15	16	17	**18**	19	20
Work (%)	14.80	17.20	20.00	22.80	26.00	29.20	32.80	**36.40**	40.40	44.40
Cost ($)	13833	16011	18369	20727	23265	25803	28521	**31239**	34137	37035

Workday	21	22	23	24	25	26	27	28	29	30
Work (%)	48.00	51.20	54.40	57.60	60.40	63.20	66.01	68.80	71.20	73.60
Cost ($)	39600	41832	44064	46296	48084	49872	51494	53116	54294	55472

Workday	31	32	33	34	35	36	37	38	39	40
Work (%)	76.00	78.40	80.80	83.20	85.60	88.00	90.00	92.00	93.60	95.20
Cost ($)	56650	57828	59006	60184	61362	62540	63538	64536	65354	66172

Workday	41	42	43	44	45	46				
Work (%)	96.40	97.60	98.40	99.20	99.60	100.00				
Cost ($)	66610	67348	67606	68064	68142	68460				

6.3 Second Example

This example application is designed to demonstrate the practical features and the efficiency of the developed model. The example project involves the construction of 10 typical houses of economy class with a living area of 1200 S.F. Accordingly, quantity of work, duration and cost are calculated from Means Residential Cost Data (1999). The construction work involves six main activities: 1) *permit,* 2) *excavation*, 3) *foundation*, 4) *framing,* 5) *finishing* and 6) *cleaning.* The precedence relationships among these activities are assumed to be finish to start with no lag time. The following assumptions are made to illustrate various practical features in the developed model:

1) Planned start date of the project is January 04, 2000.
2) Activities *permit* and *cleaning* represent non-repetitive activities in this project. The duration and direct cost of these activities are considered 3 days and $ 1500.00, respectively.
3) The relation between activities *permit* and *excavation, finishing* and *cleaning* illustrate relationships among repetitive and non-repetitive activities (i.e. Hetero relation). *Excavation* can start only after activity *permit* is finished. *Cleaning* work can start as soon as the *finishing* work in the tenth housing unit is completed.
4) Activity *excavation, foundation* and *framing* are assumed to have identical duration in all housing units and are classified as typical repetitive activities. However, *finishing* has different duration in each house and represents a non-typical repetitive activity in this project. It is assumed that the difference in

duration is attributed due to two available options for the potential buyer. These options are 1) having both living and master bedroom decorated or 2) having only the living room decorated. The buyer's of units 1,5,7 and 9 have chosen option one and the rest, option two.

5) Scheduling for developers' own force is demonstrated by assuming that the *foundation* activity is planned to be performed by utilizing own work force. In order to accelerate this activity, two crews (crew 1 and 2) are being used. The productivity and cost of these crews are obtained from the developed database.

6) To illustrate subcontractor scheduling, activity *excavation, framing* and *finishing* are assumed to be performed by subcontractors. A lump sum contract for labor ($72,000.00) is awarded to perform excavation with an expected duration of 30 days. A unit price contract for labor and equipment ($ 19.00 S.F) is awarded to execute *framing* activity with an expected duration of 20 workdays. It is assumed that material for both these activities are provided by the developer ($ 99,000.00 and $ 192,000.00 respectively). However, finishing activity is awarded a one payment lump-sum contract ($ 20,000.00), which includes both material and installation costs with an expected duration of 14 days.

7) Crew availability constraint is illustrated by assuming that crew 2 utilized in *foundation* activity can move to the site on February 05, 2000 and can stay only till February 20, 2000. Such a scenario can often occur due to prior commitments of the crew in other concurrent projects.

8) To demonstrate the impact of construction execution order, it is assumed that the *finishing* activity has to be performed in a user-specified order (2,1,3,4,6,5,7,8,9,10). Such a situation can occur due to contractual obligations regarding move-in date. Other three repetitive activities are executed in ascending order (i.e. 1,2,3...9,10).
9) The project average indirect cost is assumed to be $ 1000.00 per workday.

The generated schedule for non-repetitive and repetitive activities is listed in Tables 6.5 and 6.6 and shown in Figures 6.6 and 6.7, respectively. The schedule strictly complies with precedence relationship, crew availability and crew work continuity constraints. It should be noted that: 1) crew 2 of *foundation* activity is assigned only to housing units 7 and 9, due to the imposed early and late available dates, 2) finishing activity is scheduled according to user-specified execution order and 3) cleaning activity is scheduled to start as soon as finishing work is completed in the tenth housing unit. As mentioned earlier, the model can generate separate schedules for subcontractors and units as shown in Figures 6.8 and 6.9, respectively. The calculated total project cost is equal to $843,587.00 and is shown in Figure 6.10. The project summary report generated by the model is shown in Figure 6.11, which summarizes the project general details, start and finish dates and direct cost of each activity.

Table 6.5 Non-repetitive Activity Schedule

Name	Duration	Early Start	Early Finish	Late Start	Late Finish	Float
Permit	3	0	3	0	3	0
Cleaning	3	41	44	41	44	0

Table 6.6 Repetitive Activity Schedule

Unit #	Excavation			Foundation				Framing			Finishing		
	Duration	start	Finish	Duration	start	Finish	Crew #	Duration	start	Finish	Duration	start	Finish
1	3	3	6	3	12	15	1	2	20	22	2	28	30
2	3	6	9	3	15	18	1	2	22	24	1	27	28
3	3	9	12	3	18	21	1	2	24	26	1	30	31
4	3	12	15	3	21	24	1	2	26	28	1	31	32
5	3	15	18	3	24	27	1	2	28	30	2	33	35
6	3	18	21	3	27	30	1	2	30	32	1	32	33
7	3	21	24	3	27	30	2	2	32	34	2	35	37
8	3	24	27	3	30	33	1	2	34	36	1	37	38
9	3	27	30	3	30	33	2	2	36	38	2	38	40
10	3	30	33	3	33	36	1	2	38	40	1	40	41

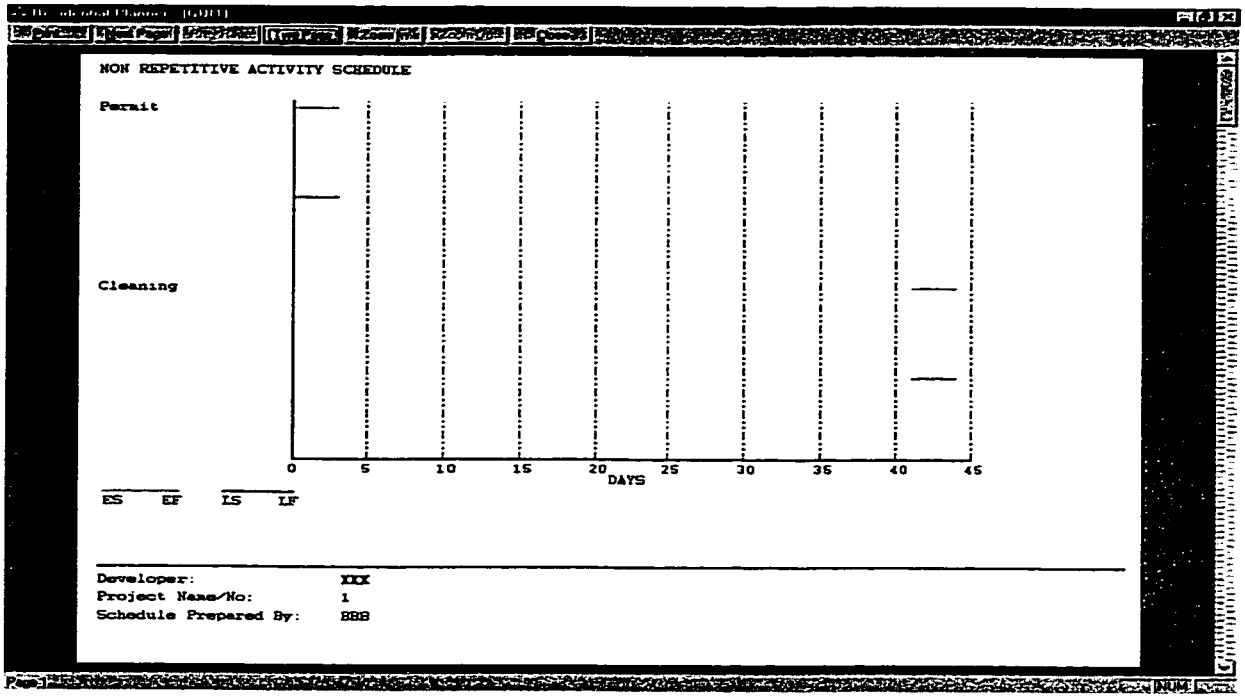


Figure 6.6 Non-repetitive Activity Schedule

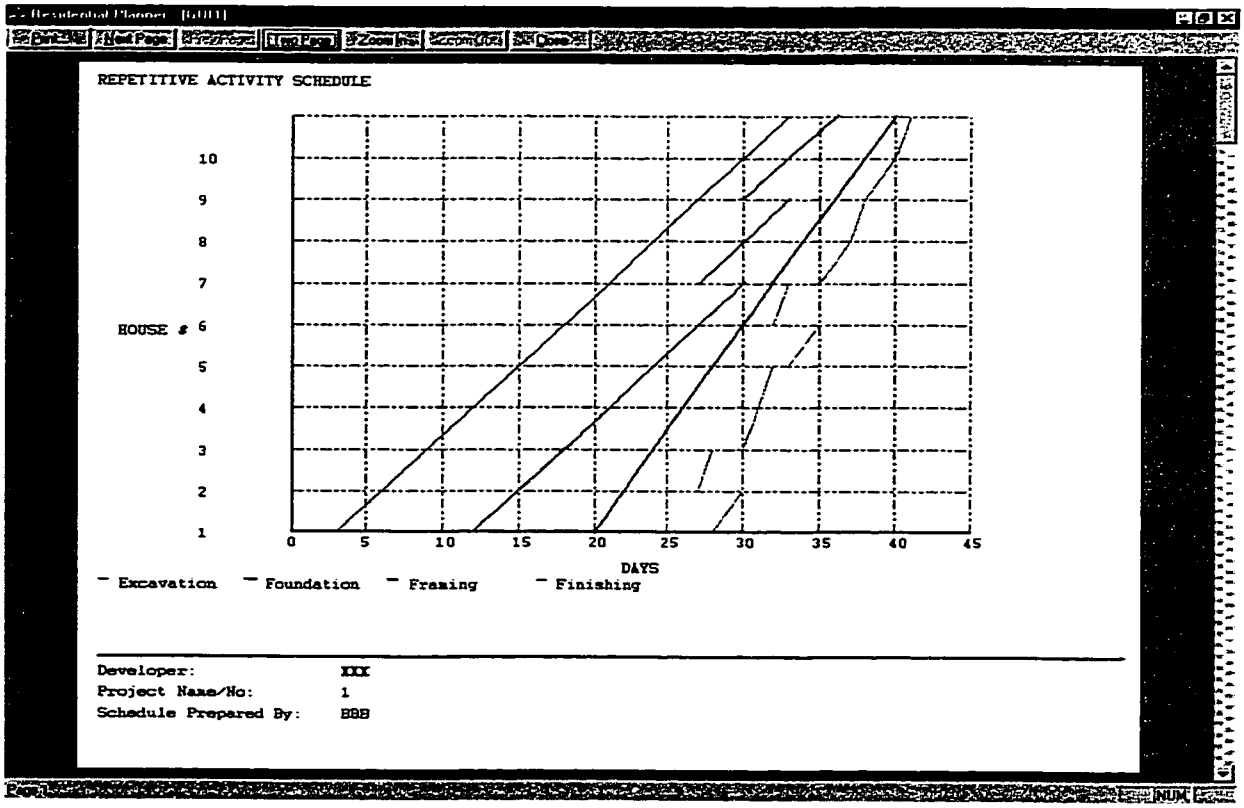


Figure 6.7 Repetitive Activity Schedule

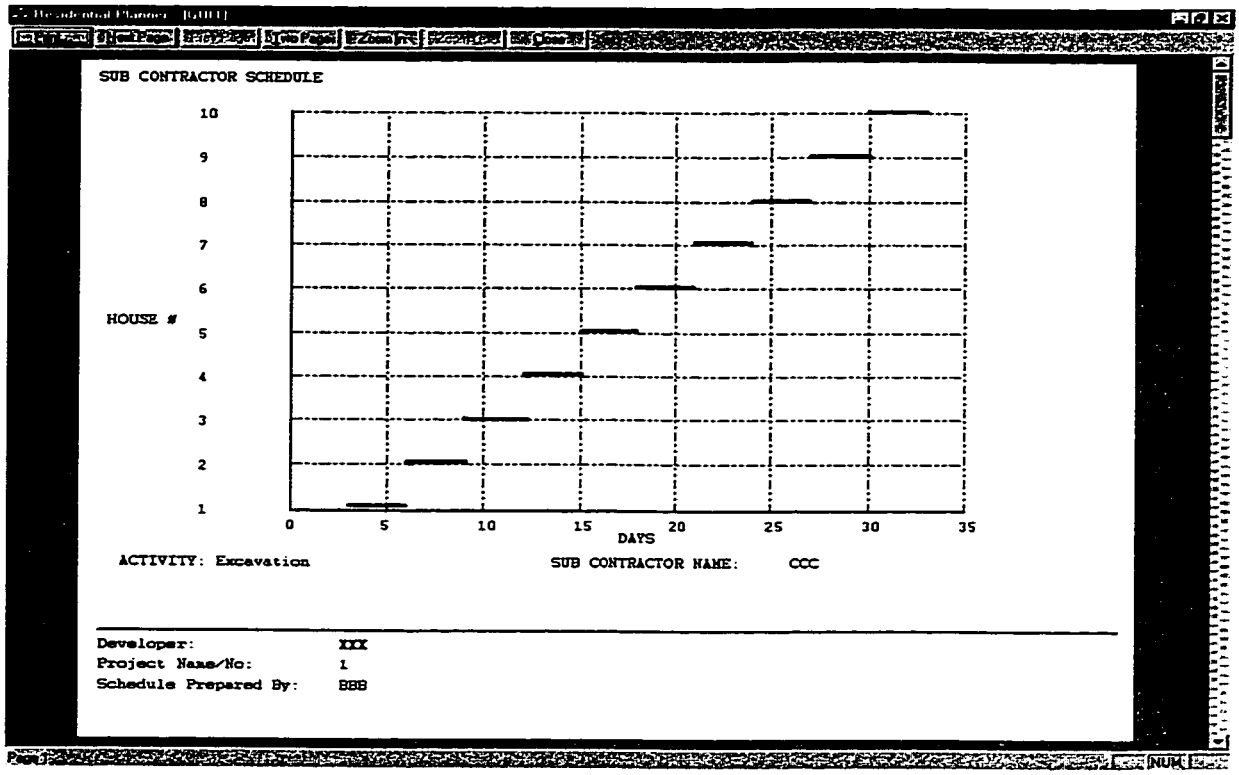
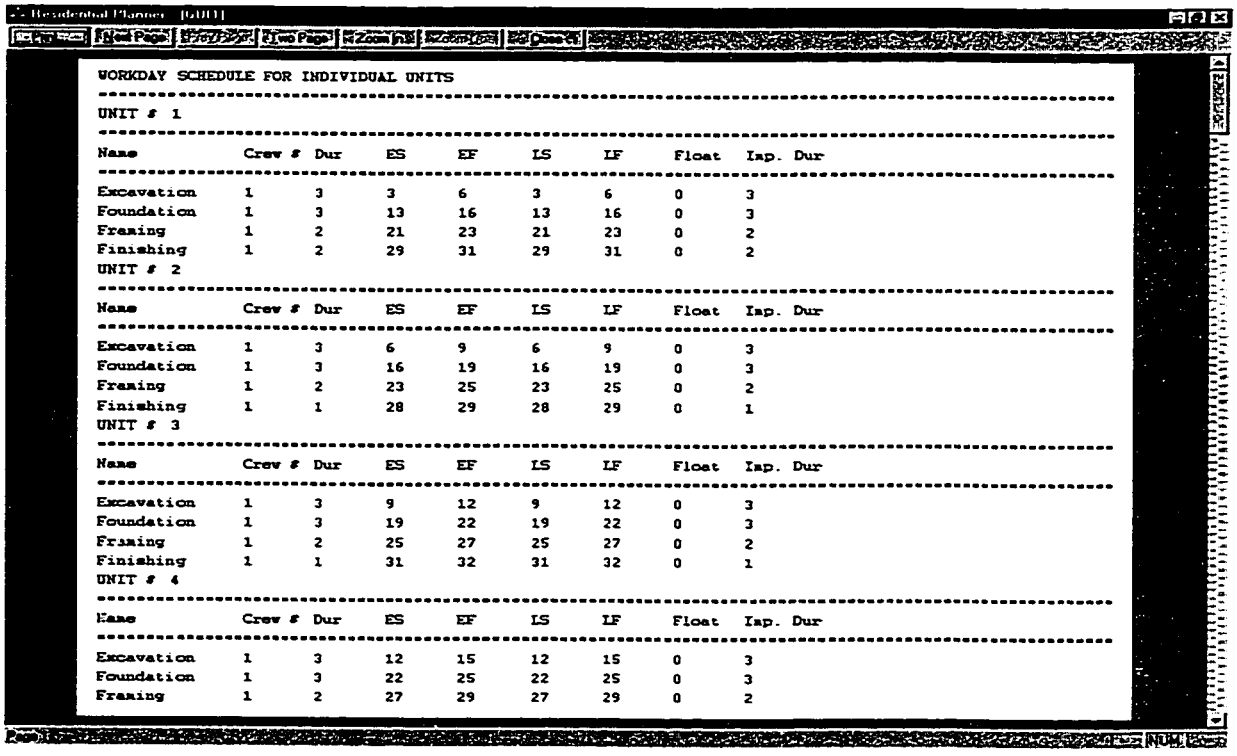


Figure 6.8 Schedule for Subcontractor



WORKDAY SCHEDULE FOR INDIVIDUAL UNITS

UNIT # 1

Name	Crew #	Dur	ES	EF	LS	LF	Float	Imp. Dur
Excavation	1	3	3	6	3	6	0	3
Foundation	1	3	13	16	13	16	0	3
Framing	1	2	21	23	21	23	0	2
Finishing	1	2	29	31	29	31	0	2

UNIT # 2

Name	Crew #	Dur	ES	EF	LS	LF	Float	Imp. Dur
Excavation	1	3	6	9	6	9	0	3
Foundation	1	3	16	19	16	19	0	3
Framing	1	2	23	25	23	25	0	2
Finishing	1	1	28	29	28	29	0	1

UNIT # 3

Name	Crew #	Dur	ES	EF	LS	LF	Float	Imp. Dur
Excavation	1	3	9	12	9	12	0	3
Foundation	1	3	19	22	19	22	0	3
Framing	1	2	25	27	25	27	0	2
Finishing	1	1	31	32	31	32	0	1

UNIT # 4

Name	Crew #	Dur	ES	EF	LS	LF	Float	Imp. Dur
Excavation	1	3	12	15	12	15	0	3
Foundation	1	3	22	25	22	25	0	3
Framing	1	2	27	29	27	29	0	2

Figure 6.9 Schedule for Individual Housing Units

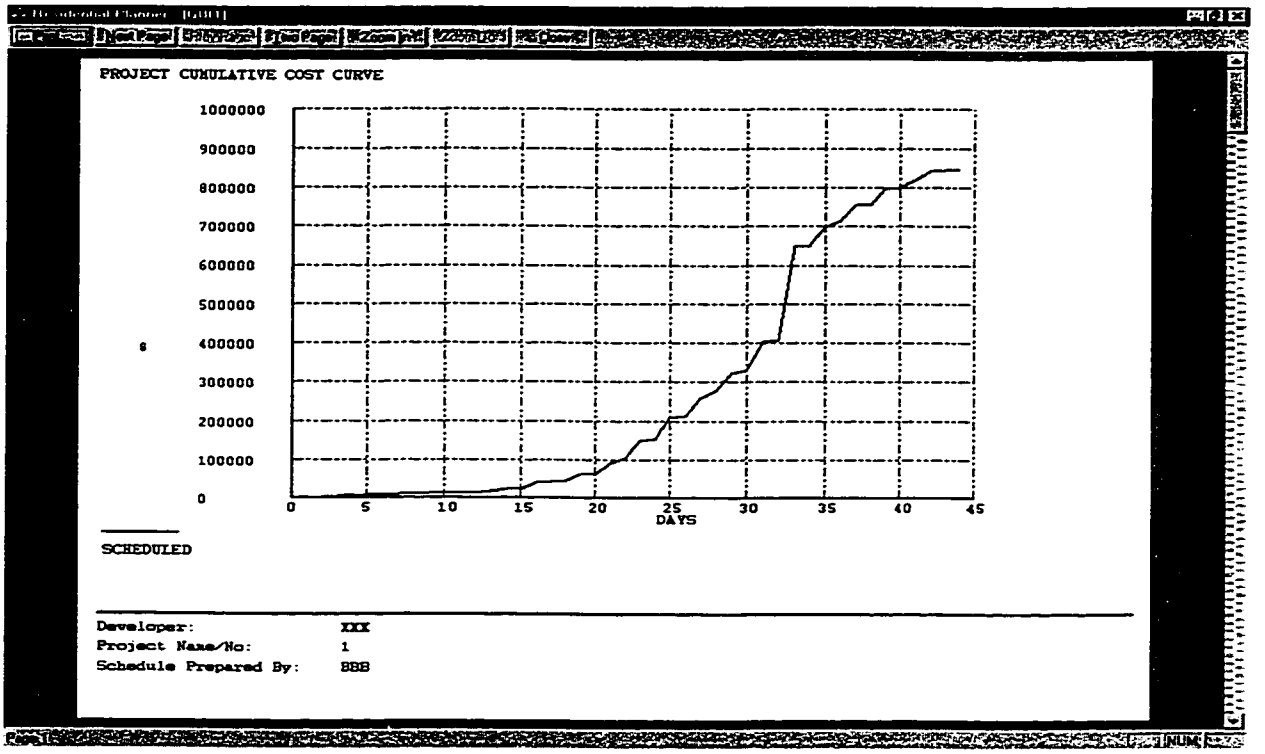
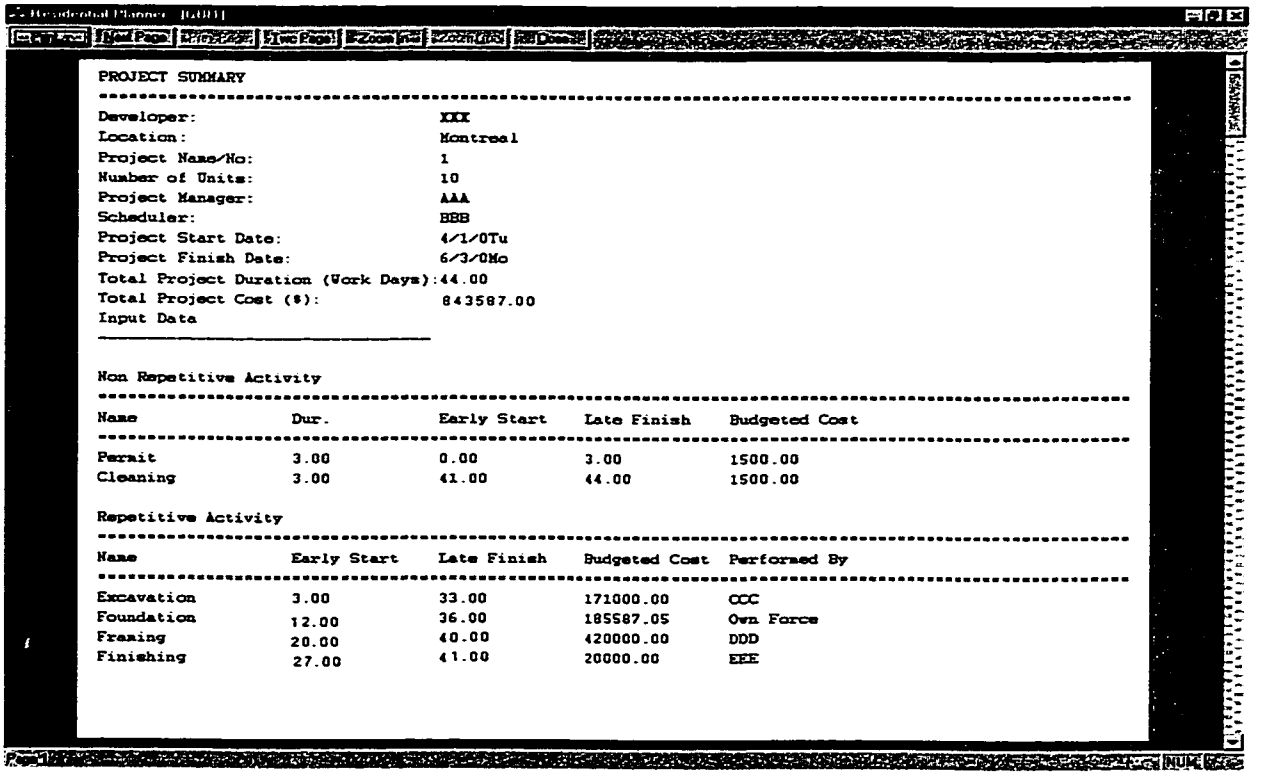


Figure 6.10 Project Cumulative Cost Curve



PROJECT SUMMARY

Developer:	XXX
Location:	Montreal
Project Name/No:	1
Number of Units:	10
Project Manager:	AAA
Scheduler:	BBB
Project Start Date:	4/1/0Tu
Project Finish Date:	6/3/0Mo
Total Project Duration (Work Days):	44.00
Total Project Cost ($):	843587.00

Input Data

Non Repetitive Activity

Name	Dur.	Early Start	Late Finish	Budgeted Cost
Permit	3.00	0.00	3.00	1500.00
Cleaning	3.00	41.00	44.00	1500.00

Repetitive Activity

Name	Early Start	Late Finish	Budgeted Cost	Performed By
Excavation	3.00	33.00	171000.00	CCC
Foundation	12.00	36.00	185587.05	Own Force
Framing	20.00	40.00	420000.00	DDD
Finishing	27.00	41.00	20000.00	EEE

Figure 6.11 Project Summary

In order to demonstrate the tracking and control features of the developed model, it is assumed that the analysis is carried out on workday 35. The actual progress till day 35 is assumed to 100%, 100%, 100% 85% and 60% in activities *permit, excavation, foundation, framing* and *finishing*, respectively and the actual cost of work performed is assumed to be $ 685,400.00. The result of the analysis is shown in Figures 6.12 and 6.13, which indicates that the project is ahead of schedule by 1 day and under budget by $ 14,600.00. The performance of an individual subcontractor (*framing* activity) and a particular housing unit (unit # 8) is also analyzed on the same report day and the results are shown in Figures 6.14 and 6.15, respectively.

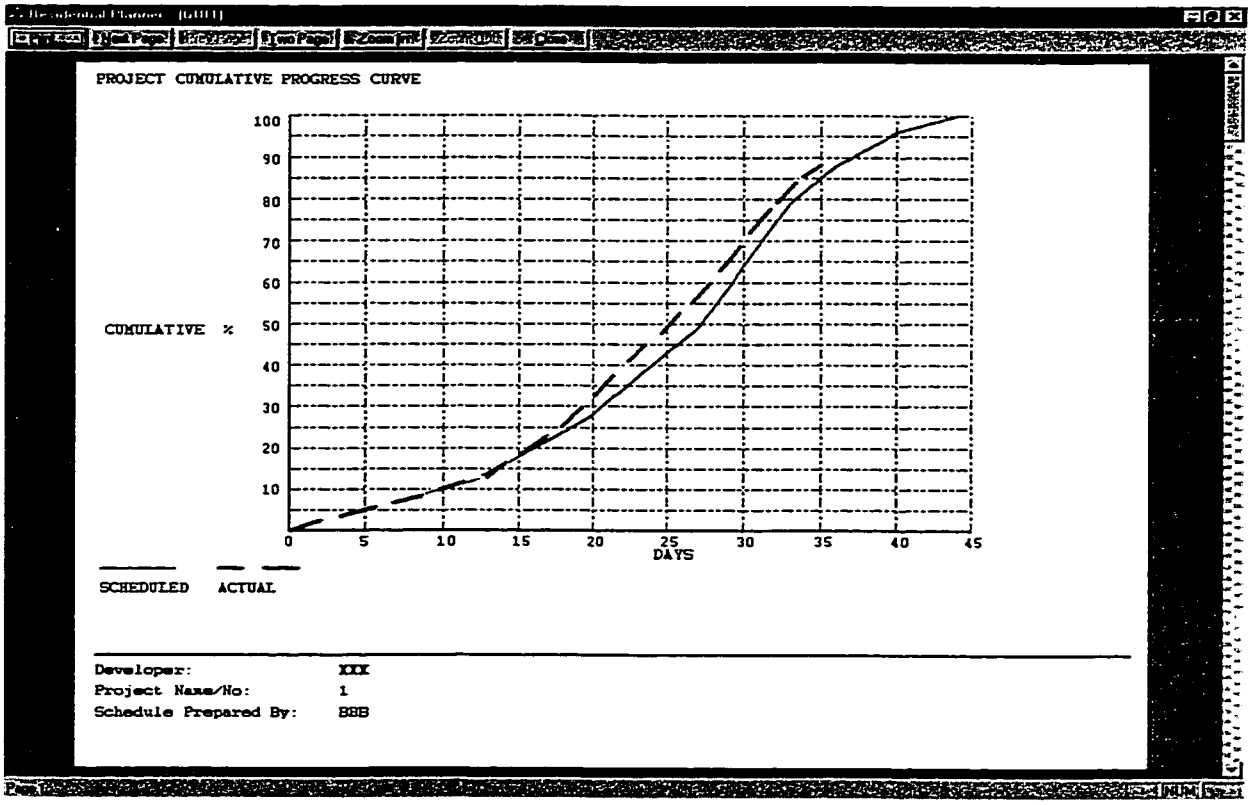


Figure 6.12 Project Performance Curves

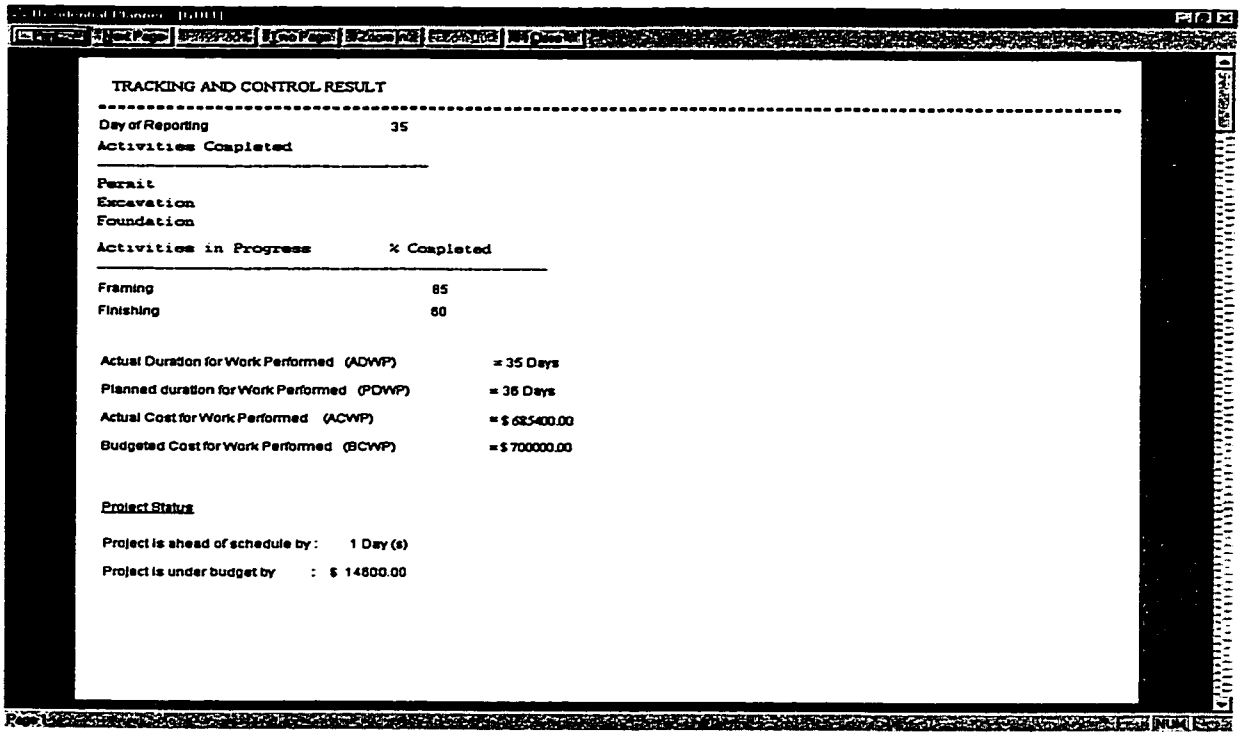


Figure 6.13 Project Tracking and Control Results

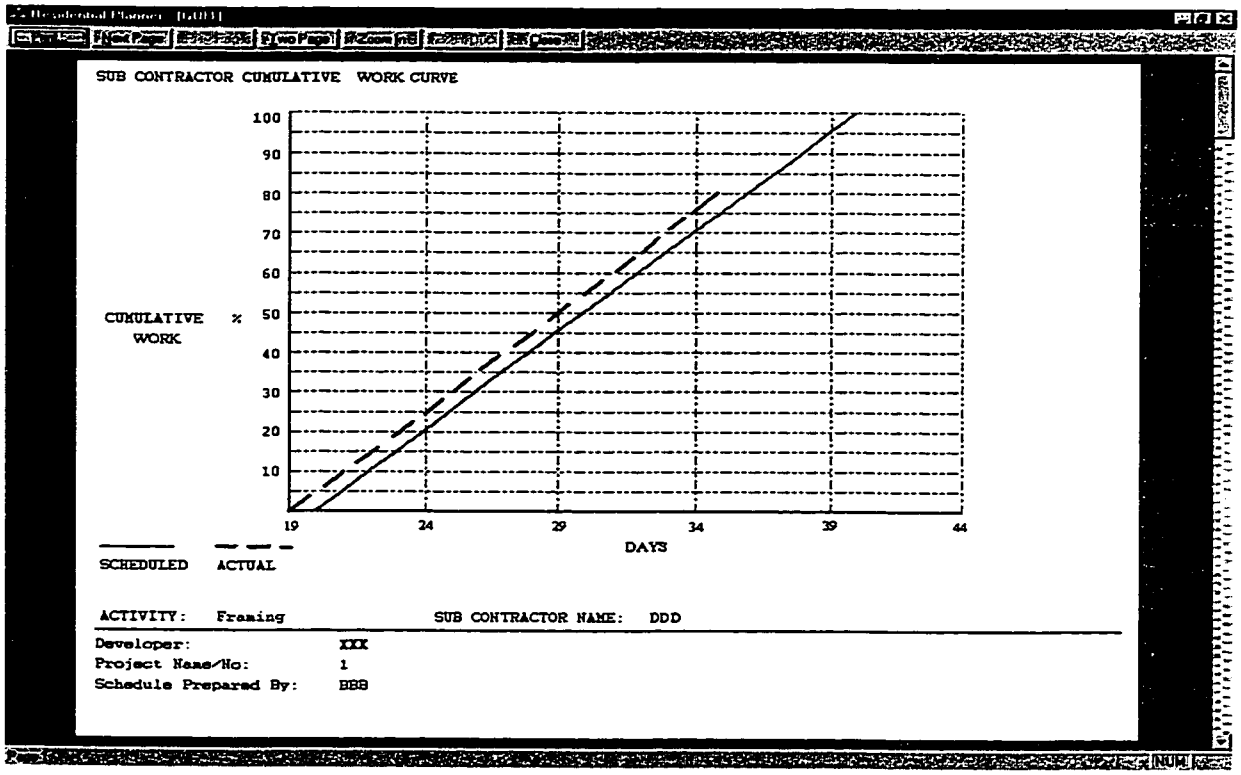


Figure 6.14 Subcontractor Performance Curves

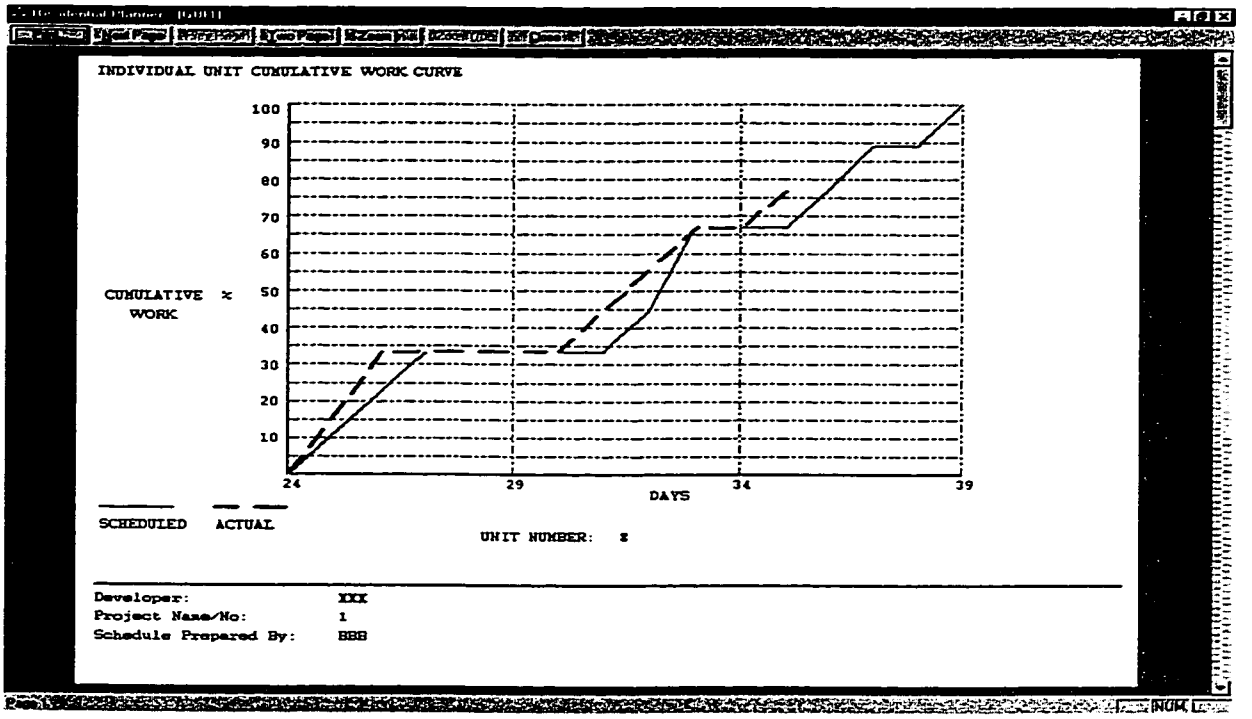


Figure 6.15 Unit Performance Curves

6.4 Summary

The practical features of the developed model have been presented in this chapter by analyzing two application examples. The first example is used to validate the results produced by the model and to highlight the important improvements in the schedule and control results generated by the model. The second example is designed to demonstrate the practicality and flexibility of the model. A number of challenges that are often faced in scheduling and tracking and control of residential housing projects are addressed in this example. The generated results and reports are briefly described and presented in a number of tables and figures. As illustrated in the second example, the model considers various practical aspects and can be effectively applied in scheduling and control of projects, leading to potential savings in construction time and cost.

CHAPTER 7

CONCLUSIONS

7.1 Summary and Concluding Remarks

An Object-Oriented model for scheduling and tracking and control of residential housing projects has been developed. A number of practical factors that affect the scheduling and control of these types of projects are considered. As such, the developed model can be effectively applied to real-life projects, and can lead to savings in project time and cost. The model is developed using an Object-Oriented modeling approach in three main stages: analysis, design and implementation. The model includes 18 classes that are designed and implemented to facilitate the scheduling and control of residential housing projects. The model also incorporates a number of scheduling and control algorithms including the newly developed algorithms for scheduling subcontractors in repetitive activities and for tracking and control of residential housing construction.

The scheduling algorithm for subcontractors in repetitive activities is designed to comply with three main constraints namely, precedence relationships, subcontractor availability period and crew work continuity. In addition to these three constraints, the algorithm considers a number of practical factors commonly encountered in scheduling subcontractors. The algorithm is applied in two stages to determine the start and finish dates of the subcontractor work in

each housing unit. The first stage considers the precedence relationships and subcontractor availability period on site and the second stage complies with crew work continuity constraint.

The tracking and control algorithm is designed to determine the time and cost performance of an on-going project and can be applied at three levels: 1) entire project, 2) housing unit and 3) subcontractor. The algorithm performs tracking and control calculations in three stages. The first stage determines the planned cumulative work and cost for each workday according to the developed base line schedule. The second stage calculates the cumulative actual work performed (AWP) up to report date based on the user-specified actual progress data. The third stage first determines the planned duration for work performed (PDWP) and the budgeted cost for work performed (BCWP). This stage then compares the actual and planned duration for work performed (i.e. ADWP and PDWP) in order to calculate the schedule variance (SV). The SV value can be 0, <0 or >0, representing on, behind or ahead of schedule, respectively. The cost status is determined by comparing the budgeted cost for work performed (BCWP) to the actual cost for work performed (ACWP) to calculate the cost variance (CV). The value of CV can be 0, >0 or <0 representing on, under or over budget, respectively.

The developed model can generate specialized reports to suit the diverse needs of residential housing projects and is supported by a database to store historical

data that can be retrieved for use in future projects. The generated reports include project schedule, unit schedule, subcontractor schedule and control results in both text and graphical formats. The developed model is implemented as a user-friendly prototype software system using Visual C++ 6.0 and Microsoft Foundation Class (MFC). The implemented model is named as 'Residential Planner' and runs in Windows 2000 and NT. It supports user-friendly interfaces and includes menus, toolbar, status bar and dialog boxes.

7.2 Research Contributions

The primary contribution of this research is the development of a scheduling and tracking and control model for residential housing projects: 'Residential Planner'. The development of this model incorporates the following contributions:

1) The development of a scheduling algorithm for subcontractors in residential housing construction.

2) The formulation of an algorithm for effective tracking and control of housing construction at three levels: project, housing unit, and subcontractor.

3) The development of effective methods to generate flexible and specialized reports to address the needs of developer and subcontractors involved in residential housing projects.

4) The implementation of the proposed algorithms and methodologies in a user-friendly prototype software system.

5) The provision of a database support for efficient use of the developed computer software and store valuable historical data.

7.3 Recommendations for Future Research

A practical scheduling and tracking and control model for residential housing projects has been presented in this study. The model is flexible and can be applied to schedule and control residential housing projects. However, in order to expand the potential applications of this model, the following recommendations for future research can be made:

1) The schedule developed by this model considers various practical aspects, but does not guarantee optimized solution with respect to cost and/or time. An optimization procedure can be adopted to provide optimized solutions.

2) The developed scheduling algorithm provides a deterministic schedule, but can be expanded to produce probabilistic schedules.

3) The developed tracking and control algorithm determines the status of an on-going project at any given time, but does not update the original schedule according to the actual progress information. This can be achieved by developing a schedule updating methodology.

4) The developed prototype software system, 'Residential Planner' can accommodate only one project at a time, but can be extended to account for multiple projects sharing the same resource pool.

REFERENCES

Al Sarraj, Z. M. (1990). "Formal Development of Line-of-Balance Technique," **Journal of Construction Engineering and Management**, ASCE, 116(4), 689-704.

Arditi, D., and Albulak, M.Z. (1986). "Line-of-Balance Scheduling in Pavement Construction," **Journal of Construction Engineering and Management**, ASCE, 112(3), 411-424.

Ashley, D.B. (1980). "Simulation of Repetitive Unit Construction," **Journal of the Construction Division**, ASCE, 106(C02), 185-194.

Barrie, D. S., and Paulson, B.C. Jr. (1992). **Professional Construction Management**, McGraw-Hill Inc., New York.

Birrell, G. E. (1980). "Construction Planning-Beyond the Critical Path," **Journal of Construction Division**, ASCE, 106(CO3), 389-407.

Booch, G. (1994). **Object-Oriented Analysis and Design with Applications**, Redwood City, CA.

Carr, R. I., and Meyer, W. L. (1974). "Planning construction of Repetitive Building Units," **Journal of the Construction Division**, ASCE, 100(C03), 403-412.

Chzanowski, E. N., and Johnston D. W. (1986). "Application of Linear Scheduling," **Journal of Construction Engineering and Management**, ASCE, 112(4), 476-491.

CMHC, (1996), "Long-Term Housing Outlook: Household Growth, 1991-2016," **Research and Development Highlights**, Canada Mortgage and Housing Corporation, Ottawa, Ontario K1A 0P7.

Davies, E. W. (1974). " CPM Use in the Top 400 Construction Firms," **Journal of Construction Division**, ASCE, 100(CO1), 39-49.

Diekmann, E., and Masoud, A. A. (1997). "Repetitive Project Scheduling for Large Scale Residential Housing," **Proceedings of Construction Congress v**, Minneapolis, Minnesota, 921-930.

Dressler, J. (1980). "Construction Management in West Germany," **Journal of the Construction Division**, ASCE, 106(C04), 447-487.

El-Rayes, K., and Moselhi, O. (1998) "Resource-driven Scheduling of Repetitive Activities," **Journal of Construction Management and Economics**, ASCE, 16, 433-446.

El-Rayes, K. (1997), "**Optimized Scheduling for Repetitive Construction Projects**," a Ph.D. thesis at the school for building, Faculty of Engineering and Computer Science, Concordia University, Montreal.

Gregory, K. (1999). **Using Visual C ++ 6**, QUE, Indianapolis, Indiana.

Hegazy, T., Fazio. P., and Moselhi. O. (1993). "BAL: An Algorithm for Scheduling and Control of Linear Projects," **1993 AACE Transactions**, AACE.

Johnston, D. W. (1981). "Linear Scheduling Method for Highway Construction," **Journal of the Construction Division**, ASCE, 107(C02), 247-261.

Kavanagh, D. P. (1985). "SIREN: A repetitive Construction Simulation Model," **Journal of Construction Engineering and Management**, ASCE, 111(3), 308-323.

Khoshafian, S., and Abnous, R. (1995). **Object Orientation: Concepts, Analysis & Design, Languages, Database, Graphical User Interface, Standards**, Wiley, New York.

Lumsden, P. (1968). **The Line of Balance Method**, Pergamon, Telgamon Press Ltd., London, England.

Martin, J. (1993). **Principles of Object-Oriented Analysis and Design**, P T R Prentice-Hall, Inc., New Jersey.

Moselhi, O., and El-Rayes, K. (1993). "An OOP Model for Scheduling of Repetitive Projects," **Proceedings of the Fifth International Conference on Computing in Civil and Building Engineering**, ASCE, California.

Neale, R. H., and Neale,D. E. (1989). **Construction Planning**. 1 st Edition, Thomas Telford Ltd., London, England.

Neale, R. H., and Raju, B. (1988). "Line of Balance Planning by Spread Sheet," **Building Technology and Management**, (January), 22-27.

Nunnally, S. W. (1998). **Construction Methods and Management**, 4 th edition, Prentice Hall, New Jersey.

O'Brien, J. J. (1975). "VPM Scheduling for High Rise Buildings," **Journal of the Construction Division**, ASCE, 101(C04), 895-905.

O'Brien, J.J. (1984). "Network Scheduling Variations for Repetitive Work," proceedings of the Spring Convention of ASCE, Atlanta, Ga.

Peer, S. (1974). "Network Analysis and Construction Planning," **Journal of the Construction Division**, ASCE, 100(3), 203-210.

Programming House Building by Line of Balance, (1966). The National Building Agency, London, England.

Rumbaugh, J., Blaha, M. Premerlani, W., Eddy, F. and Lorensen, W. (1991). **Object-Oriented Modeling and Design**, Englewood Cliffs, Prentice Hall, New Jersey.

Russell, A. D. (1989). "Advanced Planning and Control Technologies for Housing construction," **Canada Mortgage and Housing Corporation (CMHC) report**.

Russell, A. D., and Caselton, W. F. (1988). " Extensions of Linear Scheduling Optimization," **Journal of Construction Engineering and Management**, ASCE, 114(1), 36-52.

Russell, A. D., and Wong, W.C.M. (1993). "New Generation of Planning Structures," **Journal of Construction Engineering and Management**, ASCE, 119(2), 196-214.

Selinger, S. (1980). "Construction Planning for Linear Projects," **Journal of the Construction Division**, ASCE, 106(C02), 195-205.

Statistics Canada, (1998). "Capital Expenditure in Construction," **http:\www.statcan.ca, Canada**.

Stradal, O., and Cacha, J. (1982). "Time Space Scheduling Method," **Journal of Construction Division**, ASCE, 108(CO3).

Thabet, W. Y., and Beliveau, Y. J. (1994). "HVLS: Horizontal and Vertical Logic Scheduling for Multistory Projects," **Journal of Construction Engineering and Management**, ASCE, 120(4), 875-892.

Vorster, M. C., and Bafna, T. (1992). "Formal Development of Line-of-Balance Technique, Al Sarraj, Z. M. (1990)," **a discussion in Journal of Construction Engineering and Management**, ASCE, 118(1), 210-211.

Whiteman, W. E., and Irwing, H. G. (1988). "Disturbance Scheduling Techniques for Managing Renovation Work," **Journal of Construction Engineering and Management**, ASCE, 114(2), 191-213.

APPENDIX I

1) **Hossein Beheshti,**

   Northgrave Architect Inc.

   66, Gloucester Street,

   Toronto, Ontario,

   M4Y 1L5.

2) **Mr. Hazem Sharara,**

   Groupe Immobilier Grilli Inc.

   Montreal, Quebec.