ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF CIVIL AND ENVIRONMENTAL ENGINEERING

Presentation of CPM Using Matrix Schedule for High Rise Building Construction

By: Tesfa Fentaw

Advisor: Dr. Abraham Assefa

A Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Civil Engineering (Construction Technology and Management)

April 2020

Addis Ababa, Ethiopia

The undersigned have examined the thesis entitled '**Presentation of CPM Using Matrix Schedule for High Rise Building Construction** presented by **Tesfa Fentaw**, a candidate for the degree of **Master of Science** and hereby certify that it is worthy of acceptance.

Dr. Abraham Assefa

Advisor	Signature	Date

Professor Abebe Dinku (Dr. Ing.)

Internal Examiner	Signature	Date

Dr. Girmay Kahssay

External Examiner	Signature	Date

Chairperson	Signature	Date

DECLARATION

I hereby declare that this study submission is my own work towards the Master of Science in Civil Engineering (Construction Technology and Management), and that to the best of my knowledge, contains no material previously published by any person nor material which has been accepted for the award of any other degree of the university, except where due acknowledgment has been made.

Tesfa Fentaw

ABSTRACT

Due to the increasing density of population and high cost of land in urban areas, especially in Addis Ababa, the architecture of commercial and residential buildings is evolving towards high rise structures. However, several studies show that many of these projects suffer from delay and cost overrun.

Planning and control of time are essential for successful completion of such high rise building construction projects. Bar charts together with network techniques are commonly used for communicating project schedules, however, such methods produce a huge complex network difficult to visualize and understand for the majority of construction personnel. This also discourages execution personnel from updating and using the schedule for progress monitoring.

The purpose of this research is thus to devise a scheduling system for high rise building construction by combining the advantages of critical path method and matrix scheduling technique and making some improvements to these methods. Matrix schedules are an efficient way to present information regarding repetitive projects but lack the analytic qualities of the critical path method. A study of planning and scheduling was performed and a system was developed and implemented as a software application.

The developed software was tested through three actual high rise building projects. The system adds a spatial dimension to the scheduling process which makes it easy to better present the plan of action and communicate the plan to everyone involved with the project. Other features include correction of duration to account for the influence of bad weather and maintaining continuity of repetitive activities. These features help to produce a realistic schedule for such repetitive projects.

Keywords: Critical path method, Matrix schedule, Planning, Scheduling.

ACKNOWLEDGMENTS

Humble thanks are first offered to God.

I would like to express my deepest gratitude and sincere appreciation to Dr. Abraham Assefa, my advisor, for his valuable advice and encouragement throughout the study. I greatly appreciated his effort and time reviewing this thesis and giving suggestions to improve the content.

Finally, I would like to thank my family and friends for their continuous support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Background

Commercial and residential buildings are becoming higher and higher these days in maximizing land use and investment return. These days, developers are very cautious in their investment, because of the skyrocketing cost of material and interest rate and want their buildings completed as soon as possible. However, several studies show that many construction projects in our country suffer from delay and cost overrun. As a result, international construction companies, leaving our local contractors out of the competition, are undertaking many high-rise buildings.

Planning and scheduling are crucial issues for successful execution of such construction projects. Construction planning includes the determination of construction methods, listing of activities that must be performed, and identify the precedence relationships between different activities, preparing estimates of costs, resources, and durations for various activities to bring about satisfactory completion of the project (Pawar, et al., 2018). Scheduling is the process of determining the sequential order of the planned activities, assigning duration and starting and completion dates to each activity of the work (Pawar, et al., 2018). There are different kinds and varieties of planning and scheduling tools. Some of them are:

- Network scheduling techniques, such as the critical path method (CPM) and project evaluation and review technique (PERT)
- Bar/Gantt chart
- Line of balance (LOB)
- Linear schedules (LS), and
- Matrix schedule

The selection of scheduling techniques depends on the nature of the project. Some methods are more efficient than others, depending on the nature of the project to be scheduled. Scheduling techniques should be clear enough to be understood by people at the field level of operation, who may not have sufficient technical knowledge to understand complex

schedules and complete tool to effectively portray all the information needed to plan and control the operations.

CPM is a powerful planning technique. Most planning software use Gantt chart generated as a result of CPM calculations. However, the presentation of the schedule becomes complicated when applied for high-rise buildings due to numerous repetitive activities.

Matrix schedule consists of rows and columns where each column represents activities. Boxes on the matrix are used to show duration, start and finish dates. Matrix schedule is an efficient way to present information. Even though matrix schedules are effective for documenting and communicating information, they lack analytical abilities for planning.

The purpose of this research is to devise a scheduling system by combining the advantages of CPM and matrix scheduling techniques and making some improvements to these methods.

1.2 Problem Statement

Bar charts and network scheduling methods are currently the most common scheduling methods for many types of projects. However, these methods present complications in projects of repetitive nature such as high-rise building construction. High-rise buildings consist of large number of activities and many of them are similar activities that are repeated on most of the floors. When a bar chart is used for presentation of the schedules of such projects the resulting network is huge and complex which is difficult to manage. It becomes difficult to understand and visualize by majority of the construction personnel. In addition, the schedule does not provide any information on where on the project site the work is currently being performed. They lack graphical output having a spatial dimension that portrays location information. It is difficult for the project manager to find which activity is going on a particular floor when it comes to the construction project that is broken down into a large number of activities, which makes the progress monitoring difficult. When the schedule is simple and easily understandable, it enables project managers, site engineers, subcontractors, supervisors, foremen, and other construction personnel to better visualize the plan of action and easily communicate the plan to team members involved with the project. This leads to improvements in productivity and reduction in cost. The site staff at operation level can involve and give feedback that can help to early anticipate problems, identify reasons for delay and take appropriate corrective

action to rectify the problems. In summary, there is a need for a better scheduling system for high-rise building construction that enables easy project monitoring.

1.3 Research Objective

The objective of this research is to devise a scheduling system based on CPM and matrix schedule. CPM is used for calculations by making some modifications so that continuity of repetitive activities is maintained to have a smooth procession of crews from unit to unit with no conflict and idle time. Adjustment of duration to account for the influence of weather is also included. The Matrix schedule is used for presentation of the plan by improving its graphics and make it simple so that many of execution personnel such as top-level managers, supervisors, and site engineers can all monitor the progress of the construction work easily.

1.4 Research Methodology

The methodology employed to achieve the research objective consisted of an in-depth review of the literature about planning and scheduling techniques. After examining the literature, specifications for the system were developed taking into account the nature of high-rise building construction. Then, a system was developed and implemented as a software application. Java programming language was used to develop the computer prototype.

The developed software was verified and validated through testing during and after its building. Three building projects were used to test the effectiveness of the system to attain the objectives. The projects are selected such that different types of work items and projects are included for testing (i.e. Judgmental sampling is used).

1.5 Significance of the Research

As it's mentioned earlier, bar charts and networks scheduling methods produce complicated network for high-rise building projects, which will discourage the construction personnel to use the schedules for tracking the progress of the work. The research provides an easy scheduling system. A simple and understandable technique can

help contractors in improving productivity and increase the possibility of completing a project on time as it can be used and understood by the majority of construction personnel.

1.6 Scope and Limitations of the Study

The scope of the research is limited to scheduling high-rise building projects. High-rise buildings tend to be repetitive and the system is not proposed for projects that consist of largely non-repetitive activities. Besides, due shortage of time, the number of projects used to test the system is limited to three.

1.7 Thesis Organization

The research is organized into six chapters.

- Chapter One: Introduces the research problem and objective of the study, research methodology, and significance of the research.
- Chapter Two: A review of planning and scheduling techniques
- Chapter Three: Contains details of the proposed system
- Chapter Four: Description of how the system was implemented as a software application and its features
- Chapter Five: Demonstrates validation of the system tested through actual construction projects.
- Chapter Six: Conclusions and recommendations were forwarded based on the findings of the system performance test.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

Construction projects are time-bound and employ huge resources of labor, material, and Equipment. The main goals of any construction project are to complete the project on the stipulated time, within budget, and with the required quality. It requires effective management of resources to achieve these objectives.

Construction planning and scheduling are some of the most important tools for the achievement of these goals in any construction project. Every construction project involves a lot of activities that need to be planned and scheduled properly for the successful completion of the project. This chapter provides a review of project planning, scheduling, and several tools and techniques used for planning and scheduling construction projects.

2.2 Performance of Construction Projects in Ethiopia

Several studies show that many construction projects in our country suffer from delay. Ayalew et al. (2016) asserted that the level of construction project management practice in the Ethiopian construction industry in terms of adapting the standard project management procedure, tools, and techniques to be unsatisfactory. Ayalew et al. (2016) found that schedule slippages were up to 80%, and cost overruns were up to 40%. According to Sinesilassie et al. (2017), the schedule performance of the Ethiopian construction industry is affected by factors such as owner's competence, conflict among project participants, poor human resource management, project manager's ignorance, and lack of knowledge. Werku and Jha (2016) mentioned that only 8.25% of projects they studied have been finished on the planned completion date. According to them the main factors that cause construction delays in Ethiopia are; difficulties in financing projects by the contractor, price escalation, Ineffective project planning and scheduling, delays in progress payments, lack of skilled professionals in the field of construction management in the organization, and fluctuating labor availability season to season. Kuhl and Seifu (2019) also studied the causes of delay in public building projects and identified ten factors that caused the delay in the public building construction projects in Addis Ababa and the first two are difficulty in project financing and poor project management system.

Dessalegne (2017) studied scheduling practices and control of public building projects in Addis Ababa. From the projects he studied, 70% were behind schedule. Regarding their scheduling practices, all of them used Microsoft project software (which is CPM-based) for scheduling and Microsoft Excel for performance evaluation.

Studies identified that poor planning and scheduling practices as one of the factors that cause delay of projects in Ethiopia (Dessalegn, 2017; Werku & Jha, 2016; Kuhil & Seifu, 2019) . Elbeltagi et al. (2016) asserted that successful construction projects are a result of project activities scheduled for optimally integrated activities. Schedule performance is found as one of the critical factors of project success (Hailemarkos, 2020). Project scheduling is among the essential tasks of project management process success factors. It enables resource planning, such as cash, human resources, and materials (Hailemarkos, 2020).

2.3 Project Planning

Planning is the most important aspect of project management. Planning involves defining objectives of the project, listing of tasks or jobs that must be performed, determining gross requirements for material, equipment and manpower, and preparing estimates of costs and durations for the various jobs or activities to bring about the satisfactory completion of the project (Khandelwal & Pumnia, 2006).

Planning thus develops a strategy and defines expected outcomes for undertaking a specific task before committing to such a task. Without planning, there is no way for scheduling and monitoring the progress of projects, and no way of deciding on corrective measures when unforeseen events happen.

2.3.1 Project Planning Steps

Elbeltagi (2012) identified the following steps that may be used as a guideline, or checklist to develop a project plan:

- Define the scope of work, work methodology, and sequence of work.
- Generate the work breakdown structure

- Develop the organization breakdown structure and link it with the work breakdown structure to identify responsibilities. Organization breakdown structure defines the different responsibility levels and their appropriate reporting needs.

- Selection of construction techniques
- Determine the relationship between activities.
- Estimate duration of activities, cost expenditure, and resource requirements.
- Develop the project network.

2.3.2 Work Breakdown Structure (WBS)

Project planning is most effective when the project scope is completely and well defined. The scope of work of a project is generally assembled from a variety of sources including a request for proposal (RFP), a proposal in response to an RFP, the negotiated contract, a discussion between contractor and client, etc. (Khandelwal & Pumnia, 2006).

A WBS, which is a product or product-oriented family tree break down which graphically portrays all of the work necessary to achieve the projects' objective, is the preferred management tool for organizing work into manageable packages of work (Kerzner, 2009). It provides an ordered framework for planning and controlling the work efforts to be performed in achieving technical objectives for summarizing data and for the preparation of the quantitative and narrative reports used for monitoring cost, schedule, and technical performance.

A WBS is developed by first identifying the major end items or systems to be produced, followed by their successive subdivisions into increasingly detailed and manageable subsidiary products (Kerzner, 2009). Detailed task descriptions are often prepared for each product on the WBS at the level where work will be performed. These task descriptions could identify the product to be produced, describe the effort to be performed, identify the resources to be applied, specify the budget and schedule constraints, the technical requirements, and identify the organizational element responsible for work accomplishment. A compilation of these task descriptions along with the WBS makes up the scope of work.

2.3.3 Work package

Once the scope of work is defined and reflected into a WBS, the project selects a level where the project will be managed and controlled (Khandelwal & Pumnia, 2006). This is called a work package level. A work package is a manageable piece of work from the standpoint of scope (not too big and not too small) cost (not too low and too high) duration (not too long and not too short) and is generally performed by a single organizational element (Khandelwal & Pumnia, 2006). WBS constitutes the basic building blocks in planning, controlling, and work performance. A work package should be a natural subdivision of effort planned according to the way the work will be performed and is scoped, in general, to match the organizational setup. The work package serves as a vehicle for monitoring and reporting the progress of work.

2.3.4 Activity Duration Estimation

Activity duration estimation is the process of estimating the number of work periods needed to execute activities with estimated resources and is a major input into the schedule development process. PMI in the PMBOK fifth edition mentions the following tools and techniques for duration estimation.

2.3.4.1 Expert Judgment

Activity duration can be estimated by someone experienced or familiar with the type of work involved.

2.3.4.2 Analogous Estimating

In this technique, the duration of an activity is estimated using historical data from a similar activity. When estimating durations, this technique relies on the actual duration of previous, similar projects as the basis for estimating the duration of the current activity.

2.3.4.3 Parametric Estimating

Parametric estimating is an estimating technique in which an algorithm is used to calculate cost or duration based on historical data and project parameters.

2.3.4.4 Three-point Estimating

This method is used when the exact duration of an activity, like research and development, is not certain. In this technique, the three times (Most likely, optimistic, and Pessimistic

times) are used to compute expected duration (an approximate range for an activity's duration) is determined.

2.3.5 Time Contingency

When estimating Activity durations, the planning team must address unknowns or faulty assumptions. This may be accomplished by defining "contingency reserves" for the amount of time that needs to be added to the project to account for risk. One way of scheduling for unknown problems is to add a fixed percentage to the schedule as a block of time or to add a contingency percentage to each of the tasks (Sears et al., 2015). Probably the most common example of project delay is that caused by bad weather such as heavy rain, snow, extreme temperatures, and high winds. There can be a profound difference in the time required for excavation, depending on whether the work is to be accomplished during dry or wet weather. Allowances for time lost because of bad weather depends on the type of work involved. Highway and utility projects are is shut down due to bad weather. Buildings can be protected from the weather and therefore, allowances for time lost are usually applied only to activities exposed to weather such as structural and exterior works. It is rare that a work of this type is completely shut down by bad weather after it is enclosed. Some activities may be interrupted while others can proceed. Activity calendars can be used to account for bad weather (Sears et al., 2015).

2.3.6 Resource Allocation

The term resource refers to manpower, machines, money, space, materials, and subcontractors that are required for completing various activities of a project. The network analysis for the critical path method is valid only if the availability of resources is unlimited (Khandelwal & Pumnia, 2006). But all the necessary resources are not available in unlimited quantities. Availability of some of the resources may be restricted. Availability of manpower (supervisory staff, technical and specialist personnel, skilled and unskilled labor, etc.), materials, funds, credits, capital investment, and heavy equipment may be restricted.

The various activities of the project are to be scheduled in such a way that the demand for various resources is more or less uniform throughout the project duration (Khandelwal & Pumnia, 2006; Naylor, 1995). Large fluctuations in their demand may cause problems in the project execution.

For a given network, the requirements of various resources are determined using the early start schedule of each activity. In a network, various activities are involved, and each activity requires some resources to perform it. There may be activities that are to be performed simultaneously and may require common resources. The requirement of resources to execute these simultaneous activities may exceed the available resources. However, at some other period of the execution of the same project, there may be very few activities that may require these resources. Hence the requirement of a particular resource may not be uniform during the project duration. Therefore, the planning should be done in such a manner that resources are utilized in a more or less uniform manner (Khandelwal & Pumnia, 2006).

2.3.7 Resources leveling

In the resources leveling process, the activities are so rescheduled that maximum or peak resources requirement does not cross the limit of available resources. The available resources should however not be less than the maximum number or quantity required for any activity of the project. In rescheduling, the available floats are first used (Khandelwal & Pumnia, 2006). If by doing so, the resources demand is more than the available resources, the duration of some of the activities is increased so that the resource requirement for these activities is decreased. Thus in the resource leveling process, the project duration, initially planned, might be changed.

2.4 Project Scheduling

Mubarak (2010) defines Scheduling as "……. the determination of the timing and sequence of activities in the project and their assembly to give the project completion time." As per Mubarak (2010), project planning answers the questions of what is going to be done? Who will be doing it? How? Where? And when (starting and completion dates)? Scheduling deals with when on a detailed level as shown in Figure 2-1.

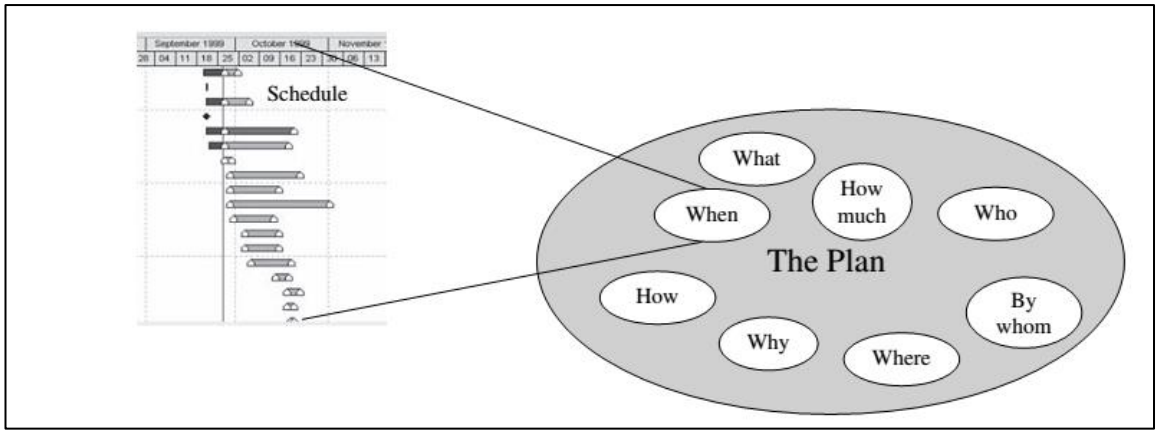


Figure 2-1 Planning and Scheduling (Mubarak, 2010)

The project schedule provides a graphical representation of planned tasks, milestones, the interdependency between different tasks, estimated duration, resources, the name of the person responsible for executing the work, the starting and completion date of each task (Pawar, et al., 2018).

2.5 Significance of Planning and Scheduling

Good Planning and scheduling provide tangible benefits in delivering projects successfully. According to Baldwin and Bordoli (2014), the major advantages include:

- Forecasting resource requirements and costs
- The ability to develop more realistic schedules
- helps to communicate with clear and reliable information to stakeholders
- Provides dependable information for risk assessment
- Provides reliable information for monitoring the work
- Minimizing materials wastage
- Provides a strong basis for coordinating the team members
- assists in the negotiation of contractual claims

2.6 A Review of Planning and Scheduling Techniques

2.6.1 Bar Chart

Bar chart, also known as Gantt chart, was introduced by Henry L. Gantt in 1917 (Subramani & Chinnadurai, 2015). A bar chart consists of two co-ordinates axes, one (usually horizontal axis) representing the time elapsed which may be expressed as months, weeks, or days and the other (the vertical axis) represent the jobs or activities to be performed. Each bar represents one activity. The beginning and end of each bar represent the time of start and finish of that activity; the length of the bar, therefore, represents the time required for the completion of the activity (Figure 2-2).

A bar chart is easy to read and understand. It promotes activity duration visualization. It is an acceptable means of communicating job progress information to technically untrained people, or even to construction professionals whose need to know is limited to progress data only. Bar charts are simple for persons who do not have sufficient technical knowledge. For example, some clients and top-level managers may better understand the project schedule by looking at a bar chart than by looking at the network (Ching, 2005).

When projects become more complex, the interpretation of relationships among activities becomes much more difficult. The traditional bar chart format does not reflect the relationship between various activities which are a common feature of all complex projects. This problem can easily be solved by using a linked bar chart.

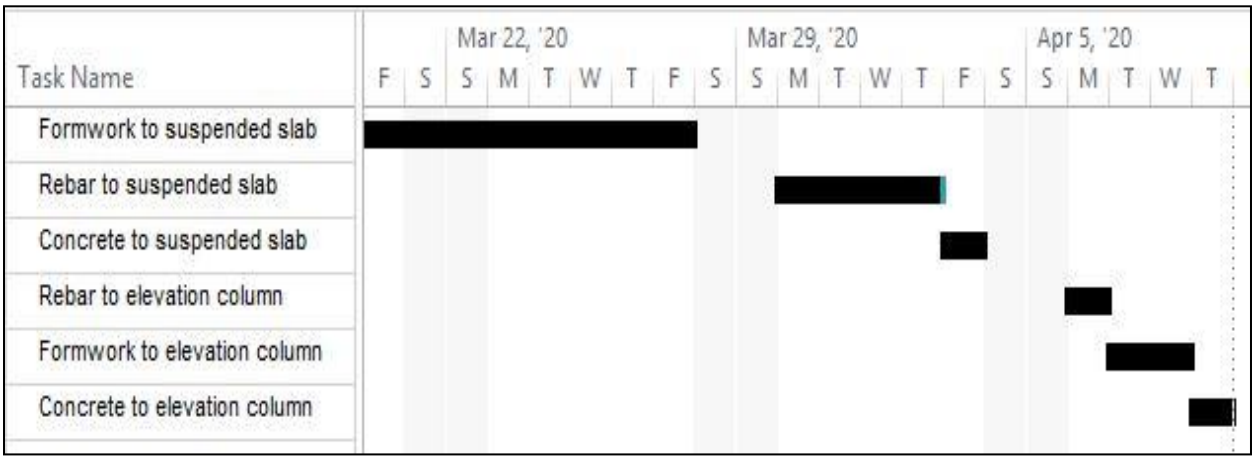


Figure 2-2 Bar chart (created using Microsoft project 2013)

2.6.2 Linked Bar Chart

In a linked bar chart, the end of a preceding activity is linked with a line to the start of the succeeding activity as shown in Figure 2-3. A linked bar chart provides a clear picture of interdependency between different activities. Linking of activities has overcome the main shortcoming of the traditional bar charts.

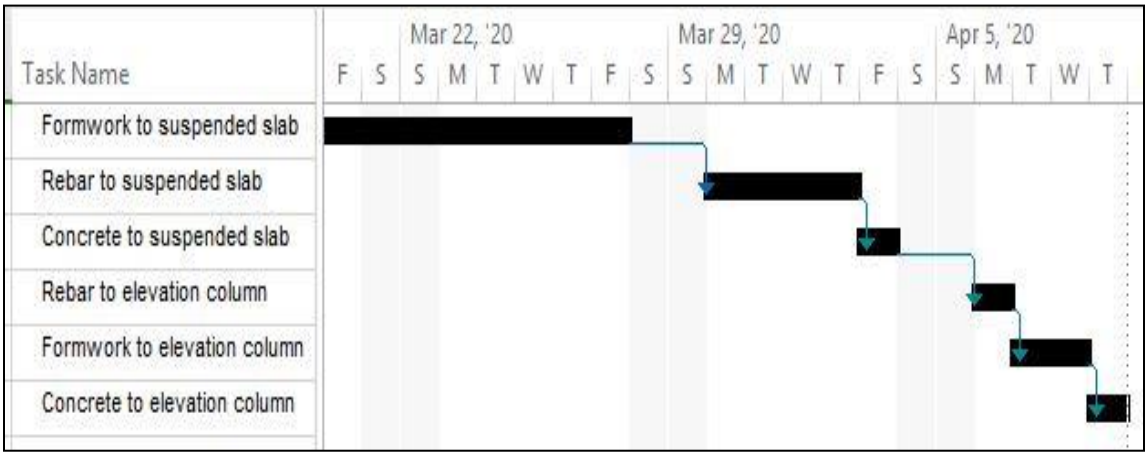


Figure 2-3 Linked bar chart (created using Microsoft project 2013)

Bar chart has several advantages. It is clear, simple, and intelligible to the unskilled. Some clients and upper-level managers may better understand bar charts. No theory or complicated calculations are involved in a bar chart (Ching, 2005). Bar charts are time-scaled, the length of the bar represents the time required for completion of activities which enables activity duration visualization. Bar charts are acceptable for the presentation of the schedule and can be loaded with more information, such as resources, man-hours (Ching, 2005).

Despite these positive features, the chart has several disadvantages, which among others include:

- Although many software programmers have tried to depict logical relationships on bar charts, the result has not always been clear. The relationship lines would get tangled, and unlike network scheduling techniques, the length of the bars can be changed or moved around to make items clearer (Mubarak, 2003).
- When the bar chart is used for linear and repetitive construction projects, a huge diagram would repeat n times in scheduling linear and repetitive projects. Besides, the bar chart is unable to indicate progress rate and actual location (Ching, 2005).

- Its inability to easily display and identify those activities generally referred to as critical activities needing special attention for preventing schedule slippages, time overruns, and other bottlenecks.
- Its inability to effectively handle the complexities inherent in most projects. Even though the bar chart format is most useful for the presentation of schedules, as a planning technique, it is not suitable for scheduling complex projects.

These deficiencies no doubt provided the impetus for increased effort in the search for more efficient planning and scheduling tools. This culminated in the development of network scheduling techniques in the late 1950s.

2.6.3 Network Scheduling Techniques

Network planning techniques were originated during the years 1956 – 1958 to solve two different planning problems. In the first case, the firm DuPont de Nemours was concerned about developing a better scheduling tool in order to improve the construction planning of several chemical plants. Some mathematicians developed the basic relationships for a schedule given the sequence of the work the duration of each activity (Laramee, 1983). In 1957, a special team which included James E. Kelley of Remington Rand, Morgan Walker of DuPont, and later Dr. John W. Mauchly, was formed to find a way of using these principles. They came up with a routine that was called at that time the Kelly - Walker method and which forms the basis for current CPM techniques. Kelly is generally credited with having made improvements to the network representation which made the overall concept practical (Laramee, 1983).

In the second case, in January 1958 the special project office of the Navy Bureau of Ordinance was concerned about monitoring and controlling the Polaris Missile Program (Laramee, 1983). It was however sometime after the project started that this becomes a problem: there were more than 3,000 contractors or agencies working on the program and it was Impossible to monitor them efficiently. They, therefore, established a task group to formulate a technique that became known as (Program Evaluation and Research Task and later Program Evaluation and review technique).

Less than a year after that, the group had developed detailed procedures: the first step was to identify events that are planned to occur along the way to a successful conclusion of the project. or “milestones”; the second step was to link these events to graphically portray the

dependencies among them; and the third element was to estimate times to move from one event to the other, taking into account uncertainty. It was demonstrated by the group that a single estimate was not practical for research and development work. They pointed out that circumstances are too flexible in R and D, unlike construction where time can be estimated with certainty. So three-time estimates were used: a most likely time, an optimistic time, and a pessimistic time. These three times were assumed to be distributed along a bell-shaped curve which, in turn, yielded an estimate of "expected" duration. That duration was then used to perform calculations similar to the CPM mentioned above. It was also possible to evaluate the probability of completing a project within a defined time interval. In this case, the Polaris project finished ahead of schedule with the use of PERT (Laramee, 1983).

A CPM network is a graphic representation of a project. At present time, two formats are used to present the network: the activity on arrow and the activity on node or precedence diagram. The activity on arrow notation was the one used on the first versions of the CPM back in the late 1950s. The precedence diagram was suggested by John Fondahl later in 1961 (Laramee, 1983).

The arrow method of critical path shows activities as an arrow that joins the start and end events of that activity. In this method, it is assumed that the preceding activity must be finished before the following activity can start. This is called a 'finish to start' relationship.

Activity on node network is also called the precedence diagram method. In this method, the nodes or boxes represent activities and the arrows represent logical relationships among the activities (Elbeltagi, 2012). The precedence method has largely replaced the activity on arrow method as the preferred CPM scheduling technique. Although it uses a similar computational approach as the AOA method, it is different in terms of network modeling (Uher, 2003).

A series of connected activities from the starting point to the finishing point is called a path and a project having a lot of activities can have many such paths. Schedules will have at least one critical path. The 'critical path' is defined as one that gives the longest time of completion that also defines the shortest project completion time. Identifying critical activities is important for planning and controlling of a project. Often, a contractor will orient his efforts to keep those activities on schedule and will worry about others if they

encounter long delays and thus become critical. Notwithstanding the importance of their execution in the overall project, he will manage them by exception and give priority to critical activities (Laramee, 1983).

A critical path is calculated using a simple mathematical procedure, which involves calculation of four event times for each activity (Uher, 2003). These are:

- **Earliest Start time**: The earliest date by which an activity can start assuming all the preceding activities are completed as planned.
- **Earliest Finish time**: The earliest date by which an activity can be completed assuming all the preceding activities are completed as planned.
- **Latest Start time**: The latest date an activity must start in order that there is no delay in the project completion.
- **Latest Finish time**: The latest date an activity must be completed to meet the planned project completion date.

When these event times are calculated, activities that have the same earliest and latest earliest start time (or earliest and latest event finish time) are critical. Other activities in a schedule will have different earliest and latest start times (or earliest and latest event finish times and are non-critical. The calculation comprises a forward pass and a backward pass through the network (Uher, 2003).

The 'forward pass' starts from the 'first' node towards the 'finish' node, and calculates the earliest occurrence times of all events. The earliest occurrence time for any node can be calculated from the maximum time taken to reach that node from the different incoming arrows. Once the earliest start time of activity in a schedule is known its earliest finish time is calculated by adding its duration and earliest start time.

The 'backward pass' calculates the latest start and finish time by working from the end of the schedule back to its start (Uher, 2003). In a situation where two or more following activities originate from the same finish event of the preceding activity, the latest finish time of the preceding event activity will be the minimum of the latest start time of the following activities.

In most schedules, only a small number of activities are critical. The remaining non-critical activities are characterized by the availability of float, which is the time by which a non-critical activity can be delayed without delaying the planned completion time of the

project. When a non-critical activity has a float, it may start between the dates given by earliest start date and latest start date event values of that activity.

Float can be considered as a time contingency and helps to reduce the risk delay (Uher, 2003). Float is important to schedule various activities of the project in such a way that the demand for various resources is more or less uniform throughout the project duration. There may be activities, one critical and the other noncritical, that are to be performed simultaneously and may require common resources. The requirement of resources to execute these simultaneous activities may exceed the available resources. The solution is using the available float of the non-critical activity. Since the availability of float in a schedule is advantageous for containing time overruns and better management of limited resources. According to Uher (2003), a party who has a contract to execute the work owns the float. In addition to this, if a variation order by the client causes a loss of float without delaying the project, time extension will not be granted (Uher, 2003).

2.6.3.1 Advantages of Network Scheduling Techniques

The following are advantages of network scheduling techniques.

- The CPM requires careful study of the work logic. The network presentation of the activities enables anybody to see and understand the interdependencies between the activities. There is a clear and shared vision of the logic of the work (Laramee, 1983).
- CPM identifies the longest path (critical path). The CPM calculations determine which activities are critical and which are not. CPM networks are good for showing how progress affects the schedule. If an activity is delayed for any reason, the updated schedule will readily show the downstream effects of the slow progress and if possible, revised logic could be adopted to compensate for the delay (Laramee, 1983).
- CPM networks enable their user to play "what if analysis" games thereby identifying the effects of schedule deviations on the overall project (Laramee, 1983).
- Network Scheduling Techniques can better represent large and complicated projects. They can compute the completion date of the project based on mathematical calculations of the CPM (Ching, 2005; Mubarak, 2003).

2.6.3.2 Limitation of Network Scheduling Methods

Network Scheduling Methods have some Limitations:

- Unlike bar charts, network scheduling is not time-scaled (Kerzner, 2009).
- Requires training to understand the CPM. Its presentation is not as acceptable to execution personnel as bar charts. Some scheduling software vendors have tried to take the advantage of the time-scaled feature of Bar chart and impose it on networks which are called time-scaled logic diagrams (REF). Resource information cannot be loaded (shown on the node or arrows) in CPM.
- CPM-based techniques have been criticized widely in the literature for their inability to model repetitive projects (Hagzey & Kamarah, 2008; Ching, 2005; Arditi et al., 2002).
    - The first problem associated with CPM is the sheer size of the network. In a repetitive project (such as high-rise buildings, highways, utility line projects) of n units, the network prepared for one unit has to be repeated n times and linked to the others. This results in a huge network that is difficult to manage and may cause difficulties in communication among the members of the construction management team.
    - The CPM is designed primarily for optimizing project duration rather than dealing adequately with the special resource constraints of repetitive projects. Its algorithm has no capability that ensures a smooth procession of crews from unit to unit with no conflict and no idle time for workers and equipment. This causes hiring and procurement problems in the flow of labor, material, and equipment during construction (Arditi et al., 2002).
    - CPM is suitable for representing and/or balancing the production rates of repetitive activities. As such, production rate imbalance can negatively affect project performance (Hagzey & Kamarah, 2008).

2.6.4 Line of Balance (LOB)

This Technique was created by the Goodyear Company in the early 1940s and was adopted and developed by the U. S. Navy in the early 1950s for the programming and control of projects involving repetitive tasks (Talodhikar & Pataskar, 2015; Pai et al., 2013). It was first developed for industrial manufacturing and production control. The technique has been applied in construction projects involving repetitive activities such as

road projects, tunnels, pipelines, high-rise buildings, and multi-unit housing developments. In LOB, usually time shown on the horizontal axis while location is shown on the vertical axis as shown in Figure 2-4.

LOB method is oriented towards determining the required rate delivery of completed units and is based on knowledge of how many units must be completed on any day so that the planned delivery of units can be achieved (ElHakeem et al., 2019) The basis of this technique is to calculate the required resources for each stage of work so that the following stages are not interfered with and the target output is attained (Harris & McCaffer, 1989).

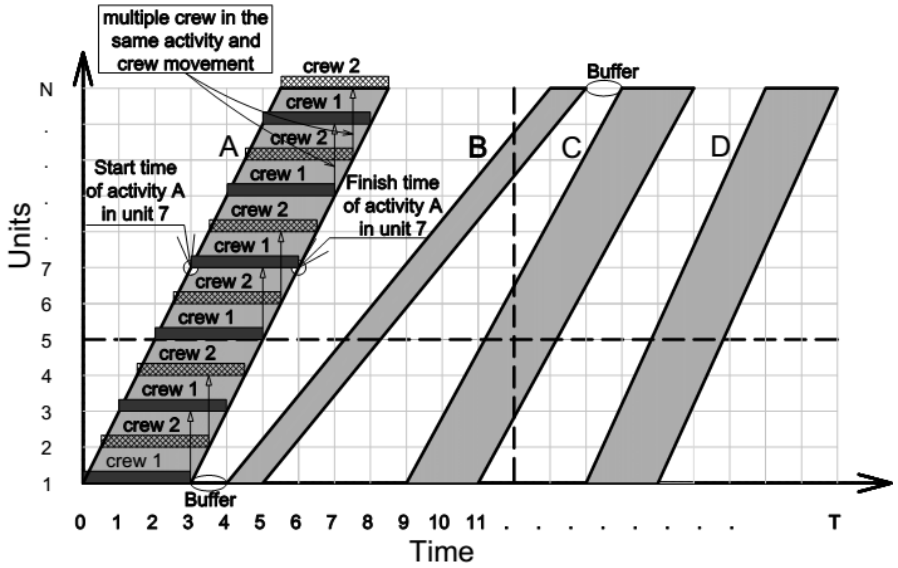


Figure 2-4 Line of balance method (Ammar, 2013)

The advantages of the LOB technique identified by many researchers are explained as follows.

- Production rate and duration of repetitive activity are shown in an easily interpreted graphical format (Yang & Ioannou, 2004).
- LOB allows the balancing of operations (activities) such that each activity is continuously performed from one unit to the other (Hegazy, 2002).
- The LOB plot can show at a glance the progress speed of activities and allows the possibility to adjust the rates to meet production targets while maintaining the work continuity of resources (Hagzey & Kamarah, 2008).
- It enables project managers to see, in the middle of a project, whether they can meet the schedule if they continue working as they have been (Hagzey & Kamarah, 2008).

- It helps to avoid hiring and procurement problems in issues pertaining to the flow of labor and material used during construction. The LOB has the ability to ensure a smooth movement of crews from unit to unit with minimal conflicts and idle time for workers and equipment (Pai et al., 2013).

Even though LOB is a powerful tool, it has some limitations. These are:

- Difficulty of showing all the information on one chart for projects with a large number of operations, especially when monitoring progress (Arditi & Albulak, 1986).
- The LOB method can't generate a clear critical path of the project schedule, relative to the one provided by CPM schedules.
- The LOB technique is criticized for not showing exact relationships between individual activities in the same way as a Bar Chart does.
- LOB can only be divided by location while in CPM the user could divide project by location and other systems like trades (Pai et al., 2013).
- When applying the LOB method to high-rise building construction, it is difficult to show all the activities on one chart, especially in cases where concurrent activities exist (Arditi et al., 2002).
- The LOB is based on assumption that the rate of progress of an activity or package is uniform or the production rate of activity is linear where time is plotted on one axis, usually horizontal, and units or stages of an activity on the vertical axis (Pai et al., 2013; Talodhikar & Pataskar, 2015). However, the rate of production of an activity may not remain constant over the project duration. Using one production rate or an average production rate would lead to erroneous estimations. In high-rise buildings projects, cycle times of work increase from floor to floor as the structure rises, due to the extra time needed to lift materials and because of the increasing intensity of wind (Uher, 2003).

2.7 Matrix Schedule

Matrix scheduling is fairly simple. It consists of rows and columns. Rows represent locations or floors of the building. Each column represents activities. Cells or boxes on the matrix are used to show information such as duration, start and finish dates for an activity at the corresponding floor as shown in Figure 2-5.

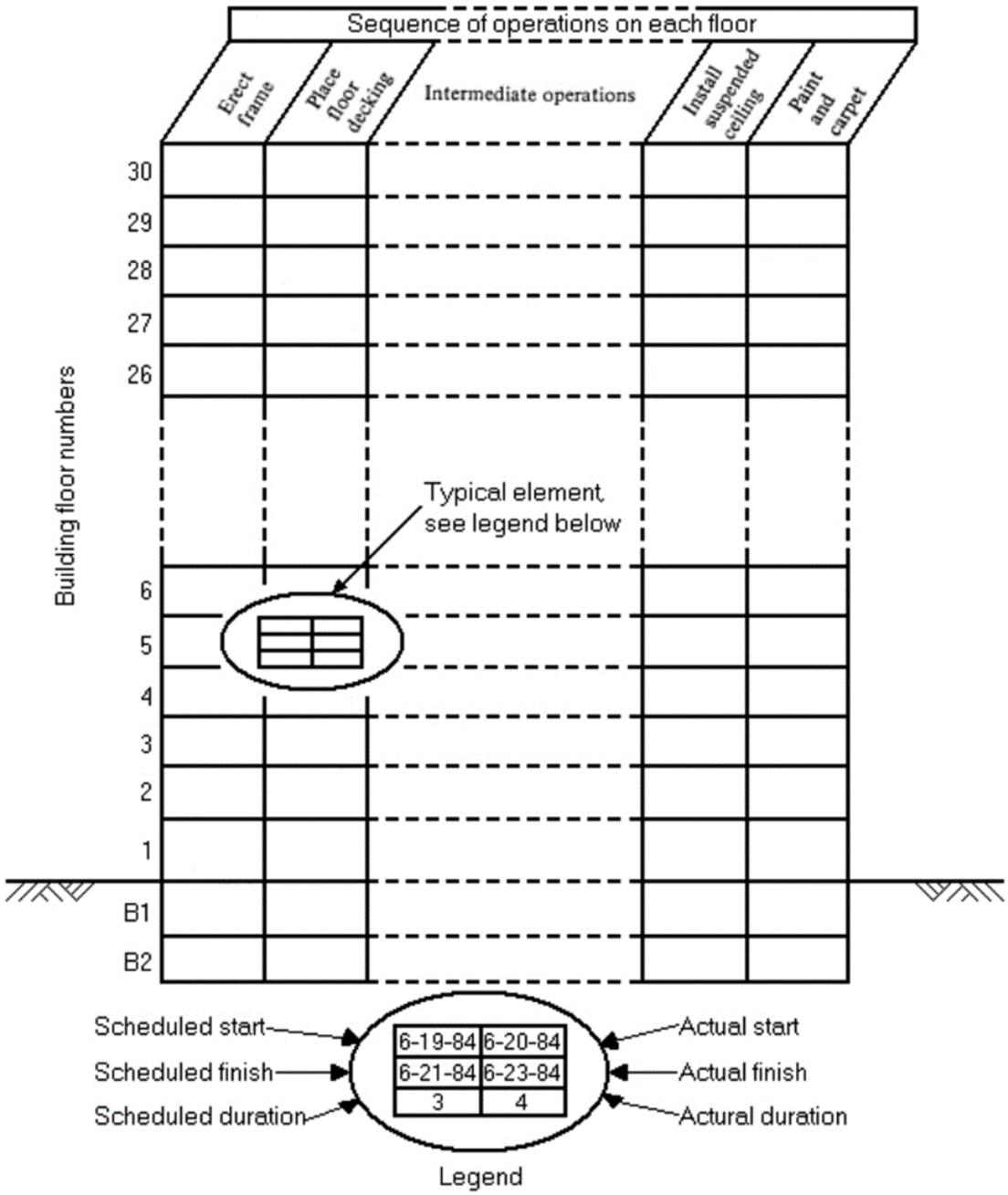
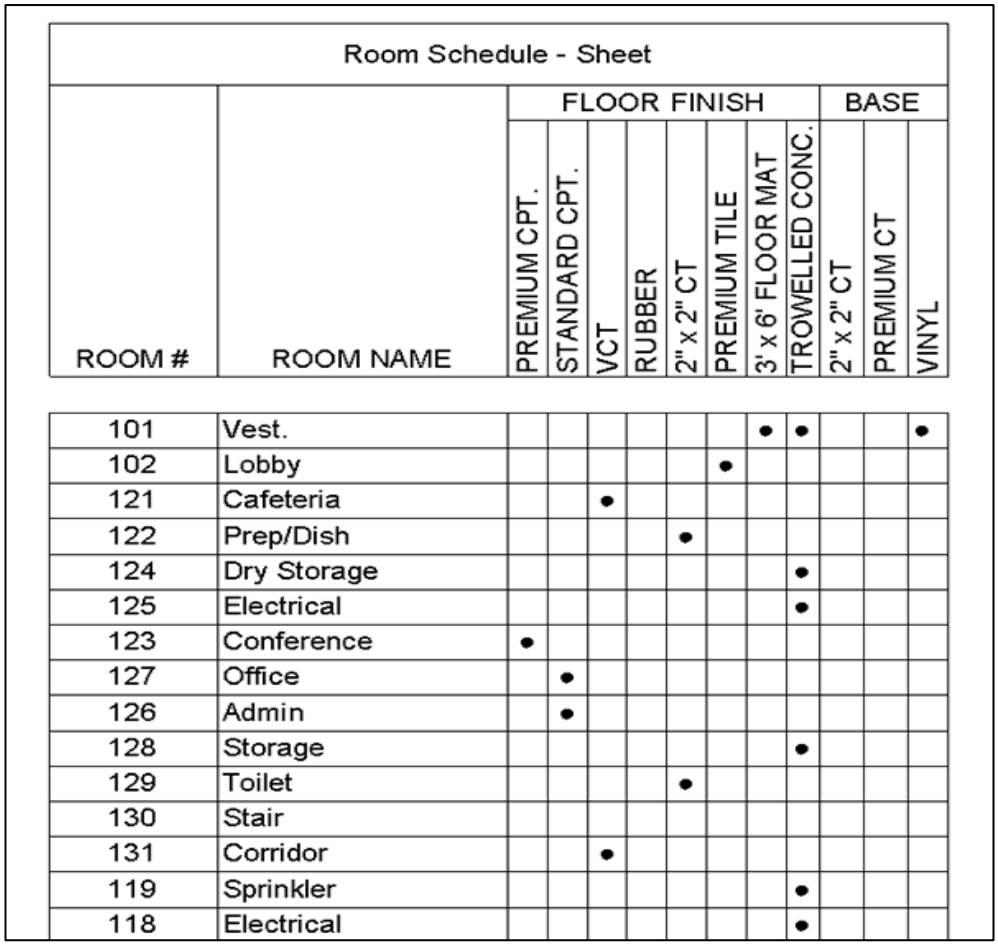


Figure 2-5 Matrix schedule for high rise building (Tsehaye, 2017)

Matrix schedules can be used for different purposes. Figure 2-6 shows a matrix schedule for room finishes with symbols (dotes) on cells. Kilkelly (2016) states that this method is very clear and easy to understand.



Room Schedule - Sheet

ROOM #	ROOM NAME	FLOOR FINISH								BASE		
		PREMIUM CPT.	STANDARD CPT.	VCT	RUBBER	2" x 2" CT	PREMIUM TILE	3' x 6' FLOOR MAT	TROWELLED CONC.	2" x 2" CT	PREMIUM CT	VINYL
101	Vest.							•	•			•
102	Lobby						•					
121	Cafeteria			•								
122	Prep/Dish					•						
124	Dry Storage								•			
125	Electrical								•			
123	Conference	•										
127	Office		•									
126	Admin		•									
128	Storage								•			
129	Toilet					•						
130	Stair											
131	Corridor			•								
119	Sprinkler								•			
118	Electrical								•			

Figure 2-6 Matrix schedule showing room finishes (Kilkelly, 2016)

This technique is also used for high-rise building construction. Figure 2-7shows a manual schedule for high-rise buildings developed using an Excel spreadsheet (Hagzey & Kamarah, 2008). This schedule is simple and easy to read but criticized for its lack of analytic qualities planning.

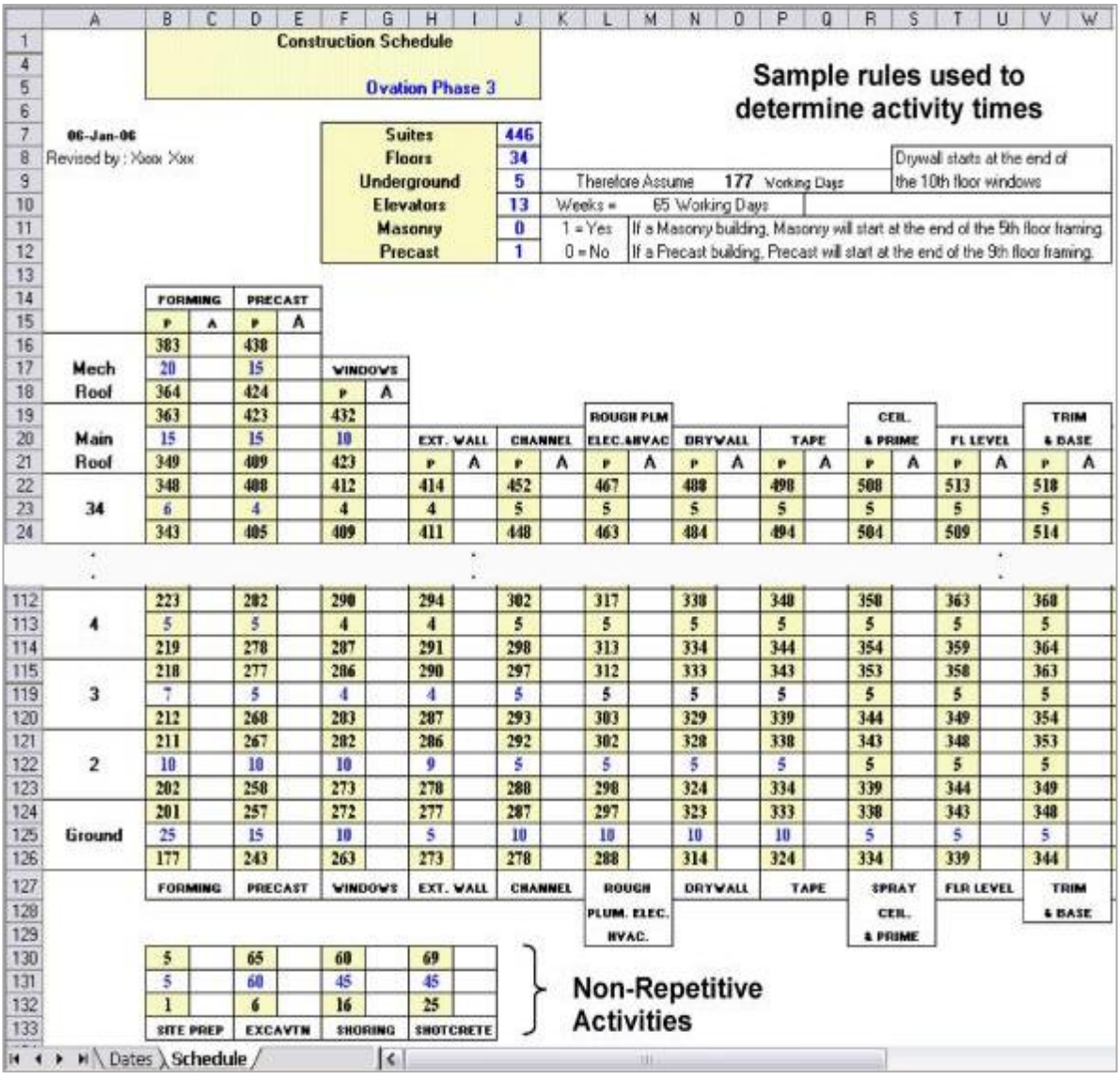


Figure 2-7 A manual high-rise building schedule (Hagzey & Kamarah, 2008)

Matrix schedules are efficient to present information. Activity information can be quickly found for each activity on a given floor (Laramee, 1983). Activity information displayed can be scheduled start and finish dates, actual start and finish dates, duration, etc. In addition, activities can be ordered chronologically from left to right. Chronological left to right flow of activities shows logical interrelationships among activities more obviously than in bar chart. Vertical columns can be made to correspond to specialty subcontractors which enables subcontractors to see their work clearly and the relationship with other subcontractors' work.

Even though matrix schedules are easy presentation formats, they don't have a clear way of showing interactions between different tasks especially for complex projects where there are large number non-repetitive of activities.

2.8 Practices and Studies on Scheduling High-Rise Building Projects

There are a lot of general-purpose software used worldwide for planning and scheduling any type of projects such as Microsoft Project, Primavera P6 and P3, Merlin, FastTrack Schedule, Projector, etc. These software develop CPM schedule and use Gantt chart for presentation of the plan. In Ethiopia, some of these commercial software are used. Dessalegne (2017) studied scheduling practices and control of public building projects. Regarding their scheduling practices, Microsoft project software was used for scheduling and Microsoft Excel for performance evaluation reports. Pawa et al. (2018) also studied planning and scheduling of high-rise building using Primavera software. Microsoft Project and Primavera software present schedules using Gantt chart which has been generated as a result of CPM calculations. Network diagram is also used for presentation of the schedule in these programs.

High-rise buildings are characterized by similar shapes and arrangements for each of the floors. They have a large degree of repetition. These projects involve repetitive activities that advance within the building not in one direction but in two directions: A horizontal direction through the floor, and a vertical direction from one floor to the next (Hagzey & Kamarah, 2008). This makes these projects different from other projects. These projects have scheduling needs that are different from other construction projects. Several studies conducted on scheduling these projects are discussed below.

Thabet and Beliveau (1997) proposed a knowledge-based system for horizontal and vertical logic scheduling (HVLS). They described a structured procedure to incorporate horizontal and vertical constraints to schedule repetitive work in high-rise building projects. In 1997, Thabet and Beliveau incorporated HVLS into SCaRC, a space-constrained resource-constrained scheduling system for high-rise projects. The system incorporates space-based scheduling techniques to consider the problem of limited space availability during the generation of the schedules. The system also incorporates scheduling procedures to account for resource constraints, as well as any defined

horizontal and vertical logic constraints, associated with the scheduling of high-rise buildings.

Arditi et al. (2002) developed a computerized high-rise integrated scheduling system (CHRISS) using line-of-balance technology assisted by an expert system. It integrates databases of the resource productivities with a knowledge-based expert system to generate schedules for high-rise building projects.

Hegazy and Kamarah (2008) developed a high-rise scheduling model (HRSM) which is primarily a scheduling tool implemented using the VBA language of Microsoft Project software. The scheduling formulation of HRSM involves five main aspects:

- CPM calculations for a single floor
- Defining the building's structural-core activities
- Crew synchronization to meet the project deadline
- Consideration of vertical constraints and
- Introducing work interruption.

HRSM calculates the number of crews needed to meet a given deadline, under the following constraints: logical relationships within each floor horizontal constraints, logical relationships among floors vertical constraints, and work continuity constraints. Figure 2-8 shows the schedule prepared by HRSM.

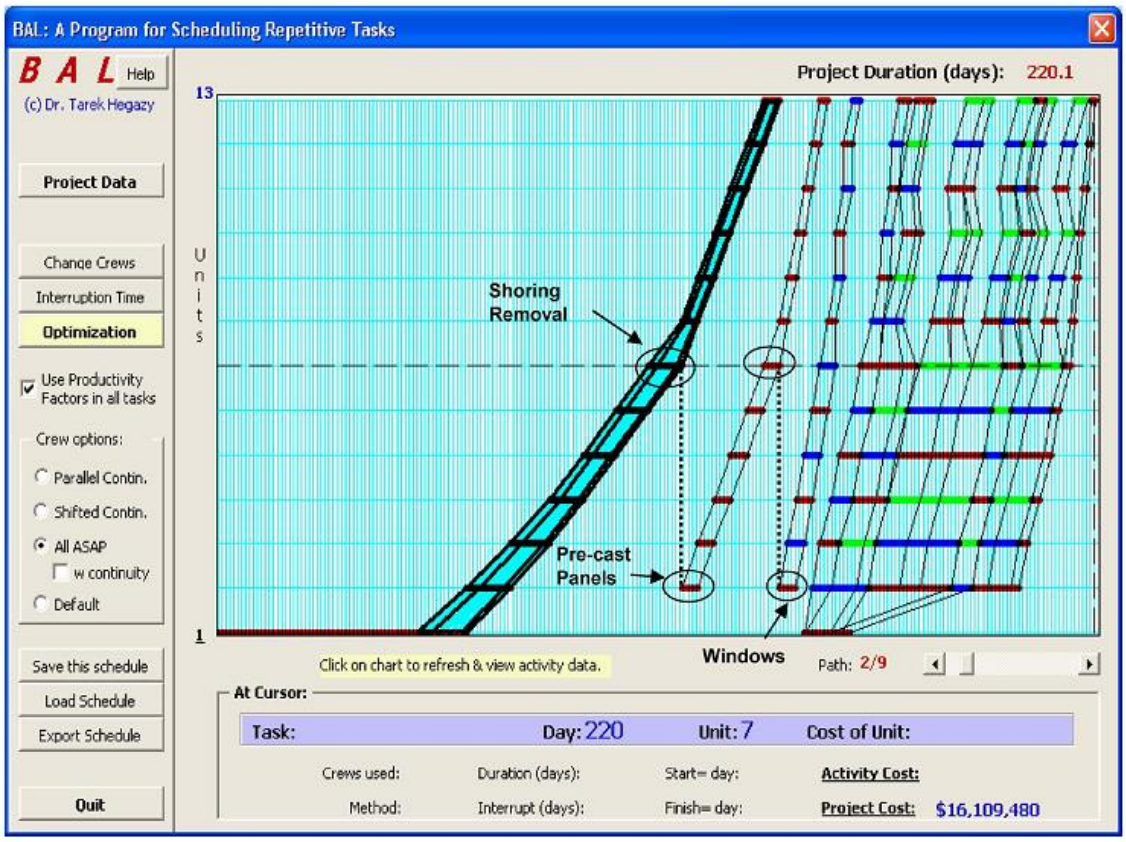


Figure 2-8 High rise building schedule (Hagzey & Kamarah, 2008)

These studies focus on developing a suitable planning system for high-rise building projects but the presentation of the schedules is not given emphasis. It is the people that perform the work, make decisions and take corrective actions, thus simple and understandable presentation of the plan will enable construction personnel to better visualize and understand the plan of action and easily communicate to everyone involved with the project. Regarding the presentation of the schedule, bar chart (Gantt chart), precedence diagram, and line of balance are used in the studies discussed above. However, these techniques present complications for communicating the schedule for high-rise building projects. This is due to the fact that, high-rise buildings comprise a lot of similar activities that repeat on every floor resulting in a large number of activities. When bar chart is used for high-rise buildings, the resulting network will be huge due large number of activities. The relationship lines also would get tangled and the result is not always clear. The same problem exists while using precedence diagrams too. When Line of balance is also used for presenting high-rise building schedules, it is difficult to show clearly all the information on one chart, especially when monitoring progress. In addition, this method

can't generate a clear critical path of the project schedule and doesn't showing exact relationships between individual activities.

2.9 Summary of Literature Review

The main goals of any construction project are to complete the project on the stipulated time, within budget, and with the required quality. Construction planning and scheduling are some of the most important tools for the achievement of these goals in any construction project. Several studies show that many construction projects in our country including high rise building projects suffer from delay (Ayalew et al.,2016, 2016; Werku & Jha, 2016; Wubishet, 2004; Sinesilassie et al., 2017; Hailemarkos, 2020; Kuhil & Seifu, 2019; Tebeje, 2016). Poor planning and scheduling practices are identified as one factor that causes the delay of the projects (Werku & Jha, 2016; Kuhil & Seifu, 2019; Dessalegn, 2017). As per the review, mostly schedules are not used for progress monitoring, are not updated frequently, and are not even prepared properly.

There are several tools and techniques used for planning and scheduling construction projects such as network scheduling techniques, line of balance, bar chart, and matrix schedule. These methods have their own advantages and disadvantages. LOB shows production rate in an easily interpreted graphical format and allows balancing and continuity of operations (activities). However, this method doesn't show exact relationships between activities, can't generate a clear critical path, and difficult to show all information on one chart for projects with a large number of operations (such as high-rise buildings), especially when monitoring progress. CPM can estimate the completion date of the project, determine which activities are critical, can better represent large and complicated projects, and good for showing how progress affects the schedule, but CPM is not suitable for representing and/or balancing the production rates of repetitive activities. Matrix schedules are a simple and efficient way to present information and they correlate locations with activities. However, these method lacks computational base for determining start and finish date of activities, identify critical activities and other qualities of CPM.

Several studies are done on scheduling high-rise buildings (Arditi et al., 2002; Pawar, et al., 2018; Hagzey & Kamarah, 2008; Subramani & Chinnadurai, 2015; Laramee, 1983). *However, these studies focus on developing a suitable planning system for high-rise building projects and the presentation of the schedules for these projects is not given*

emphasis. A study indicates that contractors in our country mostly use programs such as MS Project for scheduling and MS Excel for performance evaluation reports (Dessalegn, 2017). High-rise building projects may comprise thousands of activities in which most of them are similar activities that repeat on many of the floors. Bar chat, network scheduling techniques, and line of balance are used for presentation of the schedules in the reviewed studies. Even though, bar charts are simple for the presentation of schedules, when they are used for such projects a huge complicated network will be produced due to large number of activities. Logical relationship lines on bar charts would get tangled and the result will not always be clear. Such complicated huge networks are difficult to manage and may cause difficulties in communication among the members of the construction management team. Similar problem is seen while using line of balance as presentation tool. Due to large number of repetitive activities in high-rise building, it is illegible to show all activities in one LOB chart. In addition, this method doesn't identify critical activities.

CHAPTER 3 DESIGN OF THE SYSTEM

3.1 Introduction

As it is mentioned in the previous chapters, the most commonly used planning and scheduling techniques are network techniques with bar charts. Most commercial scheduling software presents the schedules using the Gantt chart that have been formulated as a result of CPM calculations. However, high-rise buildings consist of a lot of activities that repeat on every floor. When these techniques are used for presenting schedules of such projects, the resulting network will be so huge and complex that composed of copies of similar repetitive tasks and will not serve the purpose of a good communication tool among construction team members.

The proposed system generates a matrix schedule based on CPM calculations. All the analytic features of CPM are preserved. Additional features included are maintaining continuity of repetitive activities and correction of the duration of activities to account for the impact of bad weather. Resource leveling is also included in this software. The next section consists of an enumeration of the major features of the scheduling system.

3.2 Major Features of the Proposed Scheduling System

The system uses the critical path method for calculations of the early and late dates, project duration, and identification of critical activities. All of the four types of relationships between activities are supported by the system. As it is discussed in chapter two, the calculation involves a forward and a backward pass through the schedule. In the forward pass (starting from the first node towards the finish node), the early start and early finish times for each activity are calculated. Early start time of an activity is the maximum early finish time from immediate predecessor(s). Then, early finish time is calculated by adding duration of activity from its early start time.

The backward pass calculates the latest start and finish time by working from the end of the schedule back to its start. Late finish time of an activity is the minimum latest time from its immediate Successor(s). Late start time can be computed by subtracting the duration of an activity from its latest finish time.

The procedures followed while developing schedules using the system include:

- Develop work break down structure
- Identify all of the activities to be performed
- Estimate durations for each activity
- Identifying dependencies between activities so that they are scheduled in the correct sequence
- Estimate quantities, resource requirements, and cost of completing each activity
- Input project information such as project start date and calendar
- Create a calendar and/or input non-working days
- Input units (floors) into the system
- Input list of resources
- Input activity data
    - Activity name
    - Duration
    - Duration adjustment factors (weather factors) for activities affected by weather
    - Quantity
    - Cost
    - Unit (floor)
    - start and/or finish dates for manually scheduled Activities
    - Required to be continuous or not: if a repetitive activity is required to be continuous then this needs to be entered into the system.
    - Input precedence relationships
    - Allocate resources for all activities
- CPM calculations are done to determine the late and early dates of activities.
- Level resource if resource requirements exceed resource limits.
- Revise the schedule if project objectives are not meet (such as project completion date).

- Use the matrix schedule for presenting and communicating the schedule and progress of the project.
- Updating the schedule

In the system, additional features are incorporated in the CPM calculations to maintain continuity of repetitive activities and correction of duration to account for the influence of the weather.

3.2.1 Continuity of Repetitive Activities

Most high-rise buildings are characterized by similar shapes and arrangements for each of the floors. We have to differentiate between repetitive activities that are found on all or some of the typical floors and non-repetitive activities that are specific only to one area of the building.

The continuity of repetitive activities is very important. Because each repetitive activity will usually be performed by the same crew on all the floors, and this crew will usually be assigned to that specific work, it is important that it works at a set rhythm or production rate and that it moves to the next floor as soon as the previous one is completed. This will also have some effects on the efficiency of the execution of the work. Maintaining a continuous flow of the work for each of the identifiable crews, especially on repetitive tasks, provides a stimulus for the crews to perform well and allows maximum opportunity to experience learning curve effects on production.

Most network models applied to high-rise construction fail to recognize the importance of continuity, which is not part of the CPM algorithm. Repetitive activities, if scheduled as one activity, their duration will be long which is difficult for controlling. Repetitive activities are broken down for each of the floors, and treated discretely, with precedence relationships with different activities, or with that same activity on a different floor. Consequently, there could be a float between the completion of one floor and the start of the next one, which has to be avoided for continuity. Figure 3-1 shows that repetitive activity "A" is not continuous, float between 1st and 2nd floor and between 3rd floor and 4th floor exists due to the relationship with other activities "B" and "C".

The proposed system will therefore allow for continuity requirements by eliminating float between the floors of repetitive activity.

Procedure for maintaining continuity of repetitive activities

1. Start and finish dates for activities at every floor are calculated: In the system, repetitive activity is considered as a summary (summary activity "A" in Figure 3-1) with all same activities at every floor are entered as its sub-activities. The standard CPM calculations will be performed for sub-activities taking into account the relationship between these sub-activities and other activities as well.
2. Check discontinuity and shift activities: Starting from the beginning date of the summery (repetitive activity), dates on which it is not performed (i.e. discontinuous) are checked. If found, then sub-activities that start before this date are moved forward to eliminate the float but the relationships are also checked and not violated. Figure 3-1 shows that there is a float between the 1st and 2nd floor of activity "A" and can be eliminated by shifting the 1st-floor start by the number of floats. If there is an activity whose predecessor and successor are both from these sub-activities and float is created due to this activity, the discontinuity can't be avoided. As shown in Figure 3-1 the float between the 3rd and 4th floor of repetitive can't be avoided due to the relationship with activity "C".

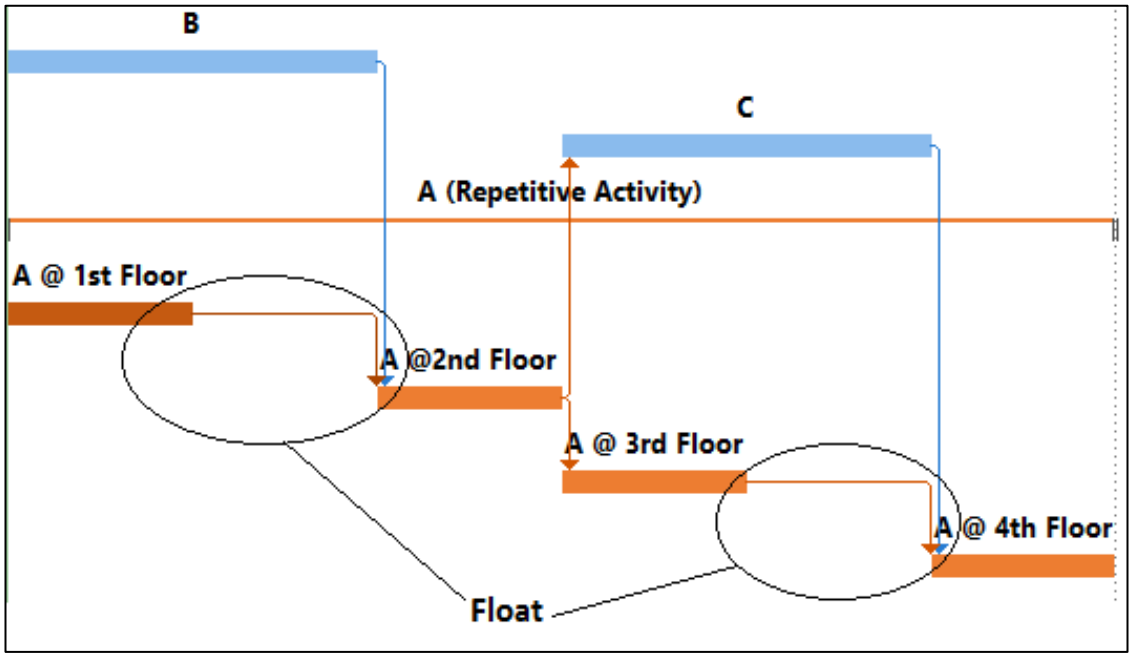


Figure 3-1 Float between the floors of repetitive activity "A"

3.2.2 Weather Factors

The system takes into account the influence of the weather while scheduling. Holidays and weekends are constant throughout the year and are predicted with certainty a long time

before their occurrence. But weather is not and it plays a significant role in delaying the schedule, especially in the wintertime. A contractor should not be blamed for being behind schedule because of unforeseeable weather but he should not take the weather as an excuse. Therefore, a realistic contingency allowance should be added to the schedule, but only for activities affected by the weather. For example, internal finishing works are not affected by the weather whereas the structure and external works are.

In this system efficiency factors for a period (such as winter) are included in the CPM calculations. Efficiency factors are percentages by which duration is divided to give an adjusted duration and are specified for each activity. These factors can be obtained from historical data on the weather for the location of the site. And therefore, the duration of activities that are influenced by bad weather are relaxed by these factors resulting in a realistic schedule. The adjustment procedure followed by the system is shown in Figure 3-2.

Adjustment procedure

1) Weather factors for affected activities for one or more periods of time with bad weather will be determined and entered into the system. Weather factors are simply values between 0 to 1 corresponding to the percentage of efficiency for each month or a specific period of time which are determined from prior data of the location on the percentage of working time lost due to bad weather. Since the weather conditions will not be the same for each month, different weather factors may be used for different months or periods like winter. For instance for June the weather factor maybe 0.8 and for August 0.65. Thus more than one period of time may need adjustment. Different activities may also have different weather factors for the same period.
2) The start and finish date of the activity affected by weather will be calculated before any adjustment
3) Determine adjusted duration and incorporate it in the CPM calculations: Adjusted duration is calculated by dividing the normal duration (duration of the activity in a normal condition) by the weather factor. Different scenarios may be encountered. If the activity is not be executed within the specified periods (Figure 3-3 Activity A), no adjustment will be required. If the activity is executed within one of the periods defined for adjustment partly (Figure 3-3 Activity B & E), part of the duration within the period will be adjusted by dividing it with the weather factor of the period. If the activity is executed within one of the periods totally (Activity

C), the duration is adjusted fully. The activity may spread over more than one period before adjustment (Activity D) or after adjustment (Figure 3-3 Activity C after adjustment). In this case, different factors will be applied to determine its finish date. For example, if the activity starts on July 25 with a duration of 15 days and needs adjustment for two months, July and August with different factors 0.8 and 0.75 respectively, then the determination of adjusted duration will be as follows by assuming all days in these months are working days. In July, 5 working days are available for the activity. This is equivalent to 4 days duration adjusted by July's weather factor (when 4 is divided by June's weather factor 0.8, it becomes 5). But the duration of the activity is 15 and the remaining 11 days are in august and this period also needs adjustment. And the adjusted duration for 11 days normal duration will be computed by dividing it by August's adjustment factor of 0.75 which will be 15 days (when rounded up). This means the activity will have a total of 20 days adjusted duration (5 days in July and 17 days in August). Thus, when considering the influence of weather, the duration of the activity is increased by 5 days. Thus, the time that will be lost due to weather is added to its normal duration which helps to develop a realistic schedule for the project.

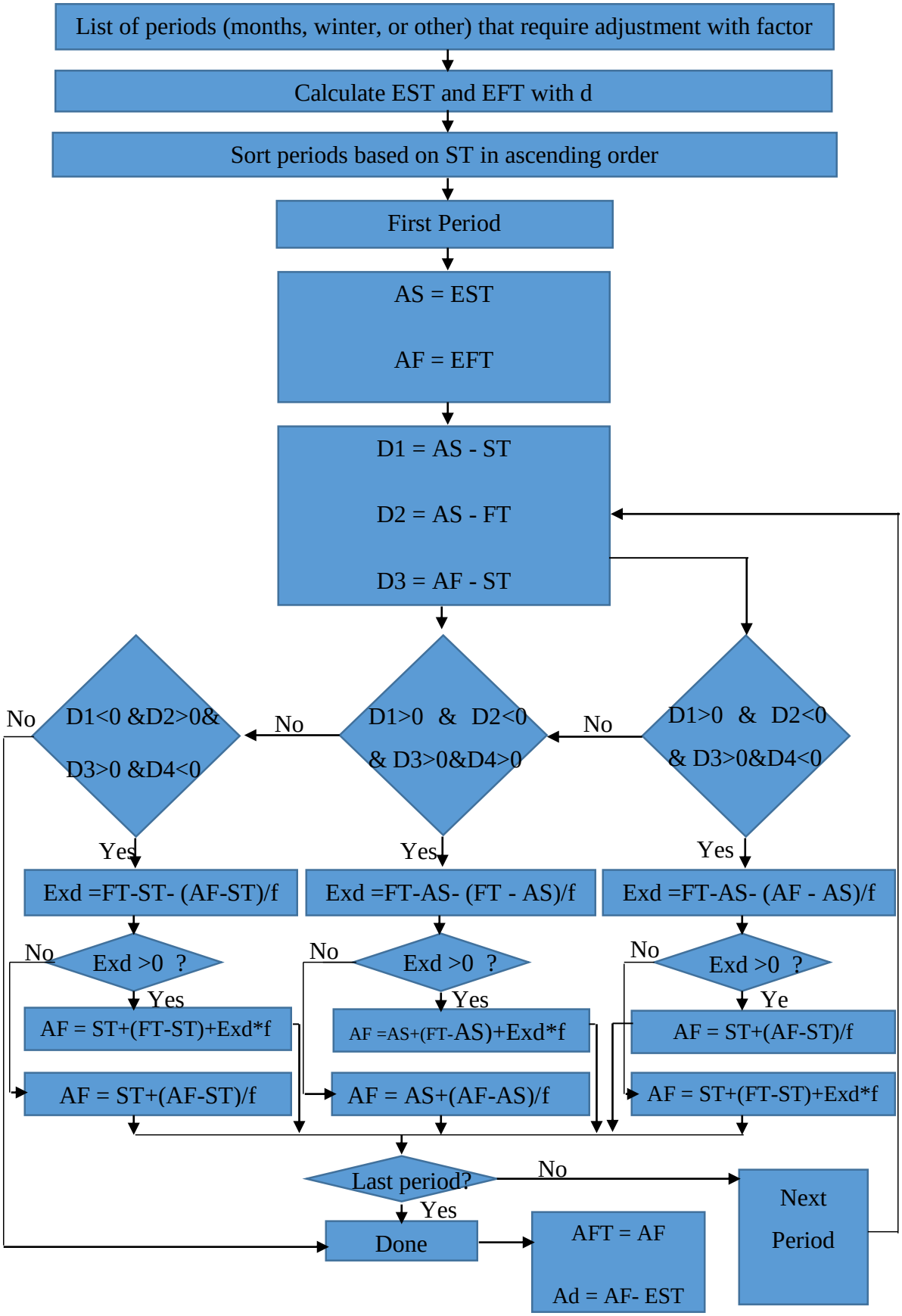


Figure 3-2 Flow chart for weather adjustment

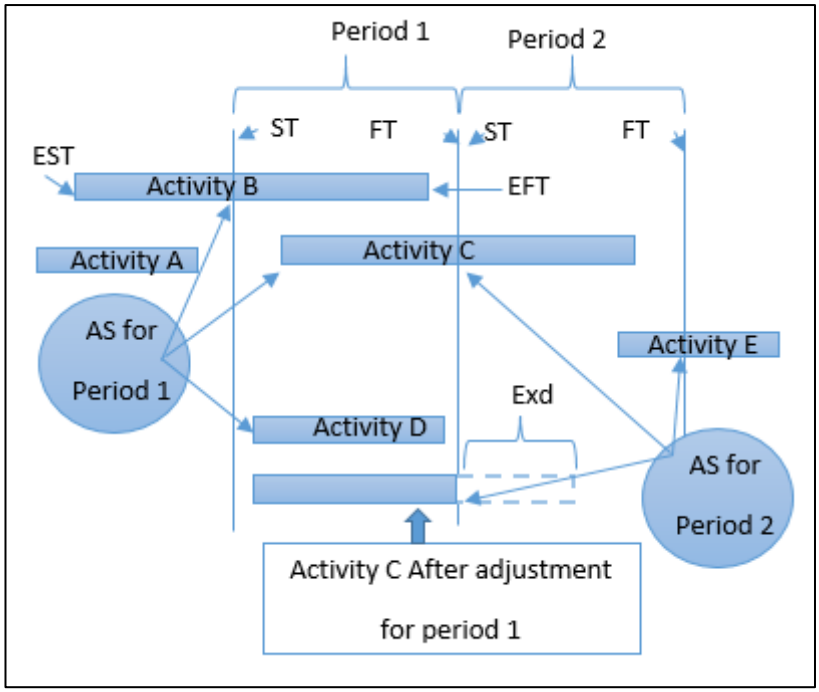


Figure 3-3 Different scenarios in weather contingency

Where

d = normal (unadjusted) duration of activity

EST = early start of activity

EFT = early finish of activity

ST = start of a period

FT = finish of a period

f = weather factor for a period

As = start date for adjustment

AS = adjustment start date of activity

AF = adjustment finish date of activity

Exd = excess duration that extends beyond the end of a period after adjustment

AFT = Adjusted finish time of activity

A_d = adjusted duration of activity

3.2.3 Resources

As it is discussed in chapter two, resources are required for completing various activities of a project. First, the required resources will be estimated and assigned for each activity. However, all the necessary resources may not be available in unlimited quantities. Availability of some of the resources may be restricted. And thus, the requirement of some of the resources may exceed the available resources. Resource leveling is necessary to revolve the over-allocation of resources. The procedures used in this system for leveling are discussed as follows (Figure 3-4).

Procedures followed for resource-leveling

1) A list of resources to be leveled will be selected by the planner along with the leveling option. There are two options for leveling: leveling using total floats only or delaying beyond total float if over allocations are not resolved.
2) Set the date for leveling: the system levels resource day by day starting from the project start date or another specified date. Once over allocations are resolved for a certain date the leveling continues for the next working day.
3) Identify demanding activities for all resources (selected for leveling) on the date of leveling.
4) Check over allocations for all resources: Sum required quantities from all activities for each resource and if this sum exceeds the maximum available unit (quantity) for a resource then the resource is over-allocated.
5) For each over-allocated resource, sort demanding activities based on the number of days the project delays if the start of the activity is changed to the next working date after the date of leveling in ascending order.
6) From the sorted list select the first one or more activities whose start date should be changed to resolve over allocation. The total quantity of resources required on the date of leveling is calculated by the summing requirements by each activity in the list. A commutative required quantity is calculated for all activities in the sorted list (from first activity to last activity). Activities, in the list, before activity at which the deduction of cumulative quantity from the total quantity required is less than or equal to the maximum available quantity of the resource are selected for changing their start dates. If the "level using available total float only" option is chosen, then the selected list will not include critical activities. If all become critical, then the leveling will end even if the over-allocation is not resolved.

7) Identify common activities from selected activities (in step 6) from all over allocated resources
8) The start date of common activities is changed to the next working date after the date of leveling. The total quantity required on the date of leveling is decreased when the start of some of the activities (which will be executed on the date of leveling) is delayed or changed to the next working date. Common activities are given priority because they resolve over allocation for more than one resource.
9) If there are no common activities, then the start of the activities selected from each resource will be changed to a date after the date of leveling as described in step 8.
10) Repeat from step 4 to 8. This iteration will continue until resource over allocation is resolved. However, if the allocated quantity for an activity exceeds the maximum available quantity of the resource, the over-allocation cannot be resolved and is skipped but the system tries to smoothen the allocation at least by decreasing the total required quantity if other activities also require the resource.
11) After over allocations are resolved for the date selected for leveling, the iteration will be started for the next working date until the finish date of the project.

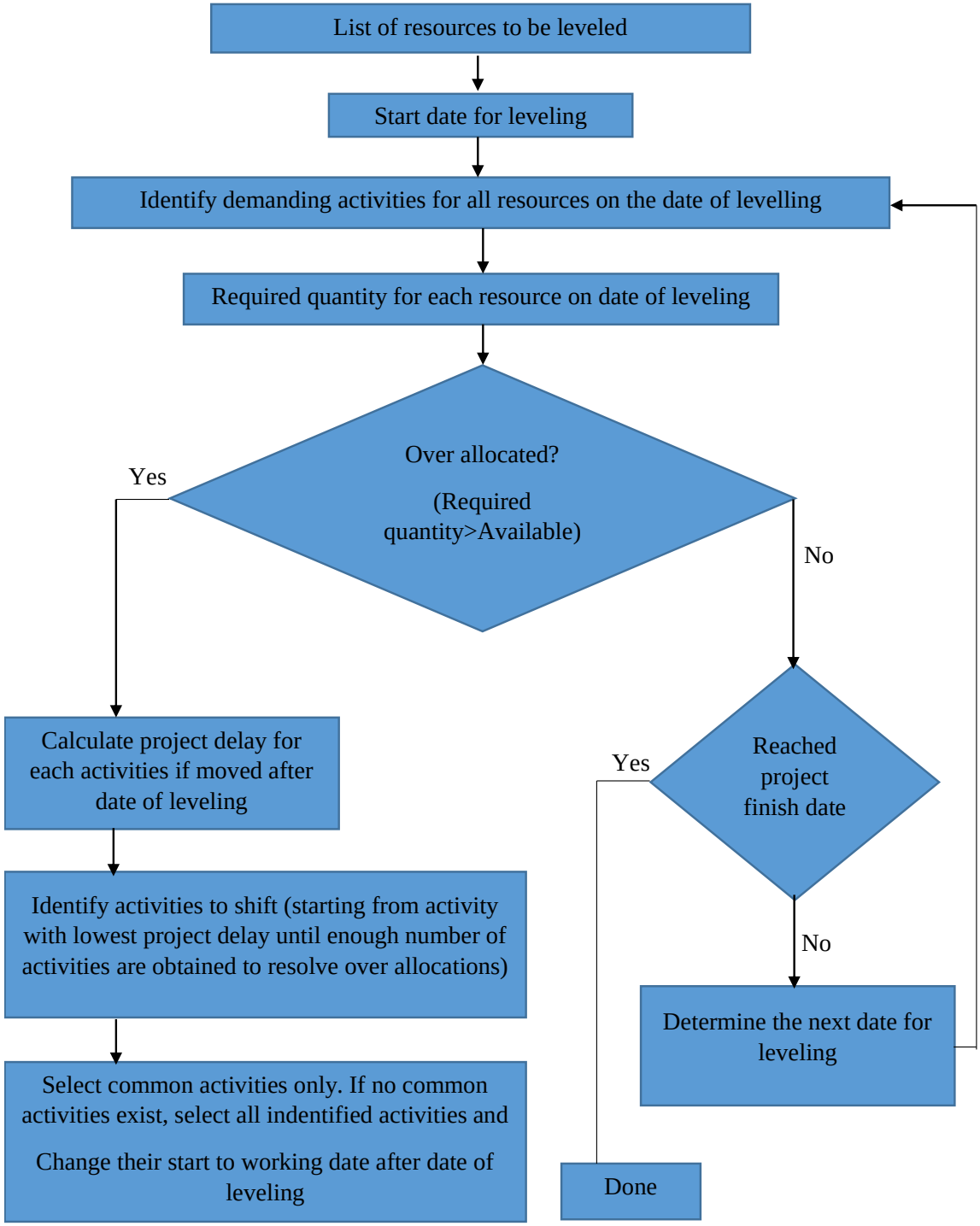


Figure 3-4 Flow chart for resource leveling

3.2.4 Matrix Schedule

Matrix schedule is used for the presentation of the results of calculations of CPM. The usefulness of the presentation of the calculations of any system is the key to its success and acceptance by various users.

As it is mentioned in the literature review matrix consist of rows and columns. Columns represent Activities. Rows represent floors. To generate the matrix schedule floors for each activity should be entered into the system. Cells or boxes are used to show various activity information. The following are some of the features included in this System.

1) Activities will be ordered based on their early start from left to right to help users easily understand the order of execution of activities.
2) In each box, various data are shown which enables one to quickly find the information for each activity on a given floor. The user can choose the information shown in each cell of the box. The information can be the start date, finish dates, actual start date, actual finish dates, duration, percent complete, planned rate of progress, assigned crew or assigned subcontractor, predecessors, successors, etc.
3) Critical activities can be shown by using a box with different borderline colors, line thickness, or fill color from non-critical activities.
4) The planned and actual Progress bars are included in the boxes showing the planned and actual percent complete of that activity at the time of the report. It provides an immediate comparison of the planned and actual progress of the activities on any given floor and enables managers to monitor the activities easily and initiate corrective action if required.
5) In this technique, activities are grouped into summaries and the user can expand and collapse to show or hide sub-activities on the matrix view.
6) It is also possible to zoom out and see the whole data in one screen with lesser details and zoom in for a close up view of the matrix schedule with greater details.
7) By using various colors, different line thicknesses, solid or broken lines, the status of each activity can be shown giving the visual status of the progress of the project. Delayed, completed, and ongoing activities are easily identified with this method.
8) The time of execution of each activity can be shown using different fill colors on nodes (boxes) for different periods such as months. This enables execution personnel to easily visualize the workflow with time and location (floors).

9) Predecessor or successors with the relationship type and lead or lag can be written in the cells.

3.2.5 Updating

Updating the schedule is a very important feature in order for the system to be used on a real construction project. As the work is executed, it is crucial to evaluate if the activities are performed according to the plan and if not, how the time delays or gains affect the remaining activities. Simply stated, updating consists of planning and scheduling the remaining work after some time has elapsed. In this system, updating is done in a sequence of two steps.

The first step is to record the actual start dates, actual finish dates, and actual duration, percent complete, or executed quantity of the activities. This should be done regularly. The second step consists of planning and scheduling the remaining work. If the work progressed as initially scheduled, the remaining work will not have to be scheduled. However, one of the main characteristics of construction work is that it can be affected by several factors, many of which are beyond the control of the contractor (e.g. changes in design, procurement delays, and severe weather). Thus, the schedule should be adjusted in order to reallocate the time lost and take some corrective actions like acceleration. Updating is done by incorporating actual start and finish dates in the calculation but keeping intact the logic and durations of the activities which have not been executed yet. If a predecessor of an activity is completed as indicated by its status, the comparison is not made with the early finish date, but rather with the actual finish date.

CHAPTER 4 COMPUTERIZATION OF THE SYSTEM

4.1 Introduction

The proposed system called HRBSS (High Rise Building Scheduling System) was implemented as a software application. The developed software was verified and validated through testing during and after its building. Even though, the study focus on presentation of schedules, other features such as resource leveling, generating reports, importing and exporting data are included to make HRBSS a complete tool for scheduling. The features of the system are presented in detail in the following sections.

4.2 Software Building

In order to develop the software, Java programming language was used. More than 19,460 lines of code are written and some of them are presented on Appendix I. The steps of software building are presented in the following sub-topics.

4.2.1 Selection of Software Building Environment

Numerous programming languages are available to develop computer programs. In building the proposed software, the present research chose Java, one of the most widely used computer programming languages.

Java is a general-purpose, object-oriented powerful computer programming language that is specifically designed to have as few implementation dependencies as possible (Hazber, 2016). Java lets programmers to "write once, run anywhere", meaning that code that runs on one platform does not need to be recompiled to run on another (Hazber, 2016). According to Oracle, the company that owns Java, Java runs on 3 billion devices worldwide, which makes it one of the top most popular programming languages for the past several years. Generally, java has the following features (Hazber, 2016):

- Platform independent: Java code written for one platform (operating system) and run it on another platform without any modification.
- Object-oriented: Java is an object-oriented language that helps to make our Java code more flexible and reusable.
- Speed: Well-optimized Java code is nearly as fast as lower-level languages like C++.

4.2.2 Software Coding

After the specification and requirements for the system were developed as discussed in the previous chapter, the program was written. There are usually three stages to writing a program:

- **Coding:** Coding is the act of translating the design into an actual program, written in some form of programming language. Text editor, which is somewhat like a word processing program, is usually used to create and save file that contains the program. Once we write the program in the text editor we save it using a filename stored with an extension of .java (Eck, 2011). This file is called source code file. An Integrated Development Environment (IDE) can also be used to create, compile, and run programs (Eck, 2011). For this software, Netbeans IDE is used.
- **Compiling:** The code in a source file stored on the disk must be translated into machine language with a compiler. The Compiler is a computer program that translates the source code. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture (Eck, 2011).
- **Debugging:** debugging means detecting, locating, and correcting bugs (mistakes), usually by running the program.

The program consists of about 40 classes. As shown in Appendix I, one large class named 'HRBProject' represents high-rise building projects. This class has 84 methods (functions), stores a list of activities in an 'Array List', and has 17 member variables. This class has methods that perform CPM calculations by iterating over the list of activities. There is also another class named 'Activity' that represents planned activities. This class has 210 methods and 52 instance variables. It stores start date, finish date, duration, list of predecessors, and other 48 properties for activities. Another class named 'Matrix' which extends 'JPanel' is responsible for drawing the matrix schedule for the activities on their respective row and column. The row is the floor at which the activity is performed which is to be provided by the user while the column is determined by the program based on early start date of activities. To determine the column the activities are ordered by their early start and the first activity gets the first column.

Unit testing and integrated testing were performed and corrections were made until all functional requirements worked properly. Unit testing involves testing the units one by

one in separate testing activities (Hatem et al., 2018). Integration testing is the second level of the software testing process that comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units. Three types of errors were captured in the present software testing, namely, syntax errors, runtime errors, and logic errors. Syntax error (or compiler error) is a mistake (such as a misspelled property or keyword) that violates the programming rules of Java. A runtime error causes the present software to stop unexpectedly during execution. Runtime errors occur when an outside event or an undiscovered syntax error forces the present software to stop in the middle of an operation. Logic error is a human error that causes the program code to produce wrong results (Hatem et al., 2018). The developer performed extensive debugging to track down and eliminate logic errors.

4.2.3 Software validation

Validation is performed to ensure that the software works accurately and gives correct results to attain the objectives (Hatem et al., 2018). Three high-rise building projects were selected, their schedules were prepared using this software and were compared with manual calculations and Microsoft project software results. The results of the software were in line with manual ones and with Microsoft project software results.

4.3 Fundamentals of the System

The graphical user interface of the program is designed to be as simple as possible. The user interface is the interface between the user and software that the user communicates with the software. Features of the system such as creating a project, adding, editing, deleting activities, inputting the WBS for the project, creating calendars, entering non-working days, assigning and leveling resources, viewing matrix schedule, updating schedules, and others will be discussed in the following topics.

4.3.1 Creating a New Project

Starting a new project is done by opening the project information dialog box (Figure 4-1). This dialog box enables you to set a start date for the project, select the calendar to use.

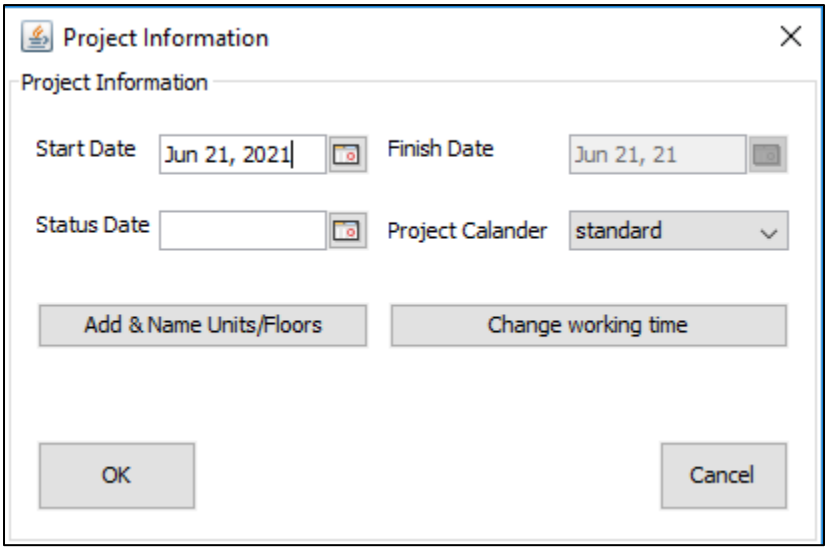


Figure 4-1 Project information dialog box

4.3.2 Inputting Units or Locations

High-rise building is a multi-story structure. Most modern high-rise buildings are characterized by similar shapes and arrangements for each of the floors. This saves both time and cost at the design stage and on the construction site. Architectural plans become much simpler only one floor layout is required for several typical floors. This will lead to repetitive activities that are found on all or some of the typical floors and non-repetitive activities that are specific only to one area of the building. In the proposed system, a project is divided by locations or units such as floors. To input units in the project, the dialog box shown in Figure 4-2 is used. The number of units is first entered and the system automatically generates a list of units with default names (unit 1, unit 2). These default names can be changed to meaningful names. Units also can be deleted alone or with all activities performed on the unit.

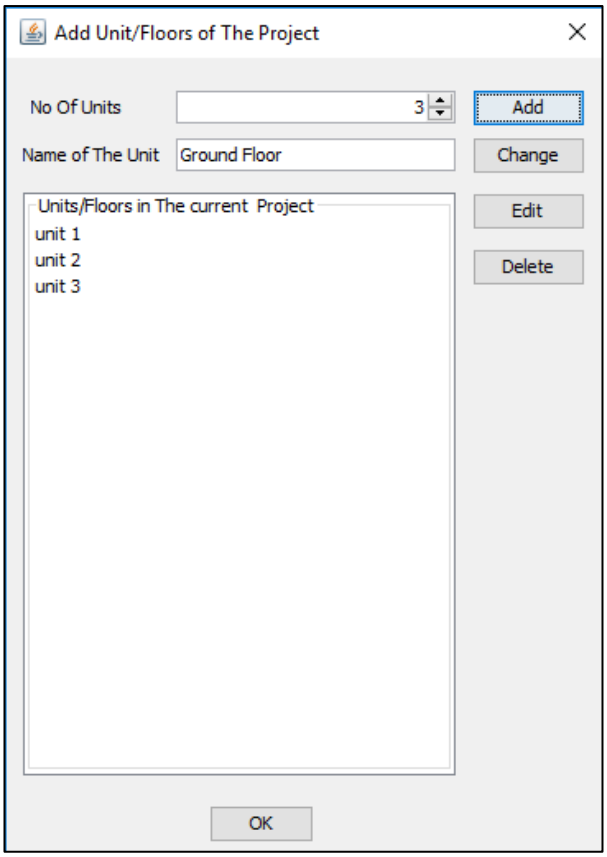


Figure 4-2 Add units dialog box

4.3.3 Inputting and Editing Activity Data

HRBSS allows adding, deleting, editing activities. To enter a new activity, the planner can provide information of the activity such as activity name, duration, and start and finish dates for manually scheduled activities, duration adjustment factors (weather factors), quantity, cost, for activities affected by weather, location (unit) and predecessors using the entry table (Figure 4-3) or Activity Information Dialog (Figure 4-4). These activity data can also be edited at any time. While a predecessor is entered, the successor will be automatically derived from it. When deleting an activity the program doesn't simply erase it, adjustments to other activities are made. The program allows manual and automatic scheduling.

4.3.3.1 Automatic and Manual Scheduling

Manually scheduled activities have a user-defined start, finish, and duration values. And automatically scheduled activities have start, finish, and duration values calculated by the program based on dependencies, constraints, calendars, and other factors.

HRBSS can be set to calculate the duration of activity automatically based on the quantity of work and productivity rate of the assigned crew.

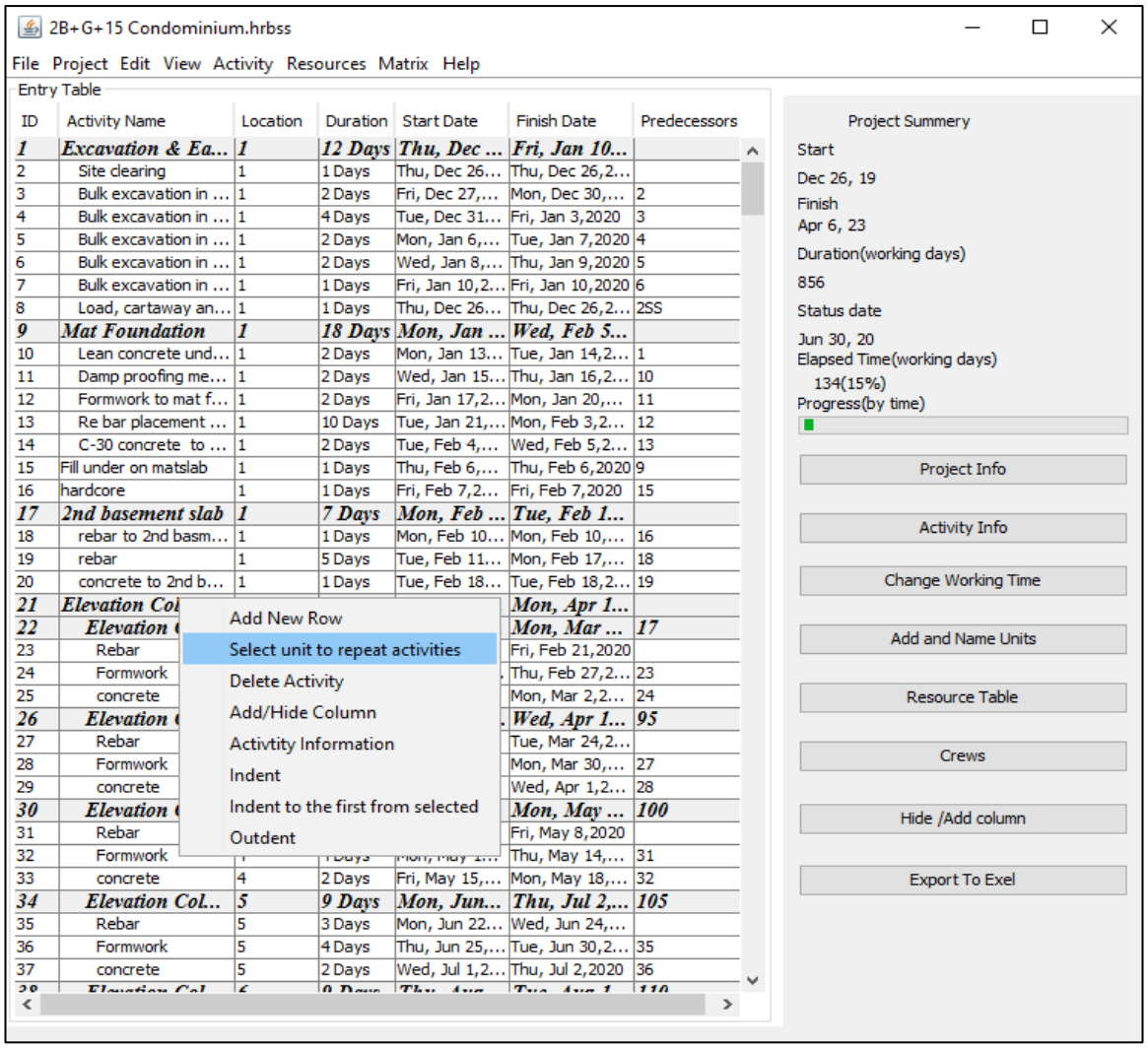


Figure 4-3 Entry Table

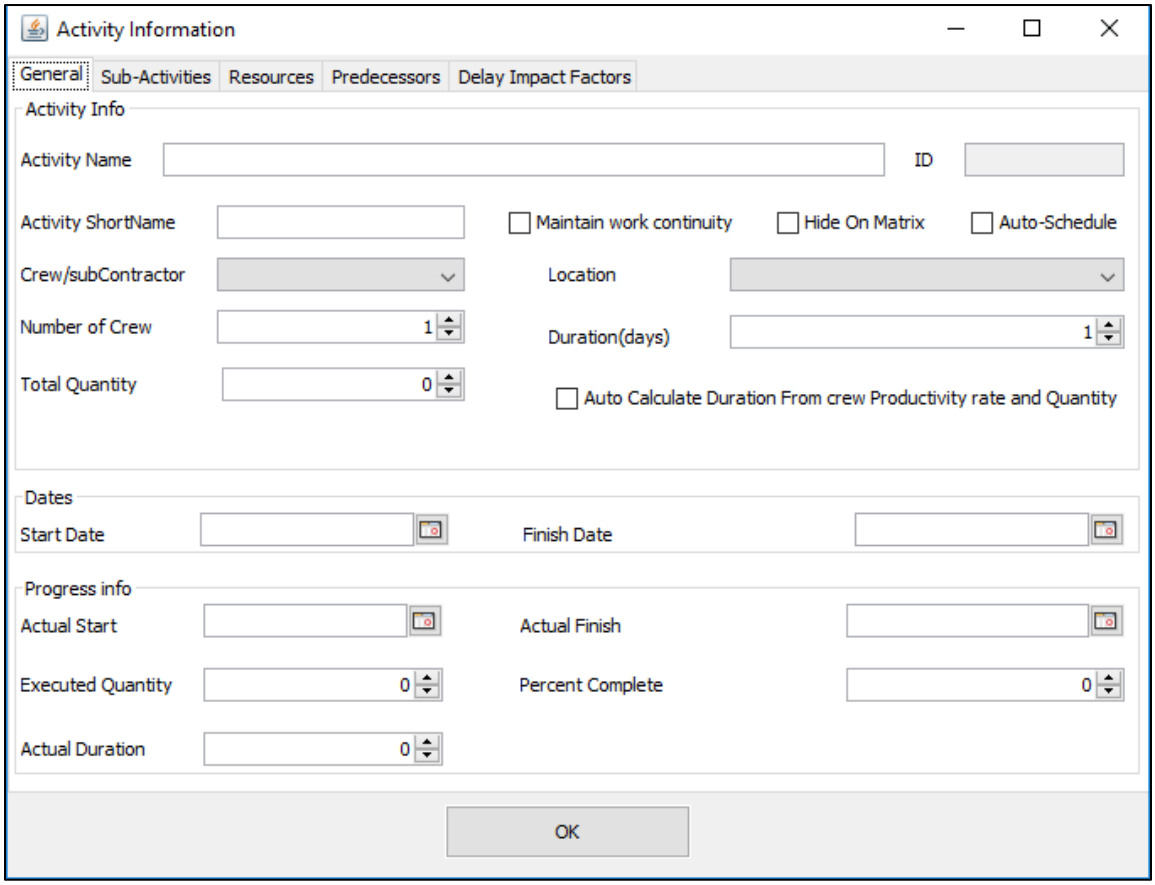


Figure 4-4 Activity information dialog

4.3.4 Entering Repetitive Activities

There is no need to repeatedly enter activities that repeat on more than one floor (repetitive activities). The user can enter similar information once and select the units where these activities are performed. If these repetitive activities have sub-activities, their data can also be entered once using the Quick entry dialog box as shown in Figure 4-5. Then, the system by itself generates the schedule for all floors where the activity is repeated based on that single input of data.

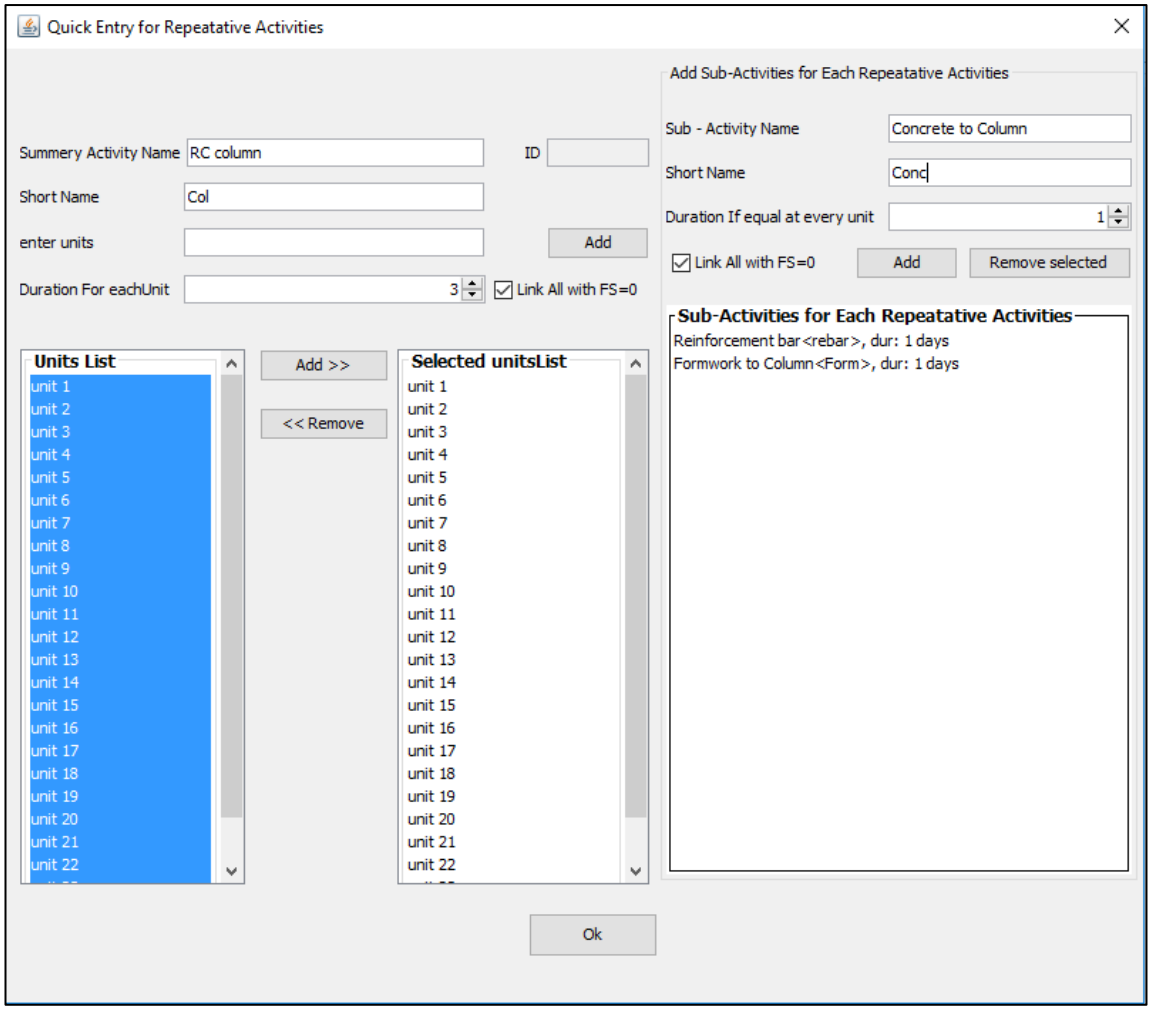


Figure 4-5 Quick entry dialog box

4.3.5 Schedule contingency

As it is mentioned in the previous chapter, the system allows to incorporate a contingency allowance for activities affected by the weather in the CPM calculations. This is done by inputting factor for a period with bad weather. These weather factors are percentages by which duration is divided to give an adjusted duration for each affected activity. The input of weather factors simply consists of inputting values between 0 to 1 corresponding to the percentage of efficiency for each month or a specific period of time (e.g. winter) as shown in Figure 4-6.

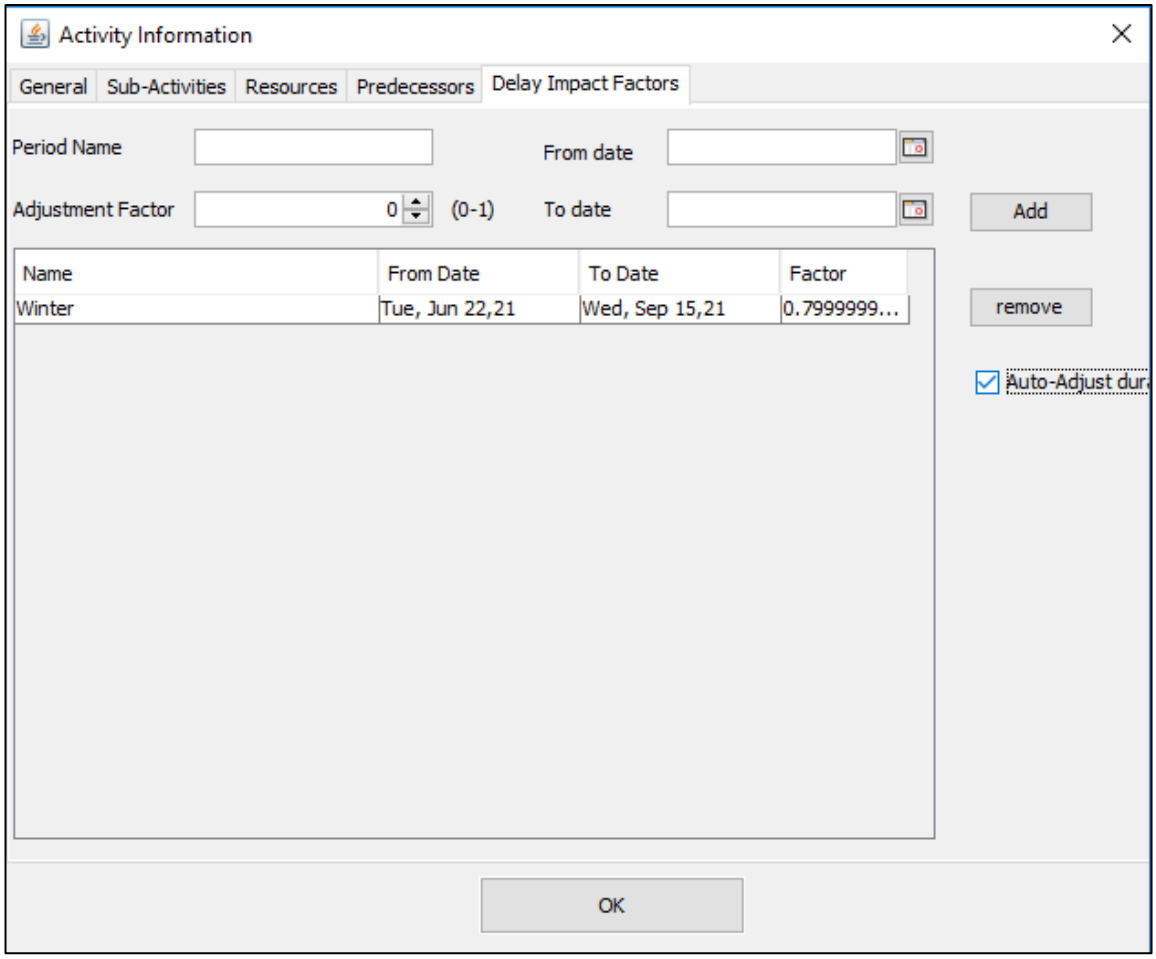


Figure 4-6 Inputting weather factors using activity information dialog box

4.3.6 Calendars

The purpose of a scheduling system is to relate various activities with time, and a common frame of reference for everybody is a calendar. In general, durations are expressed in working days and various calculations required to obtain a complete schedule of activities are related to those working days. Translating from working days to calendar days requires the identification of holidays, weekends, etc. which are considered as non-working days. The standard HRBSS calendar assumes five days as working days from Monday to Friday. However, the user can create a custom calendar to meet the unique project requirements. Nonworking periods or holidays can be specified as shown in the change working time dialog box as shown in Figure 4-7.

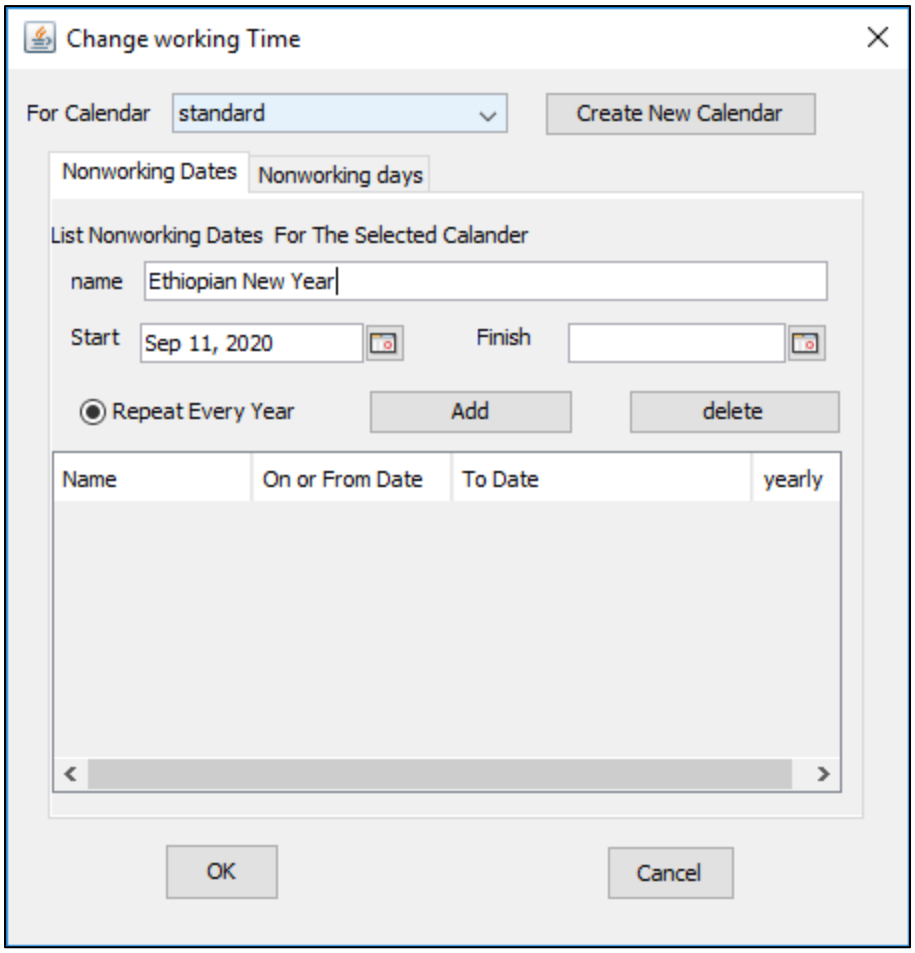


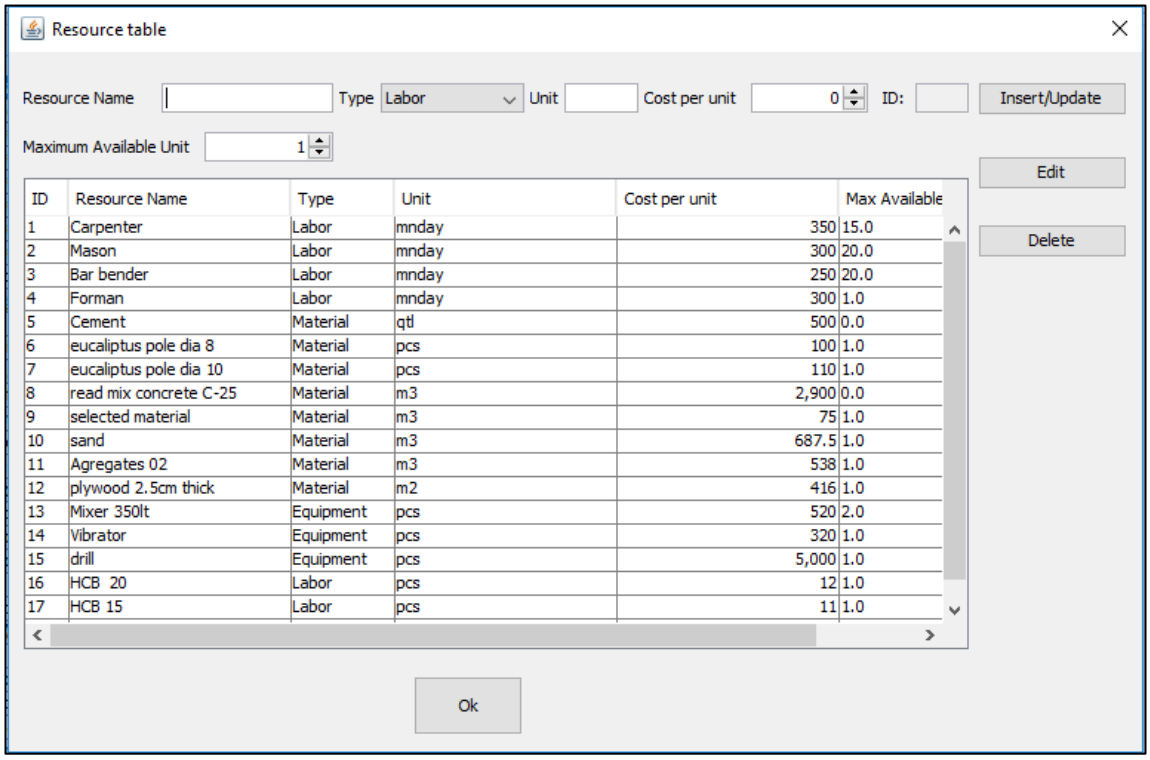
Figure 4-7 Change working time dialog box

4.3.7 Resources

A list of the resources available to complete any project can be created as shown in Figure 4-8 by providing the following information:

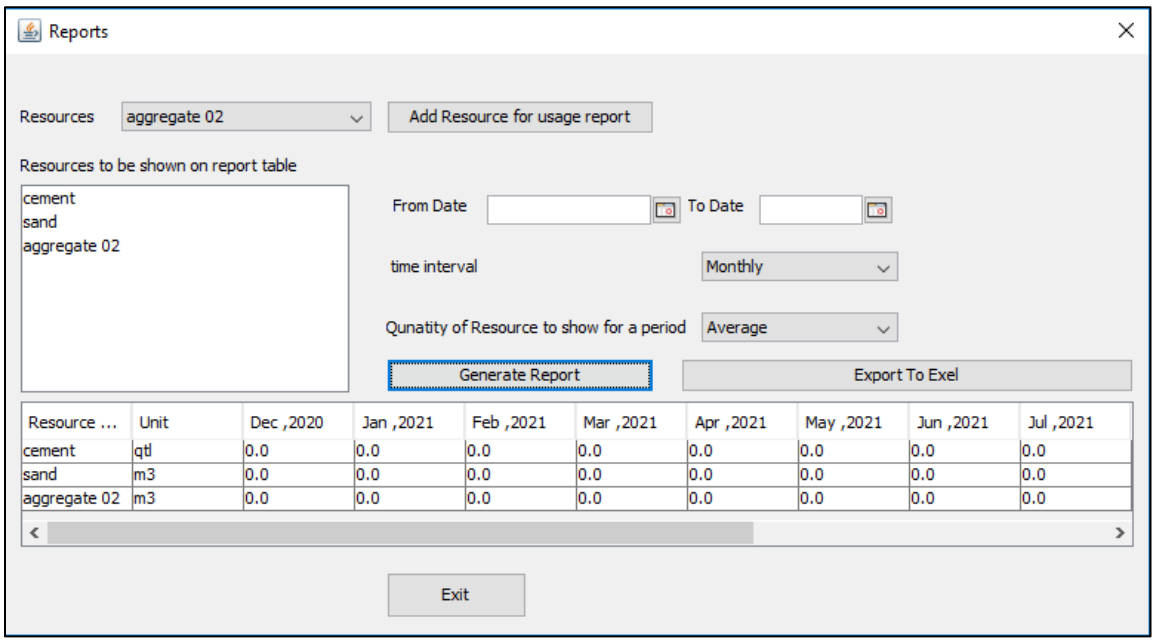
- Resource name
- Resource unit of measure
- Resource limits, which is the maximum number of units of this resource available for the current project
- Resource unit cost

Then, resources can be assigned to activities using activity information dialog. In addition to this, the program can also generate a resource schedule and export it to excel (Figure 4-9).



ID	Resource Name	Type	Unit	Cost per unit	Max Available
1	Carpenter	Labor	mnday	350	15.0
2	Mason	Labor	mnday	300	20.0
3	Bar bender	Labor	mnday	250	20.0
4	Forman	Labor	mnday	300	1.0
5	Cement	Material	qtl	500	0.0
6	eucaliptus pole dia 8	Material	pcs	100	1.0
7	eucaliptus pole dia 10	Material	pcs	110	1.0
8	read mix concrete C-25	Material	m3	2,900	0.0
9	selected material	Material	m3	75	1.0
10	sand	Material	m3	687.5	1.0
11	Agregates 02	Material	m3	538	1.0
12	plywood 2.5cm thick	Material	m2	416	1.0
13	Mixer 350lt	Equipment	pcs	520	2.0
14	Vibrator	Equipment	pcs	320	1.0
15	drill	Equipment	pcs	5,000	1.0
16	HCB 20	Labor	pcs	12	1.0
17	HCB 15	Labor	pcs	11	1.0

Figure 4-8 Resource table



Resource ...	Unit	Dec ,2020	Jan ,2021	Feb ,2021	Mar ,2021	Apr ,2021	May ,2021	Jun ,2021	Jul ,2021
cement	qtl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
sand	m3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
aggregate 02	m3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 4-9 Reports dialog

4.3.8 Resource leveling

HRBSS has a resource leveling feature to resolve over allocations by re-scheduling the start date of some of the activities. The user has options for leveling using the available total float of non-critical activities or delaying activities beyond their late start date (if over

allocations are not resolved using total floats) (Figure 4-10). The second option delays the project.

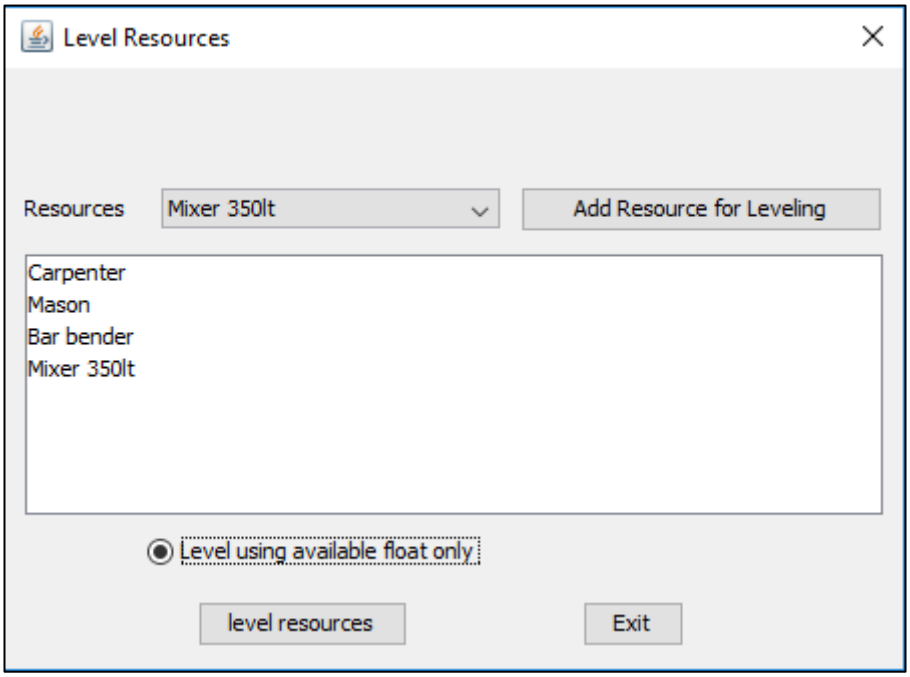


Figure 4-10 Resource leveling dialog

4.3.9 Matrix schedule

The matrix schedule consists of rows and columns where rows represent floors and columns represent activities (Figure 4-11). Activities information such as start and finish date on the corresponding floor are displayed on nodes or boxes as shown in Figure 4-12 The system uses different line thicknesses and line types and colors to show the status of activities. Coloring the box, varying line thickness and line types (such as solid and dashed lines) for borders of the box helps to easily understand the schedule. The format of these boxes (nodes) is customized with the format box dialog shown in Figure 4-13.

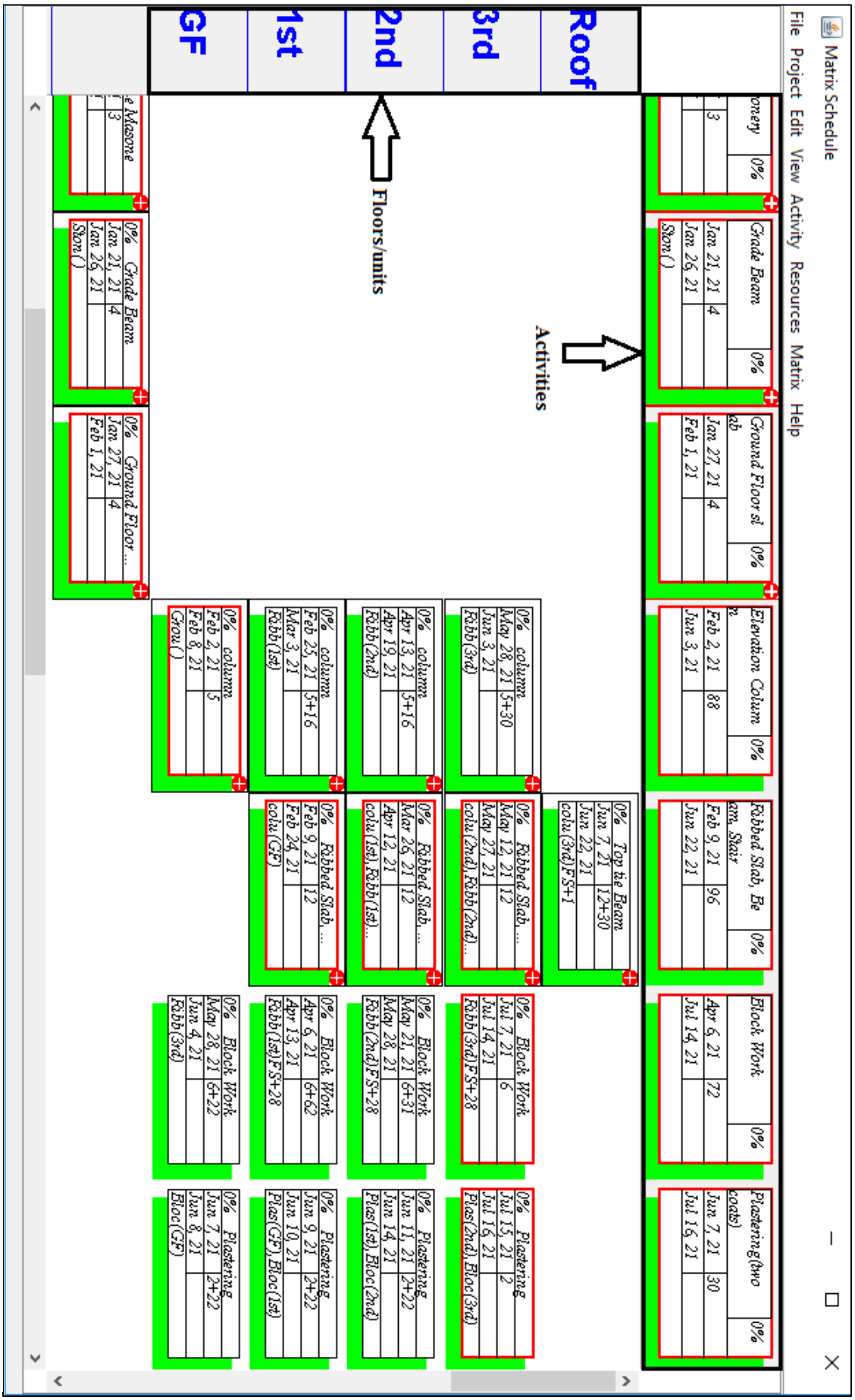


Figure 4-11 Matrix schedule

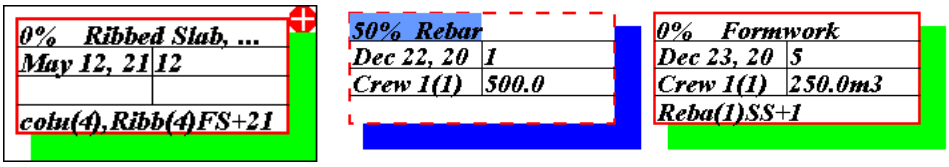


Figure 4-12 Different types of nodes

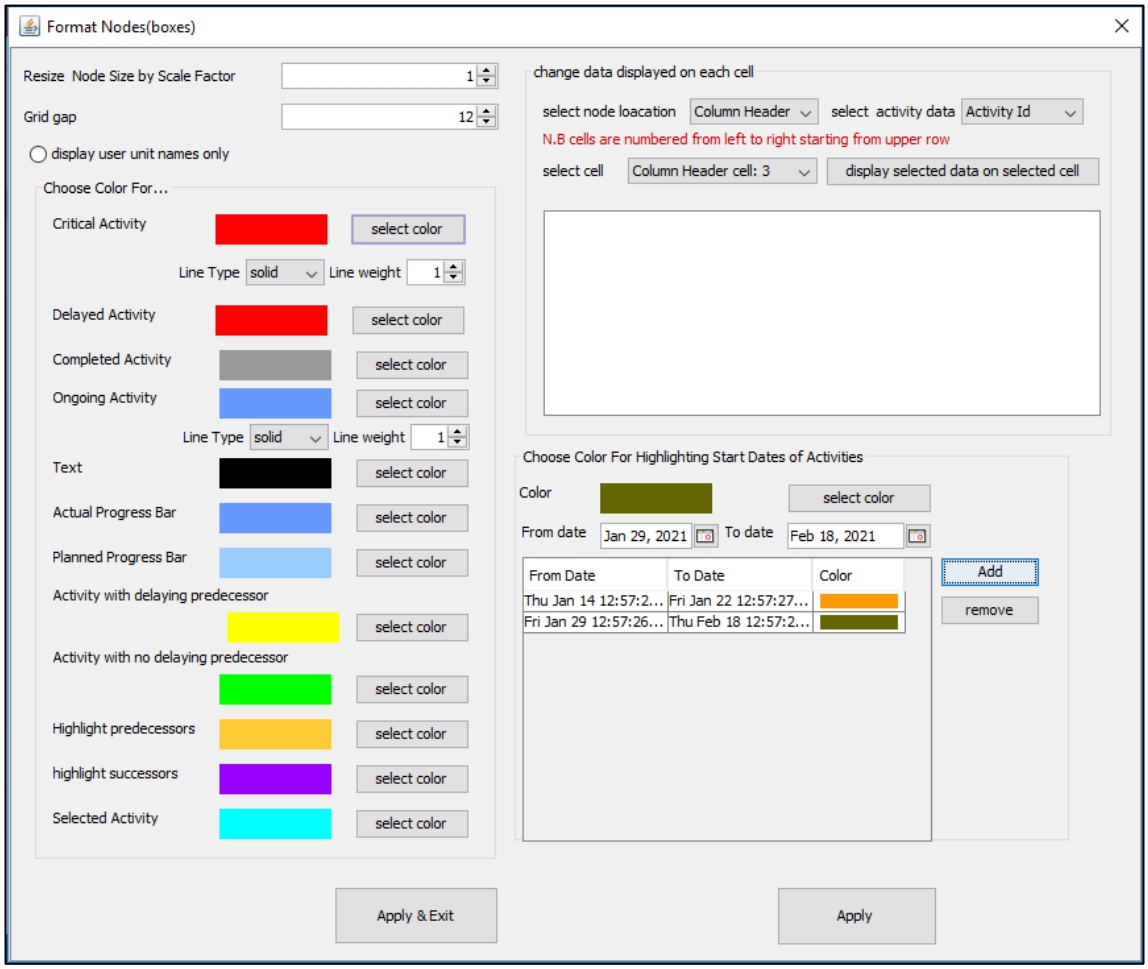


Figure 4-13 Format nodes dialog box

4.3.10 Updating

To update the schedule actual start dates, actual finish dates, actual duration, percent complete, or executed quantity of the activities are entered using activity information dialog or entry table as shown in Figure 4-14. When percent complete is entered, the executed quantity will be calculated by the system if the quantity of work is provided and vice versa. When the progress data are filled, the system automatically updates the schedule by incorporating actual start and finish dates in the calculation but keeping intact the logic and durations of the activities which have not been executed yet.

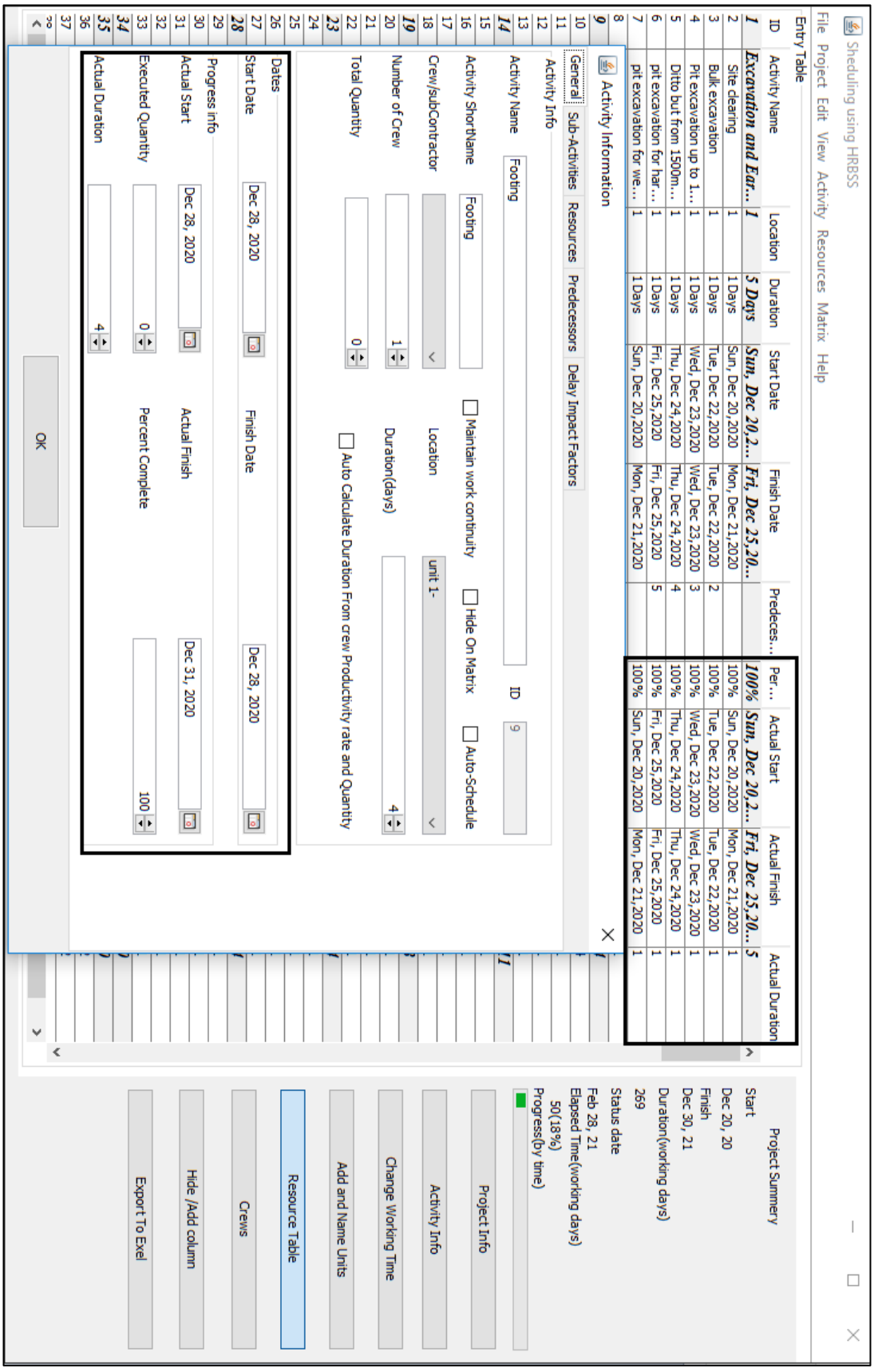


Figure 4-14 Inputting progress data using entry table and activity information dialog box

CHAPTER 5 VALIDATION OF THE SYSTEM

5.1 Introduction

The performance of the system is verified through three actual building projects. The first project is a 2B+G+18 apartment, the second project is a 2B+G+15 condominium project and the third one is a B+G+10 hotel building. Most building projects are very similar in nature as to the components and the process involved to put them together. It is clear, however, that a hotel and condominium building will not be done with the same quality standards. Thus, the projects are selected such that different types of work items are incorporated. However, the planning process used for these building types is similar. Thus, the schedule preparation of the 2B+G+18 apartment project using the HBRSS software is discussed in detail and screenshots of the schedules for the other two are presented in the following sections.

5.2 Case 1

The first project used for validation is a 2B+G+18 apartment built in Addis Ababa. The schedule and bill of quantities of the projects were obtained from the contractor. As it is mentioned earlier, high-rise buildings consist of repetitive activities and non-repetitive activities. Repetitive activities are performed on more than one floor while non-repetitive activities are executed once.

Before the structure starts on the first typical floor, some non-repetitive activities have to be performed. Some demolition work has to be done. The building has two basements that will be used as parking spaces for the users of the building and storage area. Due to the higher depth of excavation shoring is required to hold the sides of the excavation. Once the excavation work is completed, the foundations have to be built. Pile foundation is used in this project. Another series of non-repetitive activities is roofing work which includes fixing truss and roof cover. Installation of an elevator, landscaping works are also non-repetitive. These non-repetitive activities are shown as single boxes in a column (Figure 5-2).

Most of the activities involved in this project are repetitive. Some of them are concrete work, block work, aluminum work, cement and gypsum plastering, painting, metalwork,

finishing, glazing, carpentry and joinery, electrical, mechanical, and plumbing works, etc. (Figure 5-2). As it's mentioned in the previous chapters, the matrix method consists of rows and columns. Here, columns display activities, and rows represent units. This project is divided into 24 units (locations). On the first unit (row) non-repetitive activities such as shoring, excavation, foundation, landscaping are shown (Figure 5-2).

The second unit is labeled 2B (to mean the second basement) and activities on this floor are displayed at this row such as the second basement column (a summary activity with three sub-activities: fixing reinforcement bar to column, formwork to column, and concreting). The third unit is named 1B to mean 1st basement. Other units are GF (ground floor), Mez (mezzanine floor), 1st (1st floor), and so on. The last unit is named roof. At this unit two summary activities are shown: roofing work which includes fixing truss and roof cover and top tie beam & gutter which has three sub-tasks formwork, rebar, and concrete to roof beam & gutter. In this software, to enter the units, the number of units (24 in this case) is entered first and the default names (unit 1, unit 2...) are changed to desired unit names like the name of a floor (1st floor, 2nd floor, mechanical floor, etc.).

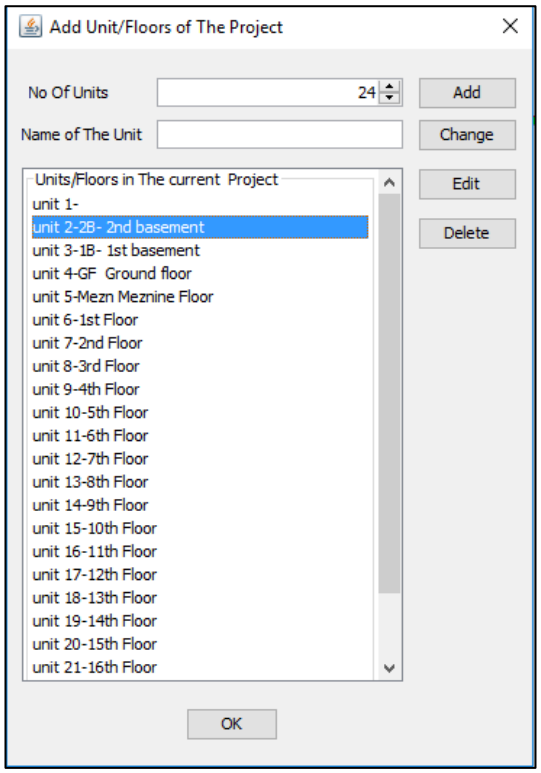


Figure 5-1 Adding and Naming Units

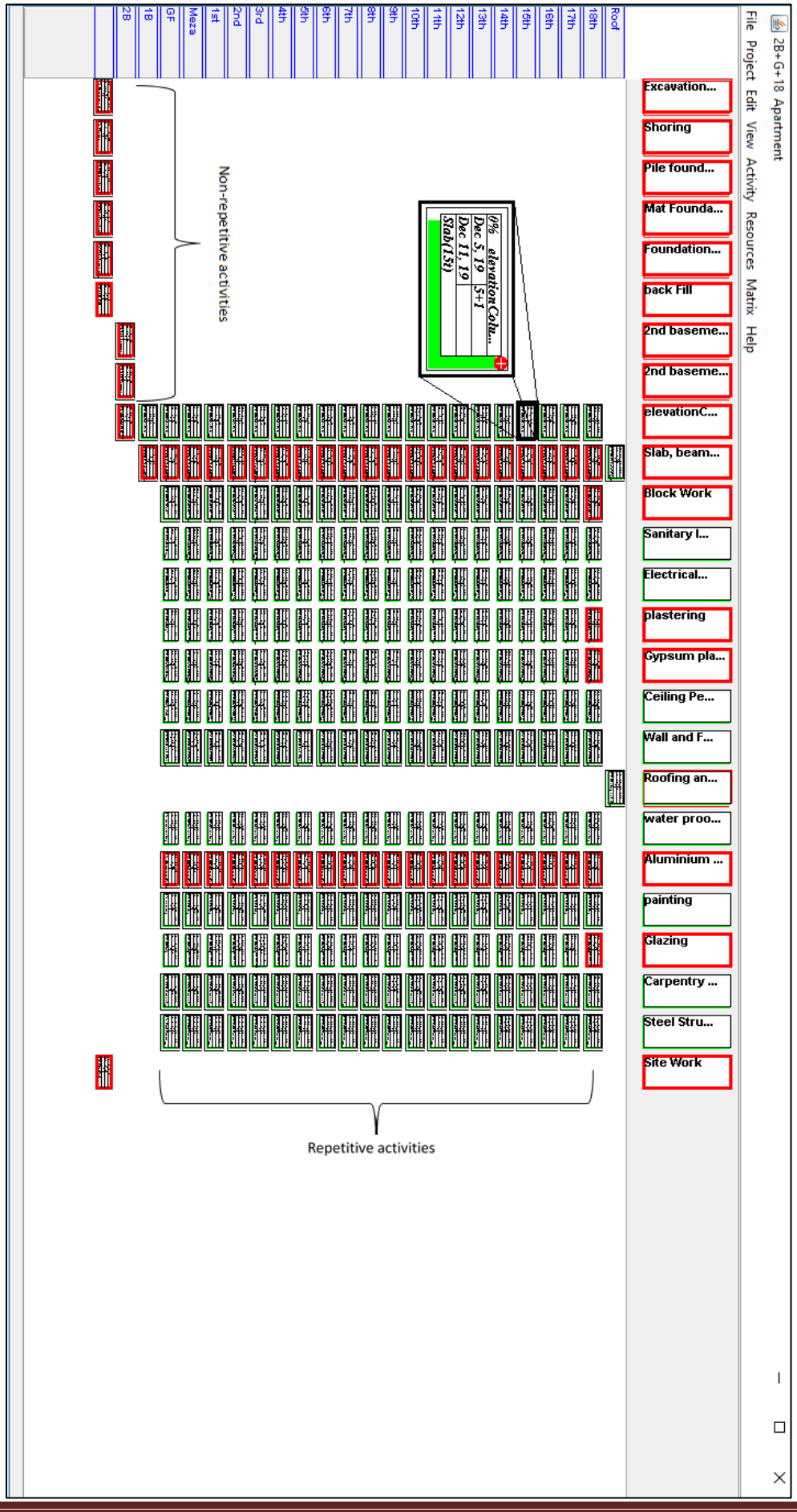
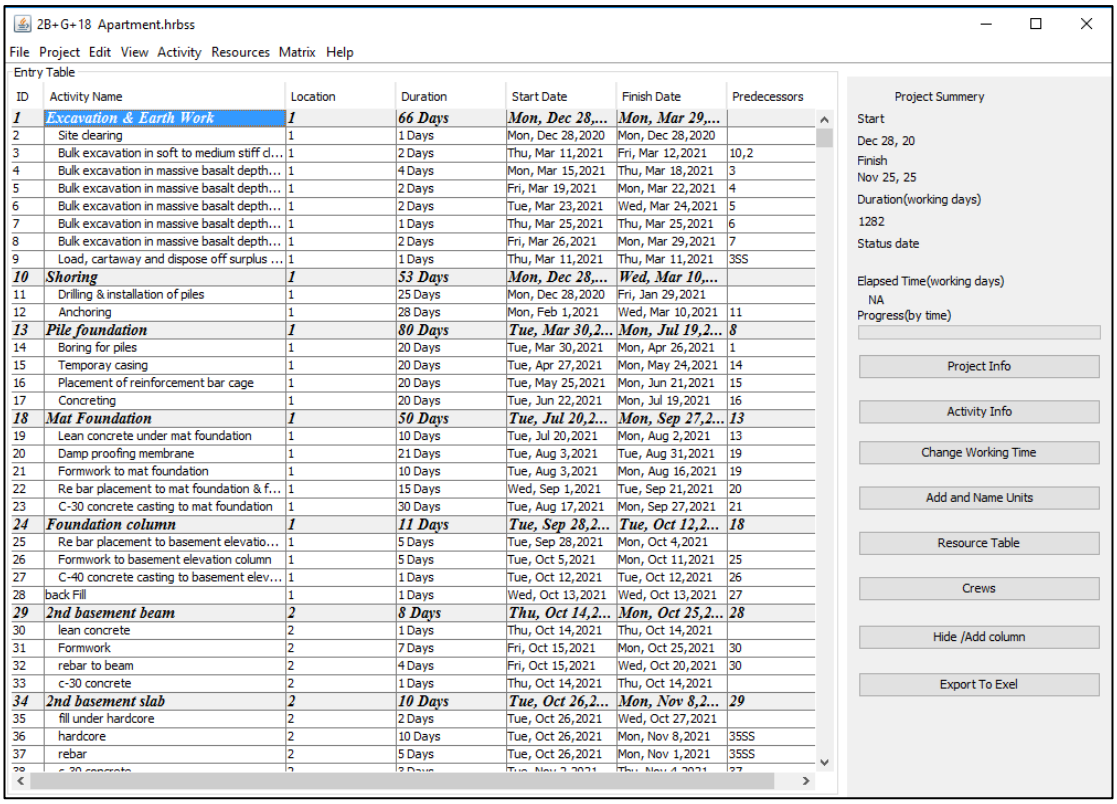


Figure 5-2 Repetitive and Non-Repetitive Activities

To set location (unit) for an activity, the unit number (1 for the first unit, 2 for the 2nd unit, and so on) can be entered using the entry table (Figure 5-3), or the unit can be selected using the activity information dialog (Figure 5-4).



ID	Activity Name	Location	Duration	Start Date	Finish Date	Predecessors
1	Excavation & Earth Work	1	66 Days	Mon, Dec 28,...	Mon, Mar 29,...	
2	Site clearing	1	1 Days	Mon, Dec 28,2020	Mon, Dec 28,2020	
3	Bulk excavation in soft to medium stiff cl...	1	2 Days	Thu, Mar 11,2021	Fri, Mar 12,2021	10,2
4	Bulk excavation in massive basalt depth...	1	4 Days	Mon, Mar 15,2021	Thu, Mar 18,2021	3
5	Bulk excavation in massive basalt depth...	1	2 Days	Fri, Mar 19,2021	Mon, Mar 22,2021	4
6	Bulk excavation in massive basalt depth...	1	2 Days	Tue, Mar 23,2021	Wed, Mar 24,2021	5
7	Bulk excavation in massive basalt depth...	1	1 Days	Thu, Mar 25,2021	Thu, Mar 25,2021	6
8	Bulk excavation in massive basalt depth...	1	2 Days	Fri, Mar 26,2021	Mon, Mar 29,2021	7
9	Load, cartaway and dispose off surplus ...	1	1 Days	Thu, Mar 11,2021	Thu, Mar 11,2021	3SS
10	Shoring	1	53 Days	Mon, Dec 28,...	Wed, Mar 10,...	
11	Drilling & installation of piles	1	25 Days	Mon, Dec 28,2020	Fri, Jan 29,2021	
12	Anchoring	1	28 Days	Mon, Feb 1,2021	Wed, Mar 10,2021	11
13	Pile foundation	1	80 Days	Tue, Mar 30,2...	Mon, Jul 19,2...	8
14	Boring for piles	1	20 Days	Tue, Mar 30,2021	Mon, Apr 26,2021	1
15	Temporay casing	1	20 Days	Tue, Apr 27,2021	Mon, May 24,2021	14
16	Placement of reinforcement bar cage	1	20 Days	Tue, May 25,2021	Mon, Jun 21,2021	15
17	Concreting	1	20 Days	Tue, Jun 22,2021	Mon, Jul 19,2021	16
18	Mat Foundation	1	50 Days	Tue, Jul 20,2...	Mon, Sep 27,2...	13
19	Lean concrete under mat foundation	1	10 Days	Tue, Jul 20,2021	Mon, Aug 2,2021	13
20	Damp proofing membrane	1	21 Days	Tue, Aug 3,2021	Tue, Aug 31,2021	19
21	Formwork to mat foundation	1	10 Days	Tue, Aug 3,2021	Mon, Aug 16,2021	19
22	Re bar placement to mat foundation & f...	1	15 Days	Wed, Sep 1,2021	Tue, Sep 21,2021	20
23	C-30 concrete casting to mat foundation	1	30 Days	Tue, Aug 17,2021	Mon, Sep 27,2021	21
24	Foundation column	1	11 Days	Tue, Sep 28,2...	Tue, Oct 12,2...	18
25	Re bar placement to basement elevatio...	1	5 Days	Tue, Sep 28,2021	Mon, Oct 4,2021	
26	Formwork to basement elevation column	1	5 Days	Tue, Oct 5,2021	Mon, Oct 11,2021	25
27	C-40 concrete casting to basement elev...	1	1 Days	Tue, Oct 12,2021	Tue, Oct 12,2021	26
28	back Fill	1	1 Days	Wed, Oct 13,2021	Wed, Oct 13,2021	27
29	2nd basement beam	2	8 Days	Thu, Oct 14,2...	Mon, Oct 25,2...	28
30	lean concrete	2	1 Days	Thu, Oct 14,2021	Thu, Oct 14,2021	
31	Formwork	2	7 Days	Fri, Oct 15,2021	Mon, Oct 25,2021	30
32	rebar to beam	2	4 Days	Fri, Oct 15,2021	Wed, Oct 20,2021	30
33	c-30 concrete	2	1 Days	Thu, Oct 14,2021	Thu, Oct 14,2021	
34	2nd basement slab	2	10 Days	Tue, Oct 26,2...	Mon, Nov 8,2...	29
35	fill under hardcore	2	2 Days	Tue, Oct 26,2021	Wed, Oct 27,2021	
36	hardcore	2	10 Days	Tue, Oct 26,2021	Mon, Nov 8,2021	35SS
37	rebar	2	5 Days	Tue, Oct 26,2021	Mon, Nov 1,2021	35SS

Figure 5-3 Entry Table

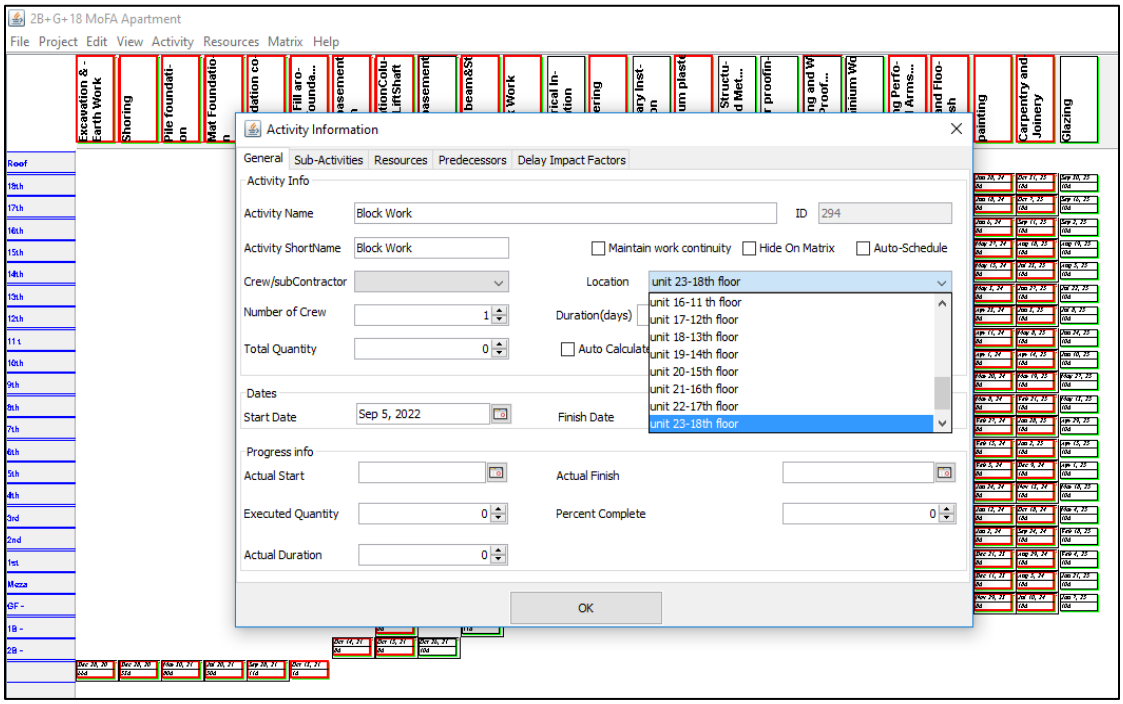


Figure 5-4 Setting units for activities using activity information dialog

This project consists of 1,134 activities. Organizing and grouping the activities are done to reduce the number of columns on the matrix schedule. And thus, related activities are grouped into summaries. For example, formwork, reinforcement, concrete for elevation columns can be summarized as one activity (summary activity) “Column” and when we want to view or enter progress data for each sub-activities, the summary can be expanded as shown in Figure 5-5. In this software, there is an option for expanding and collapsing these summaries. The schedules can be detailed enough for planning and tracking while minimizing the number of activities displayed on the matrix schedule.

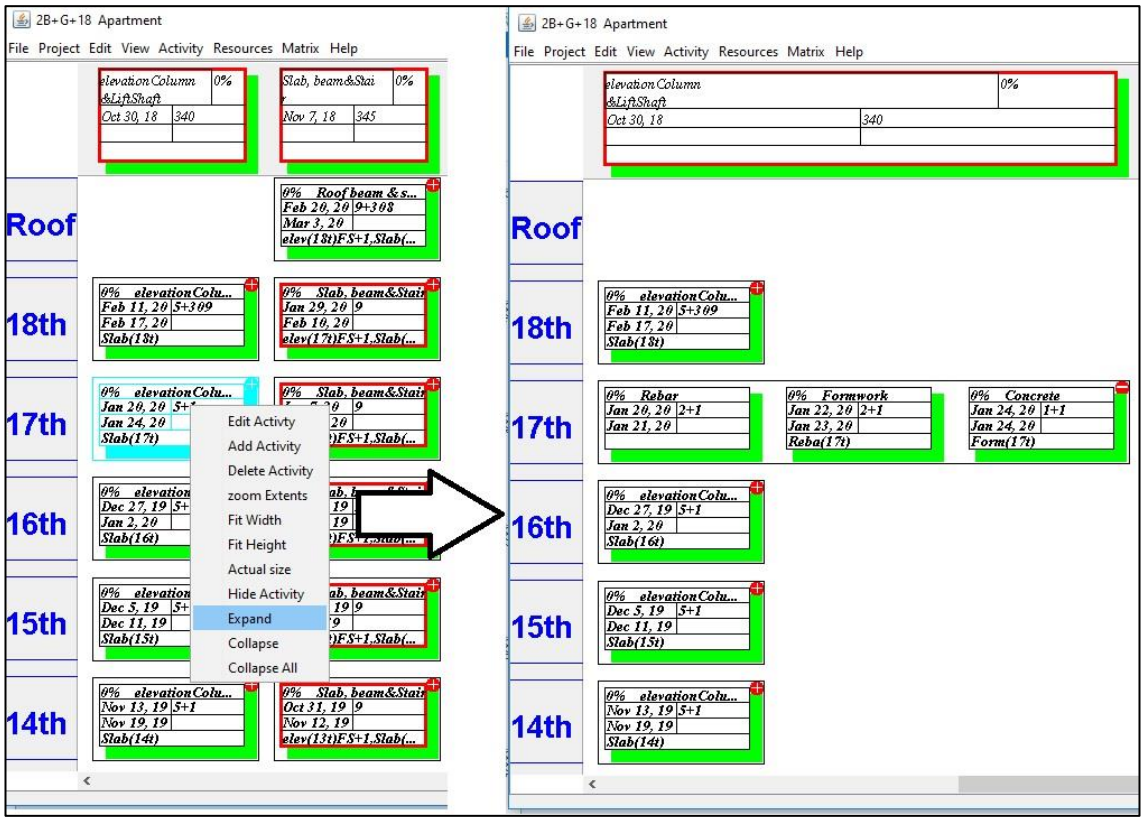


Figure 5-5 Expanding Summery Activity "Elevation Column" at 17th Floor

Activity duration is estimated based on the quantity of work, number, and crew productivity rates using the following formula Equation (1).

Activity duration = Quantity of work/ [number of crews x crew productivity rate] (1)

In this program, for some of the activities, the duration is set to be automatically calculated from the quantity of work, number, and productivity rate of the assigned crew (Figure 5-6). When crew number is increased or decreased or a different crew is used the effect on the project duration is seen as the duration is automatically updated. When Crew is created its composition and its productivity rate are specified and thus this data is used to estimate duration.

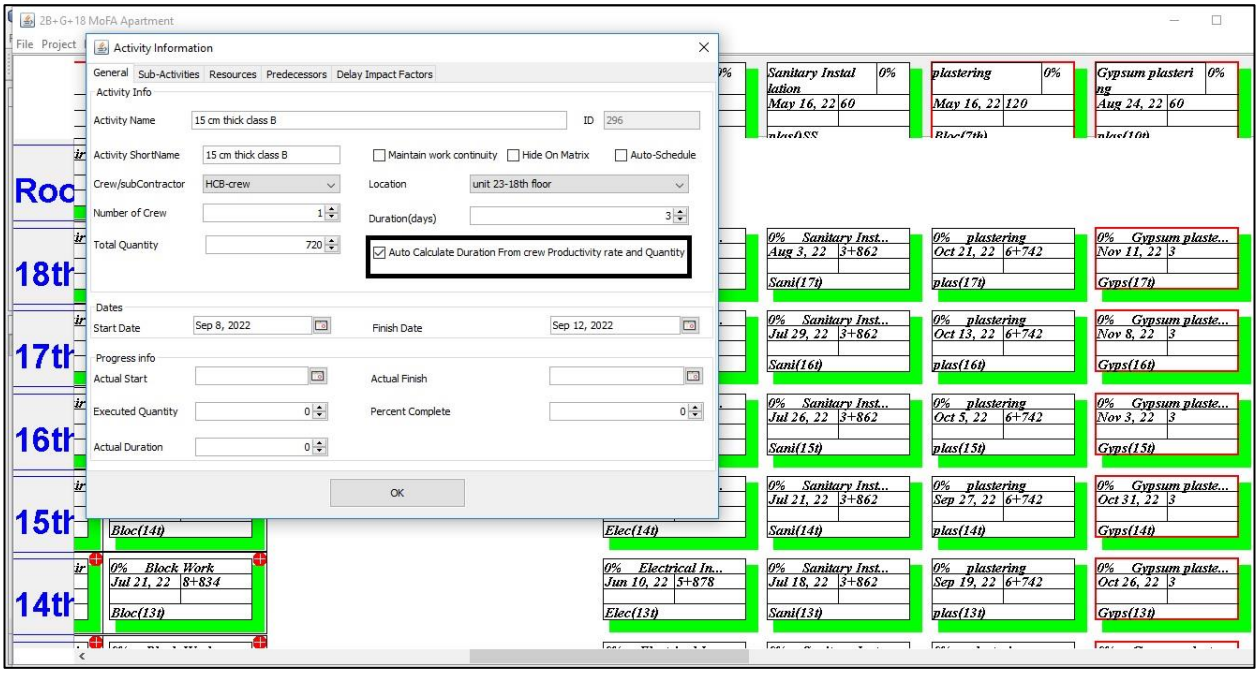


Figure 5-6 Automatic calculation of duration from crew productivity rate and quantity

Contingency for winter is set to be automatically added for some of the activities. As it is mentioned earlier weather can influence substantially the progress of construction work. Durations of Reinforced concrete structural works, External finishing works are selected to be adjusted to account for the influence of weather. In the summer months, the productivity of these works reaches its peak because of predominantly good weather conditions. However, winter months remain the most difficult to execute these activities. It is difficult to do formwork, fixing rebar, and concrete pouring while it is raining, meaning the workers will be idle and some time is lost causing a delay. Thus when the schedule is prepared, some provisions (contingency) for lost days should be included in

order to evaluate better the real time needed to perform the construction work. Figure 5.42 shows that the duration of summery activity named “slab and beam” is set to adjust by an efficiency factor of 0.75, if its sub-activities will be executed from date June 30 to September 10. Thus, the normal duration of this activity will be divided by the weather factor (0.75) resulting in a longer duration.

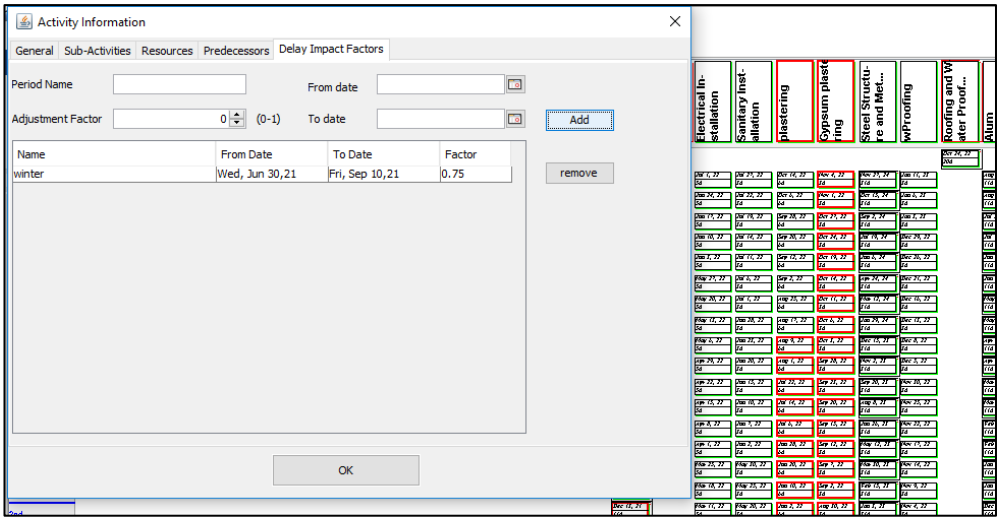
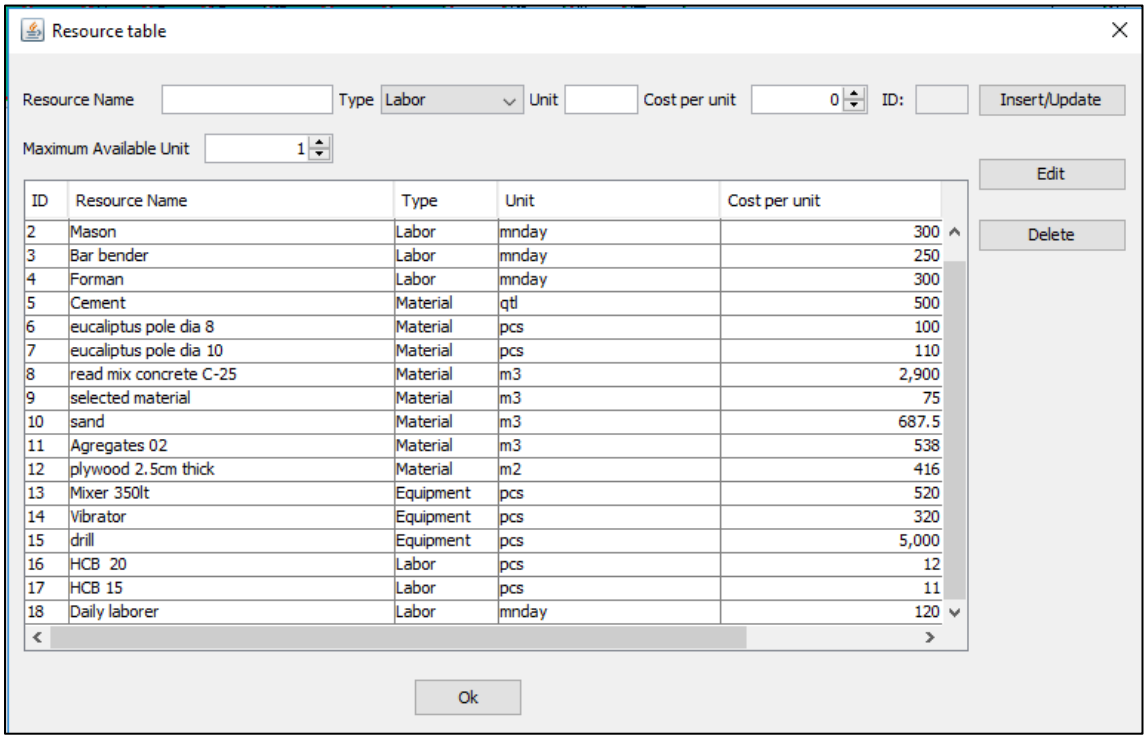


Figure 5-7 Adding weather factor for "winter"

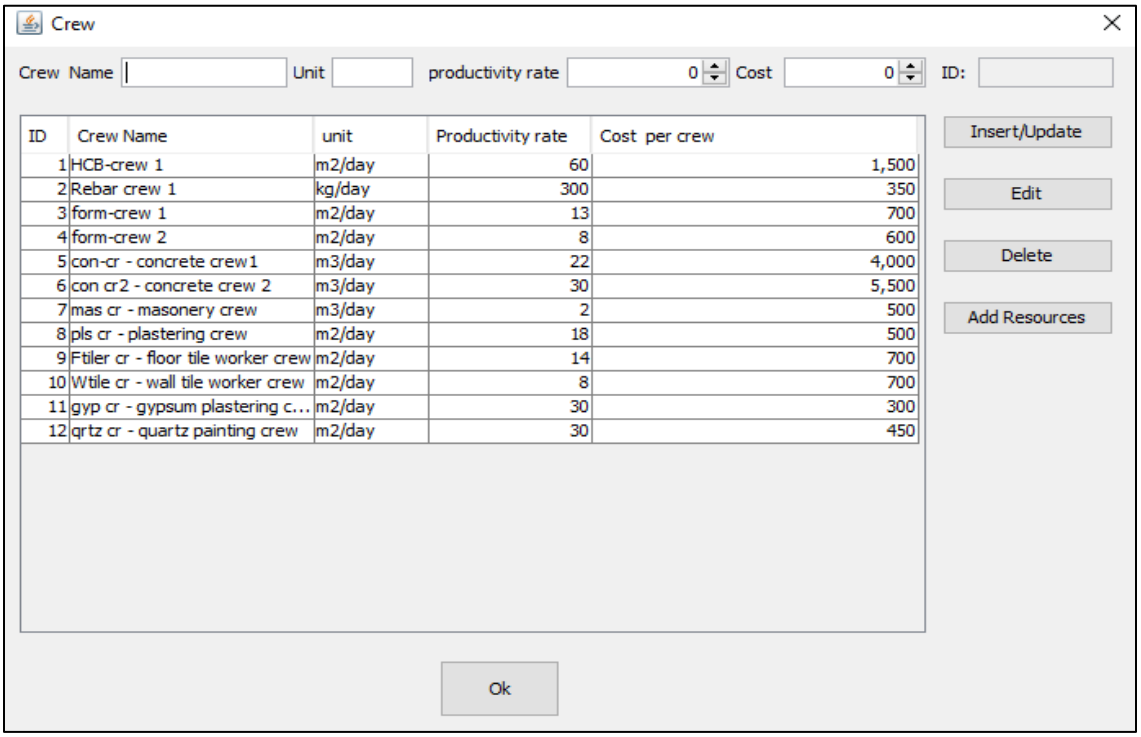
Resource allocation is done by creating a list of the resources required to execute the activities (Figure 5-9) and then assigning resources for activities using the activity information dialog. As shown in Figure 5-10 the resource name is selected and the required number or quantity is entered. When a resource is assigned for a summery, the system automatically assigns it to all sub-activities.

Crews are also created with the resources they comprise and their productivity rate as shown in Figure 5.42. So when a crew is assigned for an activity, the resources in the crew will be also assigned for the activity and are available in resource reports and for resource-leveling. The quantity of such resources (from the assigned crew) required by the activity is obtained by multiplying the number of crews and the quantity of the resource in the crew. In the matrix view, the information shown on each cell of the box can be customized and the assigned crew name and number can be shown on boxes on the matrix.



ID	Resource Name	Type	Unit	Cost per unit
2	Mason	Labor	mnday	300
3	Bar bender	Labor	mnday	250
4	Forman	Labor	mnday	300
5	Cement	Material	qtl	500
6	eucaliptus pole dia 8	Material	pcs	100
7	eucaliptus pole dia 10	Material	pcs	110
8	read mix concrete C-25	Material	m3	2,900
9	selected material	Material	m3	75
10	sand	Material	m3	687.5
11	Agregates 02	Material	m3	538
12	plywood 2.5cm thick	Material	m2	416
13	Mixer 350lt	Equipment	pcs	520
14	Vibrator	Equipment	pcs	320
15	drill	Equipment	pcs	5,000
16	HCB 20	Labor	pcs	12
17	HCB 15	Labor	pcs	11
18	Daily laborer	Labor	mnday	120

Figure 5-9 Resources



ID	Crew Name	unit	Productivity rate	Cost per crew
1	HCB-crew 1	m2/day	60	1,500
2	Rebar crew 1	kg/day	300	350
3	form-crew 1	m2/day	13	700
4	form-crew 2	m2/day	8	600
5	con-cr - concrete crew1	m3/day	22	4,000
6	con cr2 - concrete crew 2	m3/day	30	5,500
7	mas cr - masonery crew	m3/day	2	500
8	pls cr - plastering crew	m2/day	18	500
9	Ftiler cr - floor tile worker crew	m2/day	14	700
10	Wtile cr - wall tile worker crew	m2/day	8	700
11	gyp cr - gypsum plastering c...	m2/day	30	300
12	qrtz cr - quartz painting crew	m2/day	30	450

Figure 5-8 Crews

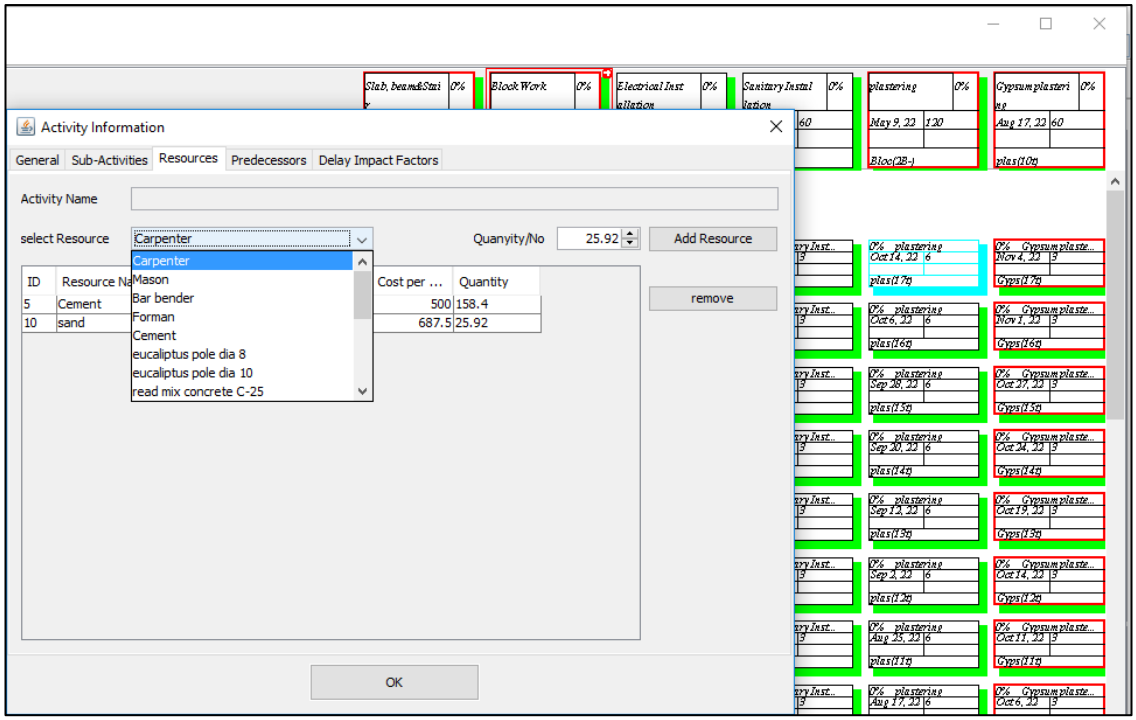
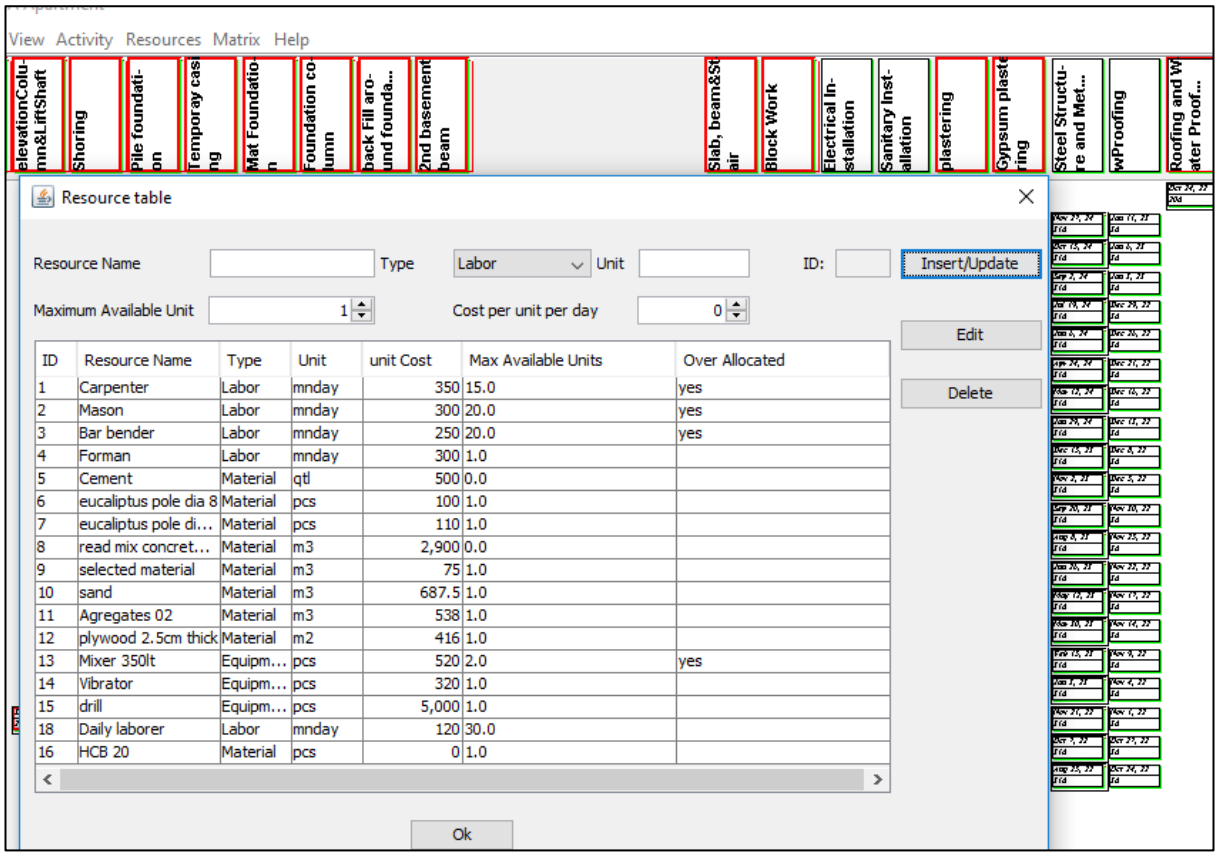


Figure 5-10 Assigning resource to activities

Once assigning resources was finished, over allocations were checked using the resource table. Some of the resources were over-allocated as shown in Figure 5-11.



ID	Resource Name	Type	Unit	unit Cost	Max Available Units	Over Allocated
1	Carpenter	Labor	mnday	350	15.0	yes
2	Mason	Labor	mnday	300	20.0	yes
3	Bar bender	Labor	mnday	250	20.0	yes
4	Forman	Labor	mnday	300	1.0	
5	Cement	Material	qtl	500	0.0	
6	eucaliptus pole dia 8	Material	pcs	100	1.0	
7	eucaliptus pole di...	Material	pcs	110	1.0	
8	read mix concret...	Material	m3	2,900	0.0	
9	selected material	Material	m3	75	1.0	
10	sand	Material	m3	687.5	1.0	
11	Agregates 02	Material	m3	538	1.0	
12	plywood 2.5cm thick	Material	m2	416	1.0	
13	Mixer 350lt	Equipm...	pcs	520	2.0	yes
14	Vibrator	Equipm...	pcs	320	1.0	
15	drill	Equipm...	pcs	5,000	1.0	
18	Daily laborer	Labor	mnday	120	30.0	
16	HCB 20	Material	pcs	0	1.0	

Figure 5-11 Over allocated resources

Thus, to resolve over allocations, the resources are selected as shown below and leveled Figure 5-12.

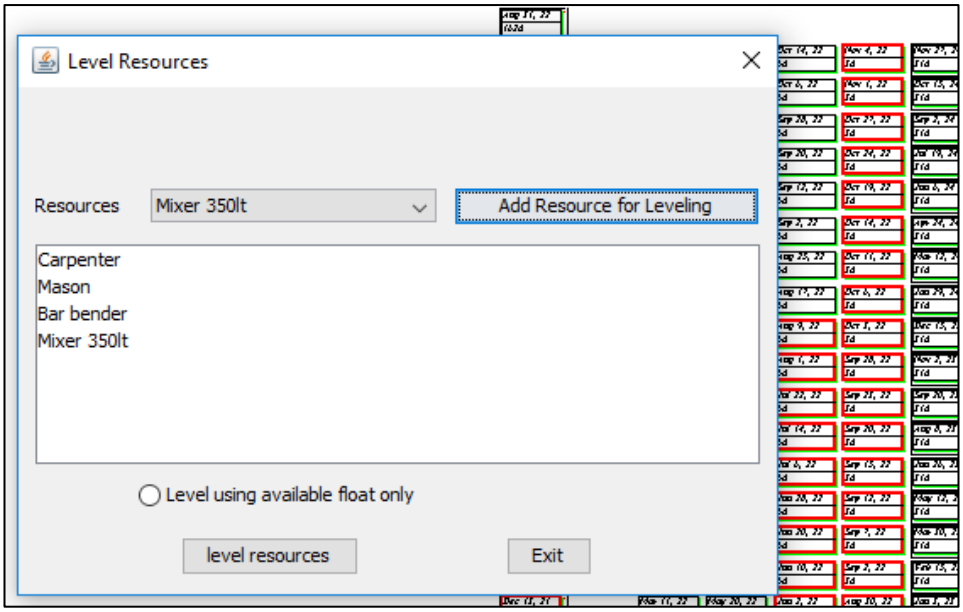


Figure 5-12 Levelling over-allocated resources

The matrix schedule for this project is shown in Figure 5-13. The critical activities are shown with red border boxes or nodes. One can easily identify these activities graphically on the matrix schedule.

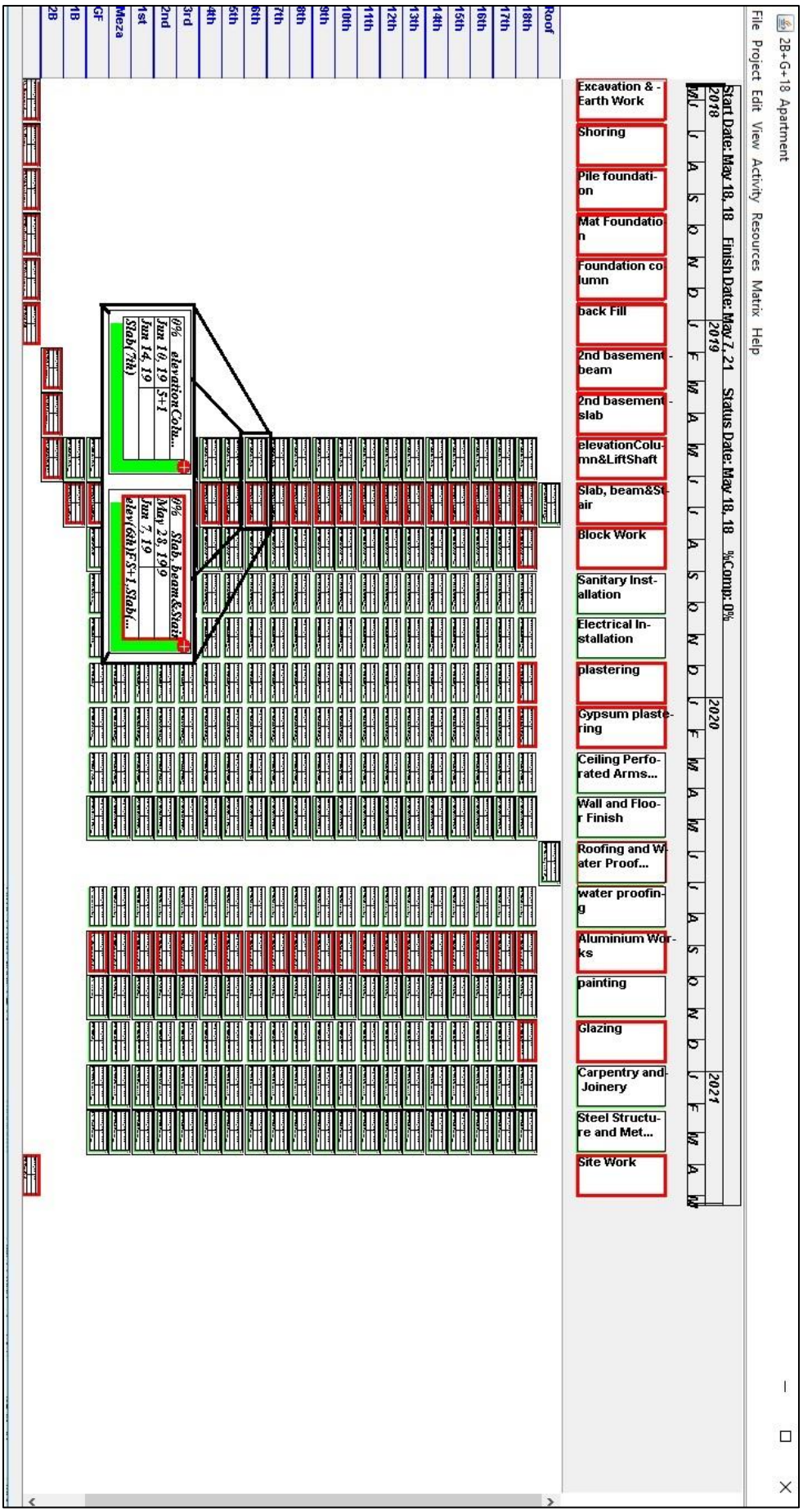


Figure 5-13 2B+G+18 Apartment matrix schedule

5.3 Case 2

The second project used for validation is a 2B+G+15 condominium project built in Addis Ababa. The schedule and bill of quantities of the projects were obtained from the contractor. The same procedures are followed to prepare the schedule as the first project. The total number of planned activities is 504. The schedule of this project is prepared using the proposed system as shown in Figure 5-14. There are 23 repetitive activities in this project and 20 non-repetitive activities which are grouped into seven summaries.

The repetitive activities are grouped into 14 summary activities to reduce the number of columns in the matrix view. Rebar, formwork, and concrete for elevation columns and shear wall are grouped as one summary activity "column and shear wall". A ribbed slab is used in this project and the activities for the ribbed slab, beams, and staircase are summarized in one column as "ribbed slab and beam". When expanding summaries, the sub-activities such as reinforcement work, formwork, and concrete will be shown.

5.4 Case 3

The third project is a B+G+10 hotel project under construction in Jimma. The schedule of this project is prepared using the proposed system as shown in Figure 5-15. This project's schedule comprises 415 activities. There are 15 non-repetitive activities summarized into 9 summaries such as mat foundation, selected fill on mat slab, hardcore, stone masonry, elevator installation, etc. there are 400 repetitive activities grouped into 12 summaries, thus there are 12 matrix columns when all summaries are not expanded. The same approach is followed for grouping repetitive activities as the above two projects.

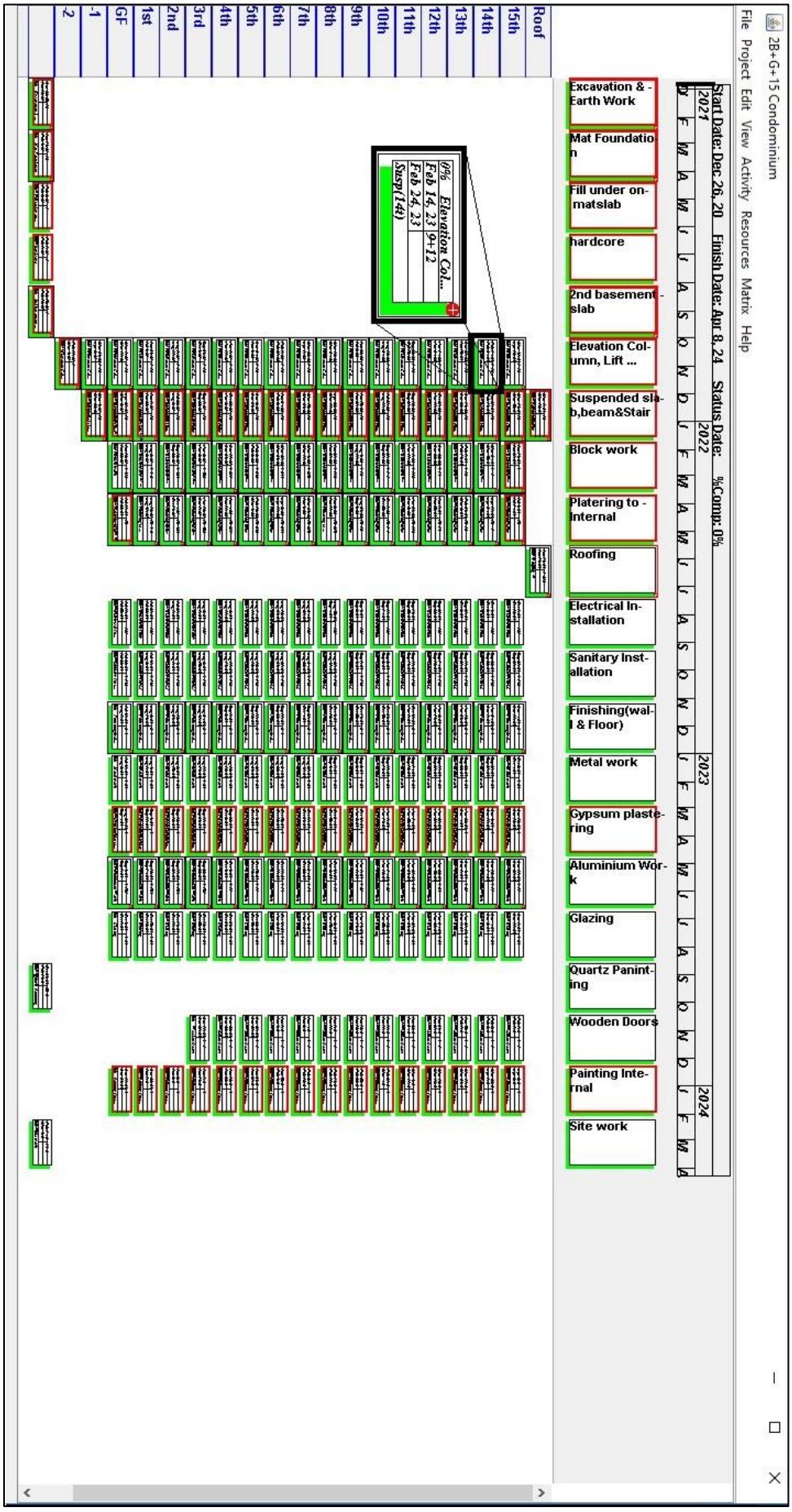


Figure 5-14 Matrix schedule for 2B+G+15 condominium project

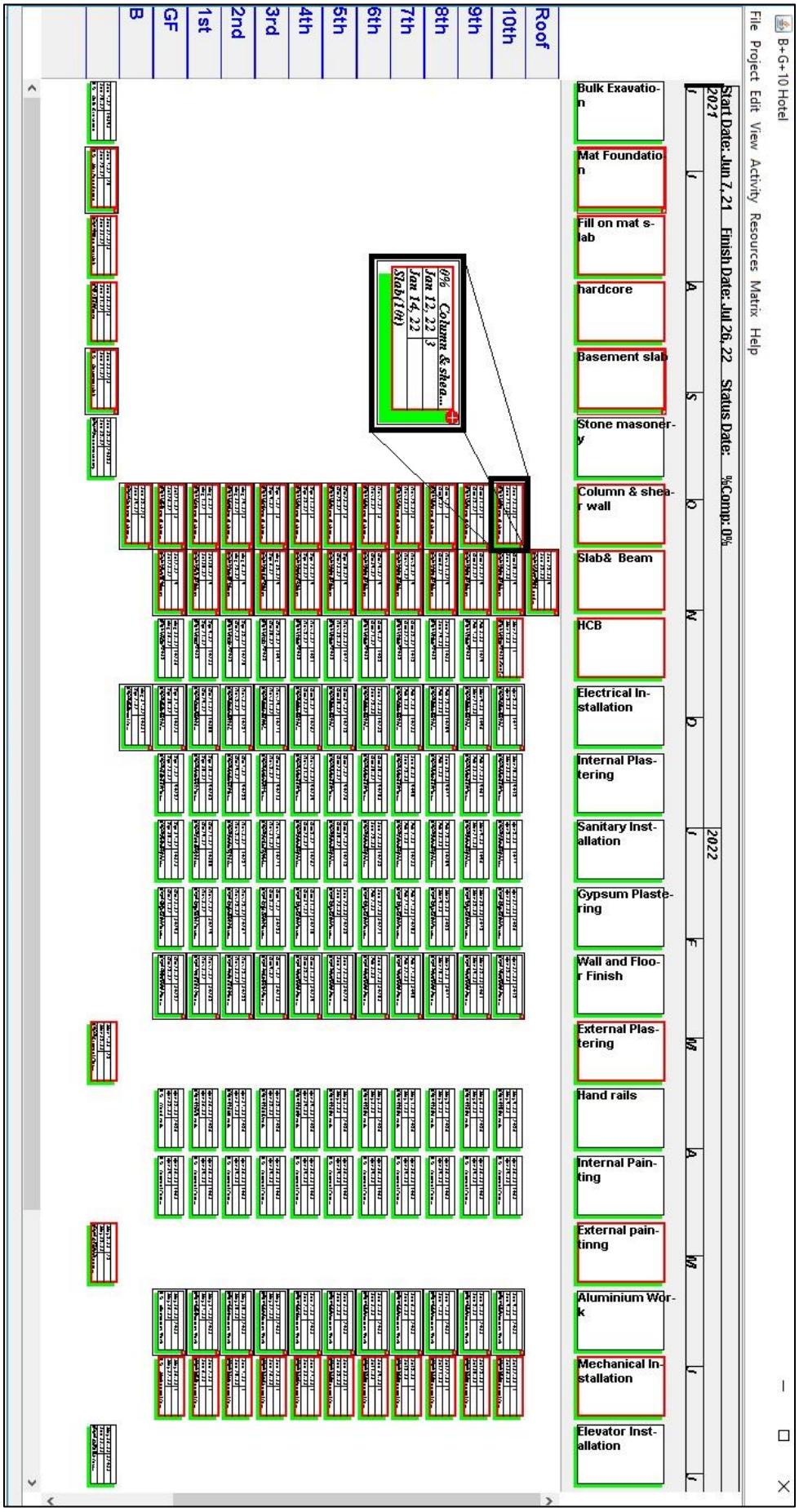


Figure 5-15 Matrix schedule for G+10 hotel project

5.5 Progress Monitoring using HRBSS

The system was used to monitor progress. In order to easily identify the status of the activities graphically the format of the boxes (nodes) of the matrix are customized as shown in Figure 5-16. By varying border line color, thickness and type (dashed or solid), fill color, and shadow color, the status of the activities can be shown. For this project, the following formats are used.

- Completed activities are represented by light gray filled nodes
- Ongoing activities have a blue shadow.
- Delayed activities are represented by nodes with red-colored lines and shadows.
- Nodes with a red solid thick line border are critical activities.
- Activities that are not commenced and their start will not be delayed due to the current progress of their predecessor till the status date are identified by a node with green shadow.
- Activities that are not commenced but and their start will not be delayed due to the actual progress of their predecessor (unless corrective measure is taken) till the status date are identified by a node with yellow shadow.
- The percent complete of each activity is shown on the first row of the boxes in Figure and with a blue progress bar.
- The planned progress of the activity is also shown with a dark gray progress bar on the same location with the percent complete bar.
- Start date, duration with total float separated by a plus sign.

The progress of the project and the status of activities (whether they are delayed, completed, being executed) is clearly shown with the formats listed above applied to the nodes. When the progress data and status date is entered, the program automatically applies these formats (Figure 5-17 and Figure 5-18). Critical activities are also clearly identified. From Figure 5-18, one can easily see that activity “slab and beam” is critical from the second basement up to the 15th floor and the last floor of block work and plastering are also critical.

As shown in Figure 5-19 different node fill colors are used to identify the time of execution of activities. If activities scheduled for next month are required to be identified, then a certain color can be used for this month and highlight the activities performed in this

month. For example, purple color is used for activities scheduled for 2021. The time scale is also highlighted with purple (Figure 5-19).

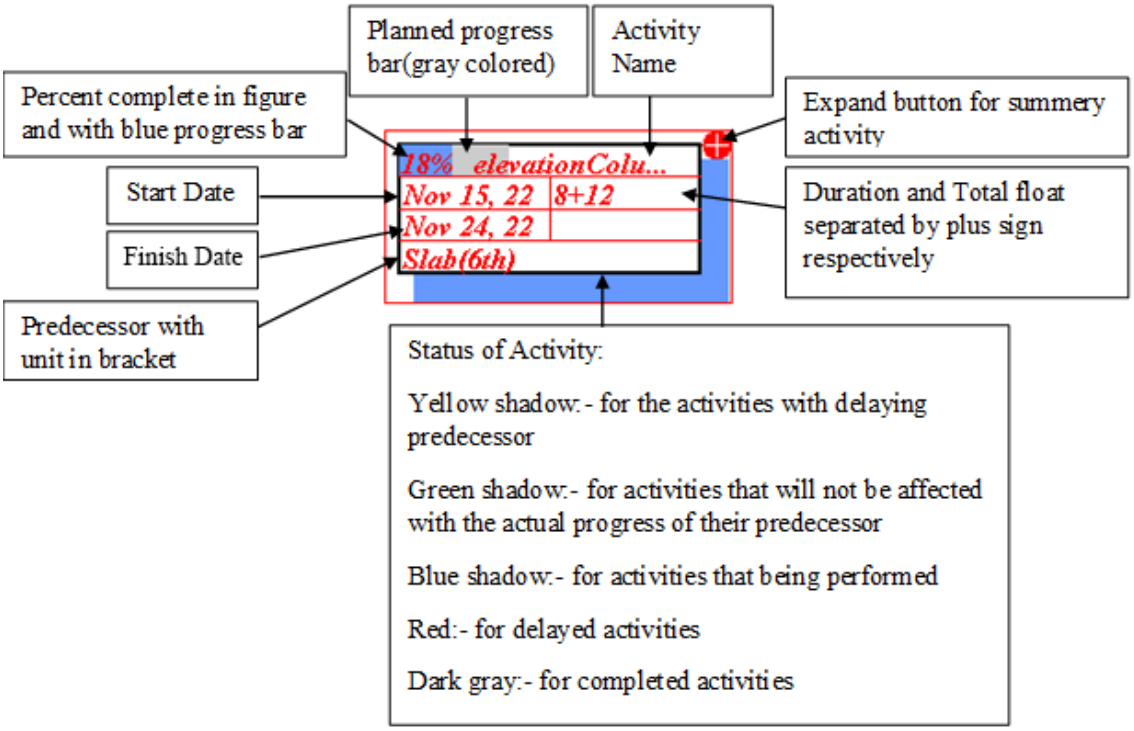


Figure 5-16 Node format used in the schedules

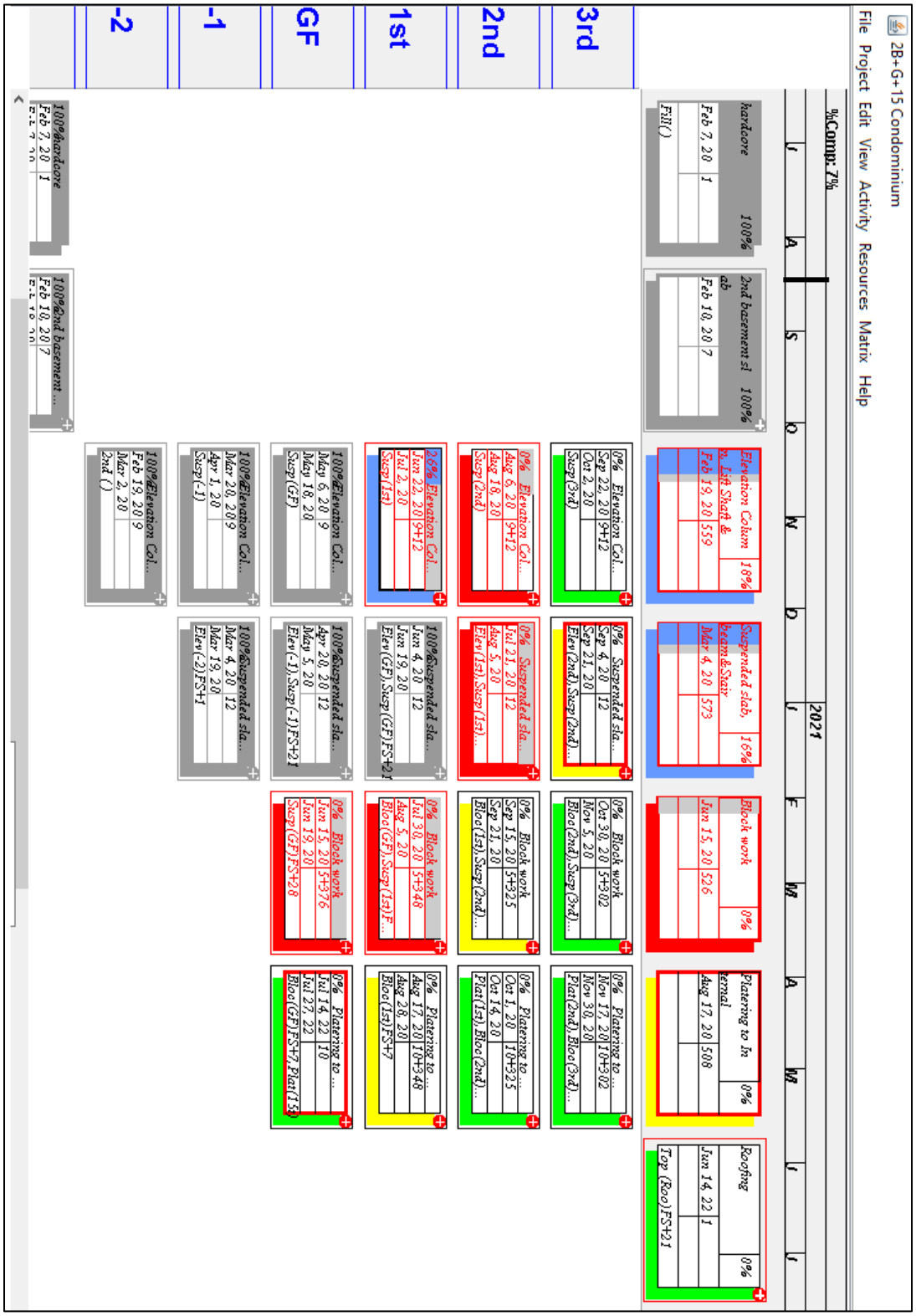


Figure 5-17 Tracking Matrix (zoomed in View)

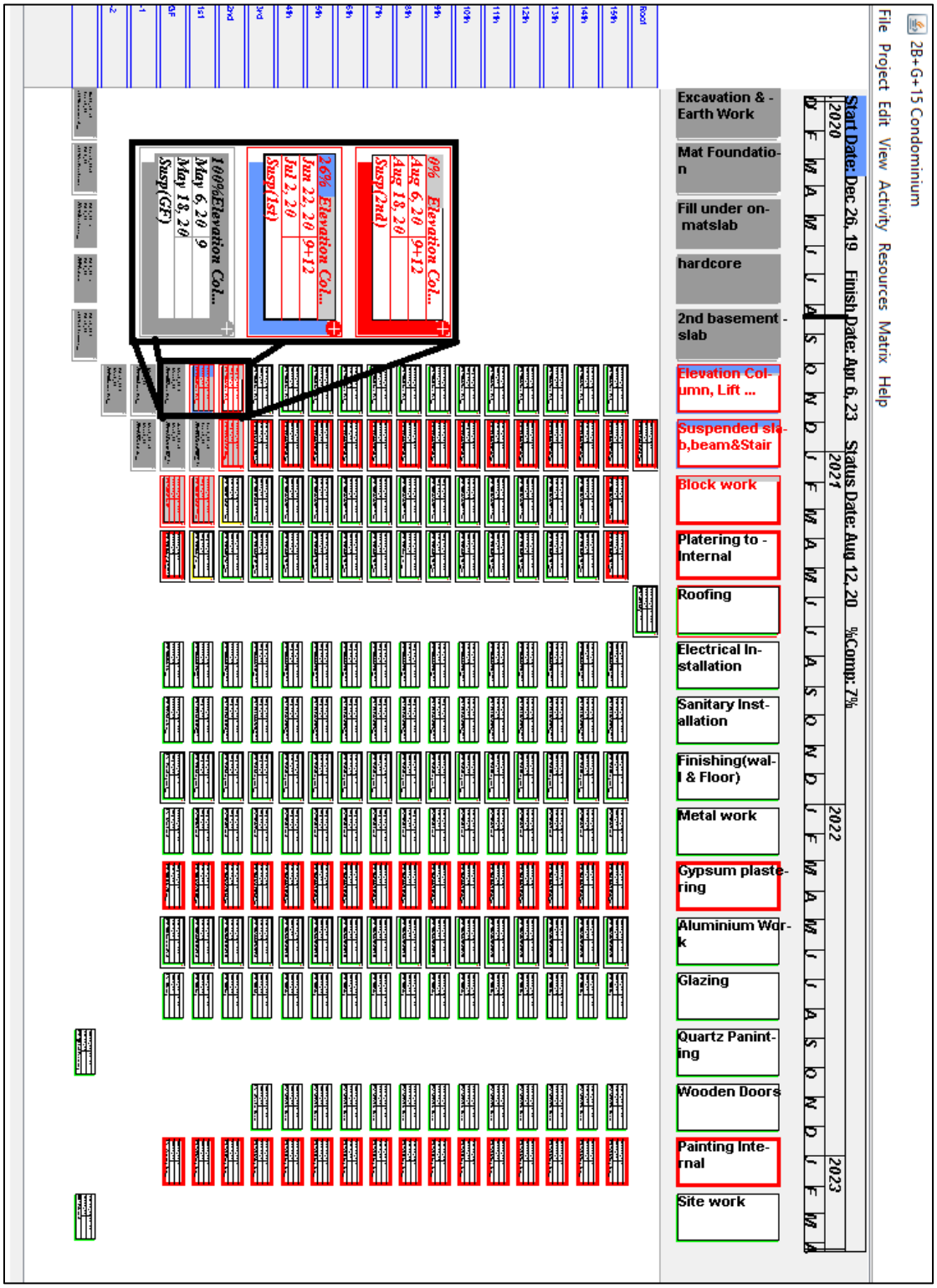


Figure 5-18 Tracking Matrix (zoomed out view)

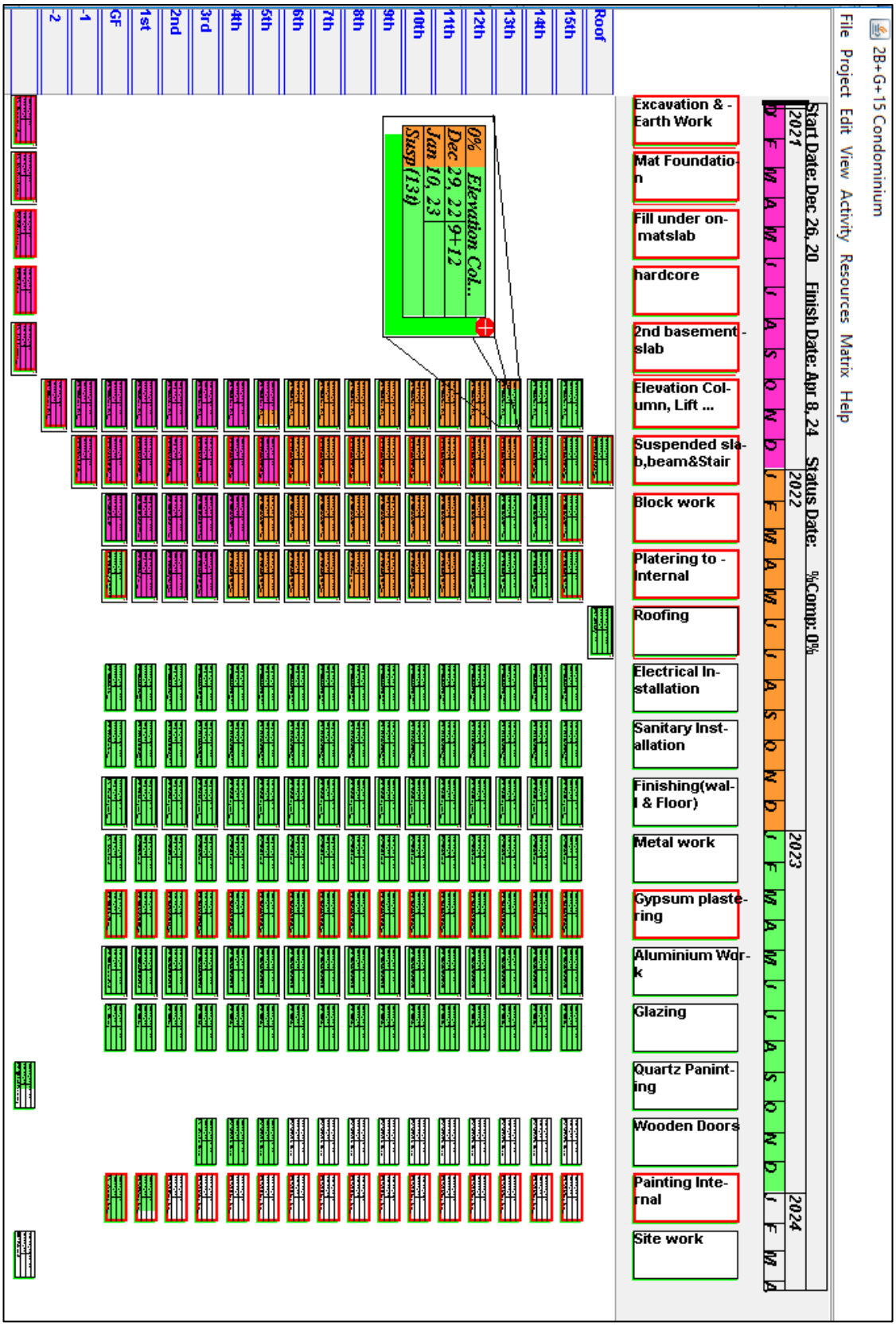


Figure 5-19 Different fill colors for nodes for different period of time

5.6 Comparison of Proposed System with Current Practices

As it is mentioned in the literature review, the most commonly used technique for presentation of schedules is Gantt chart (bar chart). The schedule for the 2B+G+15 condominium is prepared using Microsoft Project and one of the pages is shown in Figure 5-20. The Gantt chart presentation becomes complicated. The logic lines cross with bars and with each other which makes it illegible. The location for an activity on Gantt chart is not easily displayed. If we write the floor numbers, activity name, and dates around the bar, the texts will overlap with each other and becomes still difficult to read. But the matrix presentation is simple. If information for an activity is required, it can be obtained by looking at the activity on the column header and going down to the required floor and the information can be found on the boxes. The progress of the project is shown clearly using the format of the nodes as shown in Figure 5-17. The status of activities can be identified easily. Delayed activities can be distinguished easily by looking at the fill color of the node which is red in this case as shown in Figure 5-17. One can easily know which activities are being performed on which floor by simply looking for nodes with blue shadow. The format can completely be customized. Actual progress with planned progress can be compared with help of the progress bars included in the nodes.

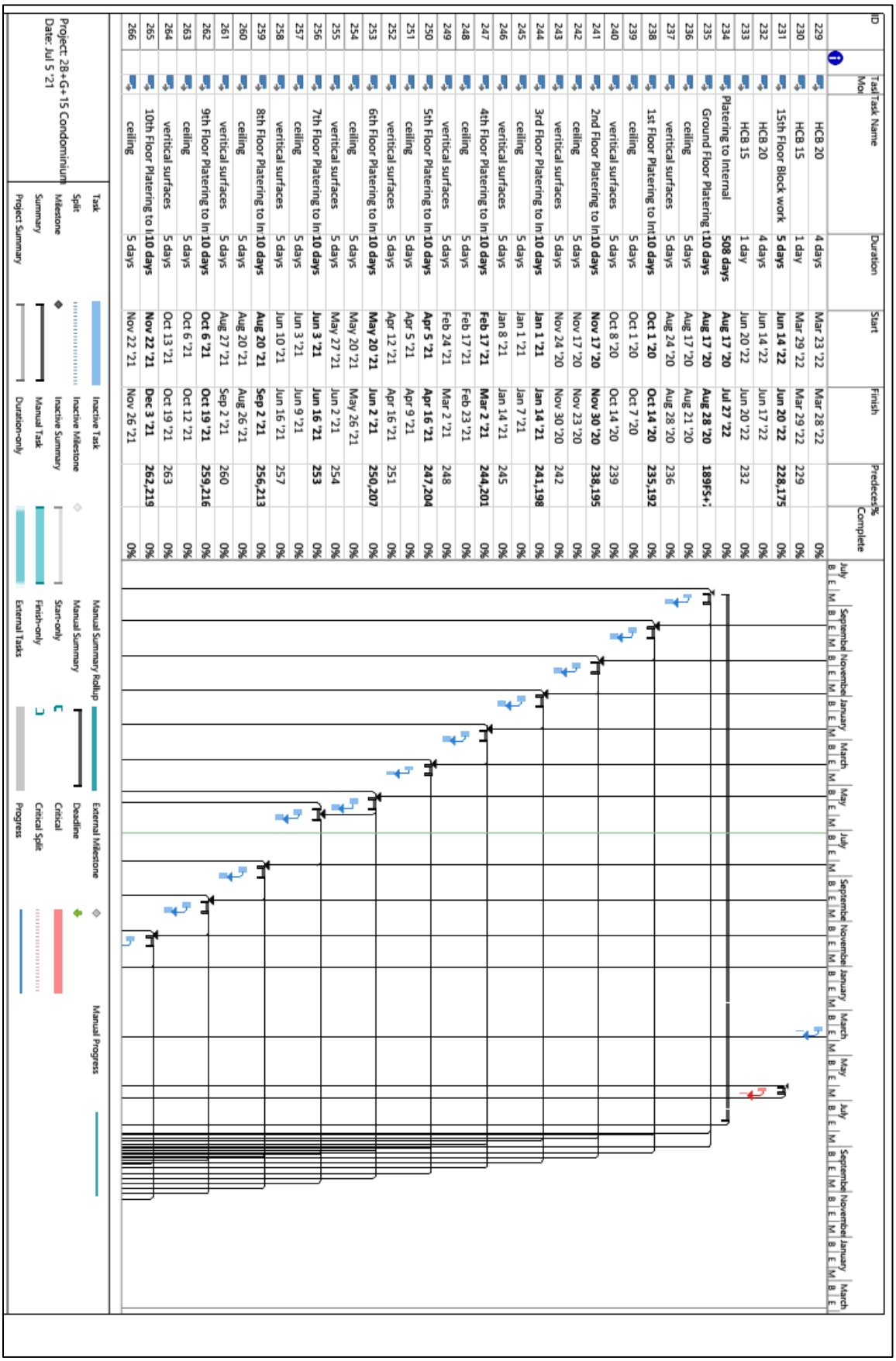


Figure 5-20 Gantt chart for 2B+G+15 condominium project (one of the pages developed using Microsoft Project Software)

5.7 Summery

The schedules for three actual high-rise building projects were prepared using the proposed system. A comparison with Gantt chart which is the common tool used for presentation. The developed schedule was understandable and simple for communication among team members. The progress of activities can be easily identified graphically. In addition to giving the visual status of the project, information such as predecessor, start and finish dates, durations and percent complete are readily available for the user. The schedule has a graphical output that correlates activities with floors. This enables project managers to know where activities are being performed and the status of activities (delayed, completed, ongoing with progress bar) along with their location.

CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The present research study focused on devising a scheduling system by combining the advantages of CPM and matrix scheduling techniques and making some improvements to these methods. The developed system was implemented on a computer and tested with three actual high-rise building projects. The result of the test showed its correctness and effectiveness to attain the objectives which are good presentation of the schedule, correction of duration to account for the effect of bad weather, and maintaining continuity of repetitive activities. Based on the findings of the validation, the following conclusions are made.

The Presentation of the schedule using matrix schedule as shown by test projects is simple and easy to read than Gantt chart. Ordering of activities based on their early start from left to right on the matrix to help users easily understand the order of execution of activities. In each box, various data are shown which enables one to quickly find the information for each activity on a given floor. The user can choose the information shown in each cell of the box. The information can be the start date, finish dates, the actual start date, actual finish dates, duration, the percent complete, planned rate of progress, assigned crew or subcontractor, predecessors, successors, etc., activities are grouped into summaries and the user can expand and collapse to show or hide sub-activities. This enables us to prepare a detailed schedule without losing readability.

The system helps to better present the progress of the project graphically. By using various colors, different line thicknesses, solid or broken lines, the status of each activity is shown giving visual status of the progress of the project. Critical, delayed, completed, and ongoing activities are easily identified with this method. Planned and actual progress bars are included in the box showing the planned and actual percent complete of that activity at the time of the report. This enables an immediate comparison of the planned and actual progress of the activities on any given floor and enables managers to monitor the activities easily and initiate corrective action if required. In addition, the time of execution of each activity can be shown using different colors for different periods such as months. This enables execution personnel to easily visualize the time of execution.

The system helps to prepare a realistic schedule by automatic correction of duration and Maintaining of continuity of repetitive activities which are other objectives of this study. Automatic adjustment of durations to take into account the influence of bad weather helps to develop a realistic schedule. Maintaining continuity of repetitive activities helps to achieve a smooth movement of the crews from floor to floor without idle time for workers and equipment.

6.2 Recommendations

Contractors are recommended to use scheduling techniques that are clear enough to be understood by people at the field level of operation, who may not have sufficient technical knowledge to understand complex schedules and that can effectively portray all the information needed to plan and control the operations. This encourages the majority of the construction personnel to use the schedules for monitoring the progress of the work, which will help in improving productivity and increase the possibility of completing a project within budget and on time.

6.3 Recommendations for Future Study

Topics that are recommended for future work include:

1) Project control techniques such as earned value analysis should be included.
2) Forecasting time to complete repetitive activities based on the performance of completed floor and historical data.
3) An expert system that can recommend possible relationships between activities, work methodologies, optimize project cost and durations.
4) A refinement of the software, in order to make the program fool-proof, optimizing to reduce computation time.

REFERENCES

Ammar, M. A. (2013). LOB and CPM Integrated Method for Scheduling Repetitive Projects. *Journal of Construction Engineering and Management, 139*(1), 44–50.

Arditi, D., & Albulak, M. Z. (1986). Line of balance scheduling in pavement construction. *Journal of Construction Engineering and Management ASCE, 112*(3), 411-24.

Arditi, D., Sikangwan, P., & Tokdemir, O. B. (2002). Scheduling system for high rise building construction. *Construction Management and Economics, 20*(4), 353–364.

Ayalew, T., Dakhli, Z., & Lafhaj, Z. (2016). Assessment on Performance and Challenges of Ethiopian Construction Industry. *Journal of Architecture and Civil Engineering, 2*(11), 01-11.

Baldwin, A., & Bordoli, D. (2014). *A Handbook for Construction Planning and Scheduling.* The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK: John Wiley & Sons, Ltd.

Burke, R. (2003). *Project Management: planning and control techniques.* England: John Wiley & sons Ltd.

Ching, B. (2005). Limitations of Existing Scheduling Tools in Planning Utility Line Construction Projects. MSc thesis, University of Florida, Florida.

Dessalegn, M. (2017). Assessment on the practice of construction work schedule preparation and control of public building projects in Addis Ababa. Msc Thesis. Addis Ababa Science and Technology University. Addis Ababa, Ethiopia.

Eck, D. J. (2011). *Introduction to Programming Using Java.* Geneva, NY.

Elbeltagi, E. (2012). *Lecture Notes on Construction Planning and Scheduling.* Mansoura University.

ElHakeem, A., Elyamany, A., & Eslam, M. (2019). Modeling LOB using CPM Software for Scheduling Repetitive Projects. *Journal of Mechanical and Civil Engineering, 16*(2), 20-27.

Fan, S., & Tserng, H. (2006). Object-oriented scheduling for repetitive projects. *Journal of Construction Engineering and Management, 132*(1), 35-48.

Hagzey, T., & Kamarah, E. (2008). Efficient Repetitive Scheduling for High-Rise Construction. *Journal of Construction Engineering and Management*, 253-264.

Hailemarkos, H. T. (2020). Ethiopian Construction Project Management Maturity Model Determination and Correlational Prediction of Project Success,Doctoral thesis,Walden University.

Harris, F., & McCaffer, R. (1989). *(cited by Arditi et al, 2002). Modern Construction Management.* Collins, London.

Hatem, M. K., Mosa, A. M., & Al-Dahlaki, M. H. (2018). Software for line of balance in the projects of highways. *Journal of Engineering and Sustainable Development, 22*(03), 119-130.

Hazber, M. (2016). Final Report: Java Programming Language. A Simple project to Draw Paint.

Hegazy, T. (2002). Computer-based construction project management. *Prentice-Hall.* Upper Saddle River, N.J.

Kerzner, H. (2009). *Project management : a systems approach to planning, scheduling, and conrolling.* New Jersey: John Wiley & Sons.

Khandelwal, K., & Pumnia, B. (2006). *Project Planning and Control with PERT and CPM* (Forth ed.). New Delhi: Laxmi Publications LTD.

Kilkelly, M. (2016, October 5). *How to Easily Create a Matrix Schedule*. Retrieved from https://archsmarter.com/

Kuhil, A. M., & Seifu, N. (2019). Causes of Delay in Public Building Construction Projects: A Case of Addis Abeba Administration, Ethiopia. *Asian Journal of Managerial Science, 8*(2), 4-9.

Laramee, J. B. (1983). A Planning and Scheduling System for High-Rise Building Construction. Msc Thesis. Concordia University, Montreal, Quebec, Canada.

Liu, M. (2013). Program Evaluation and Review Technique (PERT) in Construction Risk Analysis. *Applied Mechanics and Materials*.

Mubarak, S. (2003). *Construction Project Scheduling and Control.* Prentice Hall, Upper Saddle River, New Jersey.

Naylor, H. F. (1995). *Construction Project Management: Planning and Scheduling.* United States of America: Delmar Publishers.

Pai, S. K., Verguese, P., & Rai, S. (2013). Application of Line of Balance Scheduling Technique (LOBST) for a Real estate sector. *International Journal of Science, Engineering and Technology Research (IJSETR), 2*(1), 82-95.

Pawar, H., Wagh, R., Shahane, S., Kakad, M., Kakad, V., & Bodke, S. S. (2018). Planning and Scheduling of High Rise Building Using Primavera. *International Journal of Scientific Research in Science, Engineering and Technology, 4*(4), 533-537.

Sears, S. K., Sears, G. A., Clough, R. H., Rounds, J. L., & Robert O. Segner, J. (2015). *Construction project management : a practical guide to field construction management* (Sixth ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.

Sinesilassie, E. G., Tabish, S. Z., & Jha, K. N. (2017). Critical factors affecting schedule performance: A case of Ethiopian public construction projects engineers' perspective. *Engineering, Construction and Architectural Management, 24*(5), 757–773.

Subramani, T., & Chinnadurai, K. (2015). Construction Management And Scheduling Of Residential Building Using Primavera. *International Journal of Application or Innovation in Engineering & Management (IJAIEM), 4*(5), 188-198.

Talodhikar, H., & Pataskar, S. V. (2015). Implementation of Line of Balance Method for Scheduling Highrise Building. *Int. Journal of Engineering Research and Applications, 5*(3), 09-12.

Tebeje, Z. (2016). Construction Projects Delay and Their Antidotes: The Case of Ethiopian Construction Sector. *International Journal of Business and Economics Research, 5*, 113-122. doi:10.11648/j.ijber.20160504.16

Tsehaye, A. A. (2017). Advanced scheduling systems, Addis Ababa Univeristy.

Uher, T. (2003). *Programming and scheduling techniques* (1st ed.). UNSW Sydney NSW 2052, Australia: University of New South Wales Press Ltd, University of New South Wales.

Werku, K., & Jha, K. N. (2016). Investigating Causes of Construction Delay in Ethiopian Construction Industries. *Journal of Civil, Construction and Environmental Engineering*, 18-29. doi:10.11648/j.jccee.20160101.13

Wubishet, J. M. (2004). Performances for Public Construction Projects in (Least) Developing Countries, Federal Road and Educational Building Projects in Ethiopia, Doctoral Thesis,NTNU,Trondheim.

Yang, T., & Ioannou, P. (2004). Scheduling system with focus on practical concerns in repetitive projects. *Journal of Construction Mangement an Economics, 22*(6), 619-630.

APPENDIX I SOURCE CODE

This appendix presents 1100 lines of source code from the total of 19,460 lines of code written and only few major classes are shown here.

```
package HRBSS;

import java.beans.PropertyChangeEvent;

import java.beans.PropertyChangeListener;

import java.beans.PropertyChangeSupport;

import java.io.Serializable;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Collections;

import java.util.Date;

import java.util.HashSet;

import java.util.LinkedList;

import javax.swing.JDialog;

import javax.swing.JOptionPane;


/**

 * This class represents high-rise building projects.  It manages resources and crews. it has methods that add, delete resources and Store them in an ArrayList and level resources. It keeps a list of activities and perform CPM calculations to determine project duration, start and finish dates of each activity it stores. This class has a calendar member

 * Field “HRBCalendar” that translates working days into calendar days

* @author TESFA

 */

public class HRBProject implements Serializable {
```

```
    private static final long serialVersionUID = 1L;
    String projectName = "project 1";
    Date StartDate = null;
    Date statusDate = null;
    HRBCalendar CurrentProjectCalendar = HRBCalendar.getStandardCalendar();
    ArrayList<Activity> listOfActivities;
    ArrayList<Unit> units;
    ArrayList<HRBCalendar> calendars;
    TableAndMatrixProperties prop = new TableAndMatrixProperties();
    int maxEFT = 0;
    int statusDateEST = -1;
    int projectFinishTime = 1;
    ArrayList<Crew> crews;
    ArrayList<Resource> resources;
    PropertyChangeSupport propertyChangeSupport = new PropertyChangeSupport(this);
    String currency = "ETB";
    public HRBProject() {
        listOfActivities = new ArrayList<>();
        units = new ArrayList<>();
        calendars = new ArrayList<>();
        calendars.add(HRBCalendar.getStandardCalendar());
        crews = new ArrayList<>();
        resources = new ArrayList<>();
    }
public ArrayList<Object[][]> getResorceUsageReport(Date fromDate, Date toDate,
ArrayList<Resource> resourcesForReport, String dailyOrMonthlyReport, String
MaximumAvearageOrMinimum, boolean includeActivityNames) {
        if (fromDate == null) {
```

```
            fromDate = this.getProjectStartDate();
        }
        if (toDate == null) {
            toDate = getProjectFinishDate();
        }
        Date reportingDate = fromDate;
        Date maxDateInMonth = reportingDate;
        ArrayList<String> headerTimePeriods = new ArrayList<>();
        ArrayList<Integer> activityIndex = new ArrayList<>();
        ArrayList<Integer> resourcesIndex = new ArrayList<>();
        ArrayList<ArrayList<String>> requiredQunatities = new ArrayList<>();
        SimpleDateFormat format = new SimpleDateFormat("MMM d,yyyy");
        SimpleDateFormat formatmax = new SimpleDateFormat("dd/MM/yyyy");
        SimpleDateFormat formatMonth = new SimpleDateFormat("MMM ,yyyy");
// iterate over period of time
        while (HRBCalendar.compareDatesOnly(reportingDate, fromDate) >= 0 &&
HRBCalendar.compareDatesOnly(reportingDate, toDate) <= 0) {
            // time column headers
            if (dailyOrMonthlyReport.equals("Monthly")) {
                headerTimePeriods.add(formatMonth.format(reportingDate));
            } else {
                headerTimePeriods.add(format.format(reportingDate));
            }
// qunantinty of resources
         ArrayList<String> qunatities = new ArrayList<>();
            if (dailyOrMonthlyReport.equals("Monthly")) {
                Calendar c = Calendar.getInstance();
```

```
            c.setTime(reportingDate);
            try {
               int maxDayOfMonth =
c.getActualMaximum(Calendar.DAY_OF_MONTH);
               int month = reportingDate.getMonth() + 1;
               int year = reportingDate.getYear() + 1900;
               maxDateInMonth = formatmax.parse(maxDayOfMonth + "/" + month + "/"
+ year);
               if (HRBCalendar.compareDatesOnly(maxDateInMonth, toDate) > 0) {
                  maxDateInMonth = toDate;
               }
            } catch (Exception e) {
            }
         } else {
            maxDateInMonth = reportingDate;
         }
         for (int i = 0; i < resourcesForReport.size(); i++) {
          Collections.sort(listOfActivities, new ComparatorEST());
            // qunaities from all activities
            double requiredQunatityTotal = 0;
            int noOfOccurancesTotal = 0;
            double maxQunatityTotal = 0;
            double minQunatityTotal = 0;
            //if(includeActivityNames)
            for (int j = 0; j < listOfActivities.size(); j++) {
// qunatities for one activity
               double requiredQunatity = 0;
               int noOfOccurances = 0;
```

```
                double maxQunatity = 0;

                double minQunatity = 0;

                Date reportDate = reportingDate;

                double activityRequiredQunatity = 0;

                if (HRBCalendar.compareDatesOnly(toDate,
listOfActivities.get(j).getStartDate()) < 0) {

                   break;

                } else if ((HRBCalendar.compareDatesOnly(reportingDate,
listOfActivities.get(j).getStartDate()) <= 0 &&
HRBCalendar.compareDatesOnly(maxDateInMonth,
listOfActivities.get(j).getStartDate()) >= 0)

                      || HRBCalendar.compareDatesOnly(reportingDate,
listOfActivities.get(j).getFinishDate()) <= 0) {

//interate over the period

                   Calendar cal = Calendar.getInstance();

                   cal.setTime(maxDateInMonth);

                   cal.add(Calendar.DATE, 1);

                   Date dateAfterMxDate = cal.getTime();

                   while (HRBCalendar.compareDatesOnly(reportDate, dateAfterMxDate) <
0) {

                      // add to qunatity from current activity

                      activityRequiredQunatity =
listOfActivities.get(j).getRequiredQunatityForResource(reportDate,
resourcesForReport.get(i));

            requiredQunatity = requiredQunatity + activityRequiredQunatity;

         //       System.out.println("activityRequiredQunatity
"+listOfActivities.get(j).activityName+"       "+requiredQunatity);

               if(requiredQunatity>0){

                  noOfOccurances++;

                }

                   maxQunatity = Math.max(maxQunatity, activityRequiredQunatity);
```

```
                    if(minQunatity==0){
                        minQunatity=activityRequiredQunatity;
                    }
                    minQunatity = Math.min(minQunatity, activityRequiredQunatity);
                    reportDate =
getProjectCalendar().getWorkingDateafterDate(reportDate);
                }
                // add to total from all activities
                requiredQunatityTotal = requiredQunatityTotal + requiredQunatity;
            //          System.out.println("  requiredQunatityTotal  "+
requiredQunatityTotal);
                if(requiredQunatity>0){
                    noOfOccurancesTotal++;
                }
                maxQunatityTotal = Math.max(maxQunatityTotal, maxQunatity);
                if(minQunatityTotal==0){
                    minQunatityTotal=minQunatity;
                }
                minQunatityTotal = Math.min(minQunatityTotal, minQunatity);
            } else {
            }
            if (includeActivityNames) {
                if (!activityIndex.contains(j)) {
                    activityIndex.add(j);
                }
                if (!resourcesIndex.contains(i)) {
                    resourcesIndex.add(i);
                }
```

```
            if (noOfOccurancesTotal == 0) {
                noOfOccurancesTotal = 1;
            }
            if (MaximumAvearageOrMinimum.equals("Maximum")) {
                qunatities.add(maxQunatityTotal + "");
            } else if (MaximumAvearageOrMinimum.equals("Minimum")) {
                qunatities.add(minQunatityTotal + "");
            } else if (MaximumAvearageOrMinimum.equals("Average")) {
                qunatities.add(requiredQunatityTotal / noOfOccurancesTotal + "");
            } else if(MaximumAvearageOrMinimum.equals("Total")){
             qunatities.add(requiredQunatityTotal + "");
            }
             else {
        }
        }
    }
    if (!includeActivityNames) {
    if (!resourcesIndex.contains(i)) {
            resourcesIndex.add(i);
        }
        if (noOfOccurancesTotal == 0) {
            noOfOccurancesTotal = 1;
        }
        if (MaximumAvearageOrMinimum.equals("Maximum")) {
            qunatities.add(i, maxQunatityTotal + "");
        } else if (MaximumAvearageOrMinimum.equals("Minimum")) {
            qunatities.add(i, minQunatityTotal + "");
```

```
                } else if (MaximumAvearageOrMinimum.equals("Average")) {
                    qunatities.add(i, requiredQunatityTotal / noOfOccurancesTotal + "");
                }
                 else if(MaximumAvearageOrMinimum.equals("Total")){
                     qunatities.add(requiredQunatityTotal + "");
                   }
                else {
                }
            }
        }
        requiredQunatities.add(qunatities);
        reportingDate =
getProjectCalendar().getWorkingDateafterDate(maxDateInMonth);
    }
// preparing report in [][]
    Object[][] report;
    Object[][] header;
    ArrayList<Object[][]> reportWithHeader = new ArrayList<>();
    int columns;
    int rows = resourcesIndex.size();
    int ativityNamecolumn = 0;
    if (includeActivityNames) {
        ativityNamecolumn = 1;
    }
    columns = requiredQunatities.size() + 2 + ativityNamecolumn;
    report = new Object[rows][columns];
    header = new Object[columns][1];
```

```
// headers
        header[0][0] = "Resource Name";
        header[1][0] = "Unit";
        if (includeActivityNames) {
            header[2][0] = "Activity Name";
        for (int i = 0; i < activityIndex.size(); i++) {
                report[i][2] = listOfActivities.get(activityIndex.get(i)).getActivityName();
       }
        }
        for (int i = 0; i < headerTimePeriods.size(); i++) {
            header[i + 2 + ativityNamecolumn][0] = headerTimePeriods.get(i);
      }
        for (int i = 0; i < resourcesIndex.size(); i++) {
            report[i][0] = resourcesForReport.get(resourcesIndex.get(i)).getResourceName();
        }
        for (int i = 0; i < resourcesIndex.size(); i++) {
            report[i][1] = resourcesForReport.get(resourcesIndex.get(i)).getUnit();
        }
        for (int i = 0; i < requiredQunatities.size(); i++) {
            for (int j = 0; j < requiredQunatities.get(i).size(); j++) {
                report[j][i + 2 + ativityNamecolumn] = requiredQunatities.get(i).get(j);
            }
        }
        reportWithHeader.add(header);
        reportWithHeader.add(report);
        return reportWithHeader;
    }
```

```
  public void levelResource(ArrayList<Resource> resourcesToBeLevelled, boolean
levelUsingTotalFloatOnly) {

      Date startDateForLevelling = this.getProjectStartDate();

      Date dateNowLevelling = startDateForLevelling;

      Date finishDateForLevelling = this.getProjectFinishDate();

      while (HRBCalendar.compareDatesOnly(dateNowLevelling,
finishDateForLevelling) <= 0) {

          ArrayList<ArrayList<Activity>> requiringActivitiesOfAllResources = new
ArrayList<>();

              ArrayList<Activity> commonActivities = new ArrayList<>();

            for (int i = 0; i < resourcesToBeLevelled.size(); i++) {

              double requiredQunatity = 0;

              ArrayList<Activity> requiringActivities = new ArrayList<>();

              ArrayList<Activity> activitiesRegisteredToResource =
resourcesToBeLevelled.get(i).usingActivities;

              for (int j = 0; j < activitiesRegisteredToResource.size(); j++) {

                 if (listOfActivities.contains(activitiesRegisteredToResource.get(j))) {

                     double requiredQunatityForActivity =
activitiesRegisteredToResource.get(j).getRequiredQunatityForResource(dateNowLevelli
ng, resourcesToBeLevelled.get(i));

                     requiredQunatity = requiredQunatity + requiredQunatityForActivity;

                     if (requiredQunatityForActivity > 0) {

                        requiringActivities.add(activitiesRegisteredToResource.get(j));

                           for(int y=0;y<requiringActivitiesOfAllResources.size();y++){


if(requiringActivitiesOfAllResources.get(y).contains(activitiesRegisteredToResource.get
(j))&&!commonActivities.contains(activitiesRegisteredToResource.get(j))){

              commonActivities.add(activitiesRegisteredToResource.get(j));

                     }

              }

                     }
```

```
                }
            }
        requiringActivitiesOfAllResources.add(requiringActivities);
         }
        while (true) {
            ArrayList<Activity>activitiesToBeMoved = new ArrayList<>();
              for (int i = 0; i < resourcesToBeLevelled.size(); i++) {

activitiesToBeMoved.addAll(activityToMoveToResolveOverallocation(resourcesToBeL
evelled.get(i), requiringActivitiesOfAllResources.get(i),
dateNowLevelling,levelUsingTotalFloatOnly));
              }
              if(!activitiesToBeMoved.isEmpty()){
            boolean commonActivitiesFound = false;
                for(int i = 0; i < commonActivities.size(); i++){
               if(activitiesToBeMoved.contains(commonActivities.get(i))){

commonActivities.get(i).setResourcelevelStart(getProjectCalendar().getWorkingDateafte
rDate(dateNowLevelling));
                commonActivitiesFound = true;
                }
            }
            if(!commonActivitiesFound){
            for(int i = 0; i < activitiesToBeMoved.size(); i++){

activitiesToBeMoved.get(i).setResourcelevelStart(getProjectCalendar().getWorkingDate
afterDate(dateNowLevelling));
            }
              }
              }else{
              break;
```

```
                }

            this.modifiedForwardPass();

            this.modifiedBackwardPass();

        }

       dateNowLevelling =
getProjectCalendar().getWorkingDateafterDate(dateNowLevelling);

           finishDateForLevelling = this.getProjectFinishDate();

       }

   }

  public ArrayList<Activity> activityToMoveToResolveOverallocation(Resource r,
ArrayList<Activity> activitiesRequiringTheResource, Date date, boolean useFloatOnly)
{

     ArrayList<Activity> activityToMoveToResolveOverallocation = new ArrayList<>();

     double maximumAvailableUnit = r.getMaximumAvailableUnits();

     double requiredQunatity = 0;

     boolean allAreCritical =false;

       if(useFloatOnly){

          Collections.sort(activitiesRequiringTheResource, new ComparatorTotalFloat());

allAreCritical = Collections.max(activitiesRequiringTheResource, new
ComparatorTotalFloat()).getTotalFloat()==0;

    for(int i=0;i<activitiesRequiringTheResource.size();i++){

   }

       }else{

        Collections.sort(activitiesRequiringTheResource, new
ComparatorDelayDaysIfMovedDescending(getProjectCalendar().getWorkingDateafterD
ate(date)));

           for(int i=0;i<activitiesRequiringTheResource.size();i++){

    }

       }

       if(!allAreCritical){
```

```
        for (int j = 0; j < activitiesRequiringTheResource.size(); j++) {

            double activityRequiredQty =
activitiesRequiringTheResource.get(j).getRequiredQunatityForResource(date, r);

                requiredQunatity= requiredQunatity + activityRequiredQty;

                if(j==0){

            if(activityRequiredQty> maximumAvailableUnit){

             }

            }

            else{

                if (requiredQunatity > maximumAvailableUnit && activityRequiredQty>0) {

activityToMoveToResolveOverallocation.add(activitiesRequiringTheResource.get(j));

             }

       }}}

        return activityToMoveToResolveOverallocation;

    }

public void modifiedForwardPass() {

        ArrayList<Activity> calculatedActivities = new ArrayList<>();

        Collections.sort(listOfActivities, new ComparatorForFwPass());


        while (!calculatedActivities.containsAll(listOfActivities)) {

            for (int i = 0; i < listOfActivities.size(); i++) {

                if (!calculatedActivities.contains(listOfActivities.get(i))) {

                    if (listOfActivities.get(i).getAllPredecessorActivities().isEmpty() &&
!listOfActivities.get(i).isSummery()) {

                        listOfActivities.get(i).calculateEarlyDates();

                        calculatedActivities.add(listOfActivities.get(i));

                    } else if
(calculatedActivities.containsAll(listOfActivities.get(i).getAllPredecessorActivities())
&& !listOfActivities.get(i).isSummery()) {
```

```
                listOfActivities.get(i).calculateEarlyDates();

                calculatedActivities.add(listOfActivities.get(i));

            } else if (listOfActivities.get(i).isSummery() &&
calculatedActivities.containsAll(listOfActivities.get(i).subActivitiesIn)) {

                listOfActivities.get(i).calculateEarlyDates();

                if (listOfActivities.get(i).isContinuos()) {

                    ArrayList<Activity> allSubTasks =
listOfActivities.get(i).getAllSubTasks();

                    int minIdexOfSubActs =
calculatedActivities.indexOf(allSubTasks.get(0));

                    Activity firstActivity = allSubTasks.get(0);

                    for (Activity a : allSubTasks) {

                        int index = calculatedActivities.indexOf(a);

                        if (minIdexOfSubActs >= index) {

                            minIdexOfSubActs = index;

                            firstActivity = a;

                        }

                    }

                    int sumActEFT = listOfActivities.get(i).getEFT();

                    ArrayList<Integer> ests = new ArrayList<>();

                    for (Activity a : allSubTasks) {

                        int start = a.getEST() + 1;

                        int end = a.getEFT();

                        for (int i2 = start; i2 < end + 1; i2++) {

                            if (!ests.contains(i2)) {

                                ests.add(i2);

                            }

                        }
```

```
                    }
                    Collections.sort(ests);
                    int indexAtListOfActs = listOfActivities.indexOf(firstActivity);
                    for (int c = indexAtListOfActs; c < i; c++) {
                        Activity a = listOfActivities.get(c);
                        if (allSubTasks.contains(a)) {
                            int index = ests.indexOf(a.getEST() + 1);

                            int noOfUninteruptedDates = ests.size() - index;
                            int noOfTotalDates = sumActEFT - a.getEST();
                            int noOfDatesToPush = noOfTotalDates -
noOfUninteruptedDates;
                            if (a.isSummery()) {
                                a.calculateEarlyDates();
                            } else {
                                a.calculateEarlyDatesForCont(noOfDatesToPush);
                            }
                        } else {
                            a.calculateEarlyDates();
                        }
                    }
                }
                listOfActivities.get(i).calculateEarlyDates();
                calculatedActivities.add(listOfActivities.get(i));
            }
        }
    }
```

```
        }

        if (!listOfActivities.isEmpty()) {

            projectFinishTime = Collections.max(listOfActivities, new
ComparatorEFT()).getEFT();

        }

        Collections.sort(listOfActivities, new ComparatorForBwPass());

    }


    public void modifiedBackwardPass() {


        ArrayList<Activity> BackwardCalculatedActivities = new ArrayList<>();

        while (!BackwardCalculatedActivities.containsAll(listOfActivities)) {

            for (int i = listOfActivities.size() - 1; i > -1; i--) {

                if (!BackwardCalculatedActivities.contains(listOfActivities.get(i))) {

                    if (listOfActivities.get(i).getAllSuccessorsExecludingFinished().isEmpty()
&& !listOfActivities.get(i).isSummery()) {

                        listOfActivities.get(i).calculateLateDates();

                        BackwardCalculatedActivities.add(listOfActivities.get(i));

                    } else if
(BackwardCalculatedActivities.containsAll(listOfActivities.get(i).getAllSuccessorActivi
tiesExecludingFinished()) && !listOfActivities.get(i).isSummery()) {

                        listOfActivities.get(i).calculateLateDates();

                        BackwardCalculatedActivities.add(listOfActivities.get(i));

                    } else if (listOfActivities.get(i).isSummery() &&
BackwardCalculatedActivities.containsAll(listOfActivities.get(i).subActivitiesIn)) {

                        listOfActivities.get(i).calculateLateDates();

                        BackwardCalculatedActivities.add(listOfActivities.get(i));

                    }

                }
```

```
            }
        }
    }
public void removeActivity(Activity acti) {
        if (acti != null) {
            ArrayList<Activity> toBeremoved = new ArrayList<>();
            toBeremoved.add(acti);
            while (!toBeremoved.isEmpty()) {
                Activity activity = toBeremoved.get(0);
                if (activity.subToActivity != null) {
                    activity.subToActivity.removeSubactivity(activity);
                }
                for (Activity act : activity.getPredecessorActivitiesIn()) {
                    activity.removePredecessor(act);
                }
                for (Activity act : activity.getSuccessorActivitiesIn()) {
                    act.removePredecessorOnly(activity);
                }
                if (activity.isSummery()) {
                    ArrayList<Activity> subActivities = activity.getSubActivities();
                    for (Activity act : subActivities) {
                        toBeremoved.add(act);
                    }
                }
                listOfActivities.remove(activity);
                arrangedList.remove(activity);
                toBeremoved.remove(activity);
```

```
            propertyChangeSupport.firePropertyChange("removeActivity", activity, null);


        }
    }
}
/**
 * this class represents construction activities. it stores resources
 * assigned to the activities in an ArrayList it has many methods for
 * manipulating activity data
 */
public class Activity implements Serializable {
private static final long serialVersionUID = 2L;
    int activityId;
    String activityShortName = "";
    String activityName = "";
    String remark = "-1";
    int column = -1;
    boolean hiden = false;
    double totalQuantity;
    double cost;
    double ActualCost;
    boolean autoSchedule = true;
    boolean adjustDuration = false;
    ArrayList<Resources> activityResources = new ArrayList<>();
    ArrayList<DelayImpactFactors> delayImpactFactors = new ArrayList<>();
    Date earlyStartDate;
    Date manualStartDate;
```

```
    Date manualFinishDate;
    Date mustStartDate;
    Date mustFinishDate;
    Date earlyFinishDate;
    Date lateStartDate;
    Date lateFinishDate;
    Date resourcelevelStart;
    Date resourcelevelledFinish;
    Date baselineStart;
    Date baselineFinish;
    int baselineDuration;
    double baselineQuantity;
    double baselineCost;
    boolean expanded;
    boolean continuos;
    ArrayList<Activity> subActivitiesIn = new ArrayList<>();
    public Activity subToActivity = null;
   ArrayList<Predecessor> predecessors = new ArrayList<>();
    ArrayList<Successor> successorsIn = new ArrayList<>();
    Unit unit = null;
public void addSubactivity(Activity activity) {
        if (!subActivitiesIn.contains(activity)) {
            if (!loopDetect(this, activity)) {
                if (activity.subToActivity != null) {
                    activity.subToActivity.removeSubactivity(activity);
                }
                this.firePropertyChange("addSubactivity", null, activity);
```

```
                this.subActivitiesIn.add(activity);
                activity.setSubTo(this);
                if (this.unit != null) {
                    activity.setUnit(unit);
                }
                calculateEarlyDates();
                activity.calculateEarlyDates();
            } else {
                JOptionPane.showMessageDialog(null, "activity: " + "'" +
activity.activityName + "' " + " can't be sub-activity. it's in the same activity chain with
the summery!");
            }
        }
    }
    public void removeSubactivityOnly(Activity activity) {
        if (subActivitiesIn.contains(activity)) {
            this.firePropertyChange("removeSubactivity", activity, null);
            this.subActivitiesIn.remove(activity);
            calculateEarlyDates();
            activity.calculateEarlyDates();
        }
    }

public void setActualDuration(int actualDuration, JDialog dialog) {
        if (actualDuration > 0) {
            if (!isSummery()) {
                if (actualStartDate == null) {
```

```
                this.firePropertyChange("actualStartDate", actualStartDate,
this.getStartDate());

                this.actualStartDate = this.getStartDate();

                this.actualFinishDate =
getProjectCalendar().getFinishDates(actualDuration, actualStartDate);



            } else {

                this.actualFinishDate =
getProjectCalendar().getFinishDates(actualDuration, actualStartDate);

            }

            percentComplete = 100;

            this.executedQty = this.totalQuantity;

        } else if (subActivitiesIn.size() == 1) {

            subActivitiesIn.get(0).setActualDuration(actualDuration, dialog);

        } else {

            JOptionPane.showMessageDialog(dialog, "can't set actual duration for a
summery, enter for each sub-activities!");

        }

    }

  public void setPercentComplete(int percentComplete) {

        int oldPercentComplete = this.percentComplete;

        Date oldActualStart = this.actualStartDate;

        Date oldActualFinish = this.actualFinishDate;

        double oldExecutedqty = this.executedQty;

        if (percentComplete < 0) {

            percentComplete = 0;

        }

        if (percentComplete > 100) {
```

```
        percentComplete = 100;
    }
    if (this.percentComplete != percentComplete) {
        if (isSummery() && percentComplete == 100) {
            for (int i = 0; i < subActivitiesIn.size(); i++) {
                subActivitiesIn.get(i).setPercentComplete(percentComplete);
            }
        } else {
            this.percentComplete = percentComplete;
            if (percentComplete == 0) {
                this.actualStartDate = null;
                this.actualFinishDate = null;
            } else if (percentComplete > 0) {
                if (this.actualStartDate == null) {
                    this.actualStartDate = this.earlyStartDate;
                }
                if (percentComplete == 100 & this.actualFinishDate == null) {
                    this.actualFinishDate = this.earlyFinishDate;
                }
                if (percentComplete < 100 && this.actualFinishDate != null) {
                    this.actualFinishDate = null;
                }
                if (this.totalQuantity != 0) {
                    this.executedQty = this.totalQuantity * percentComplete / 100;
                }
            }
        }
```

```
        }
    }
public void setActualStartDate(Date date) {

        if (!isSummery()) {
            Date oldActualStart = this.actualStartDate;
            Date oldActualFinish = this.actualFinishDate;
            if (date == null) {
                this.actualStartDate = null;
                this.percentComplete = 0;
                this.actualFinishDate = null;
            } else {
                if (HRBCalendar.compareDatesOnly(date, getProjectStartDate()) >= 0) {
                    if (actualFinishDate == null) {
                        this.actualStartDate = date;

                    } else {
                        if (HRBCalendar.compareDatesOnly(date, actualFinishDate) > 0) {
                            this.actualStartDate = date;
                            this.actualFinishDate = this.actualStartDate;
                            this.EST =
(getProjectCalendar().getESTForActualStartDate(actualStartDate,
getProjectStartDate()));
                            this.setLFT(EST);
                        } else {
                            this.actualStartDate = date;
                            this.EST =
(getProjectCalendar().getESTForActualStartDate(actualStartDate,
getProjectStartDate()));
```

```
                    }

                }

                this.firePropertyChange("actualStartDate", oldActualStart,
this.actualStartDate);

                this.firePropertyChange("actualFinishDate", oldActualFinish,
this.actualFinishDate);



            } else {

                JOptionPane.showMessageDialog(null, "Actual Start Date you entered is
before project start date, Change the date!");

            }

        }



    } else {

        if (subActivitiesIn.size() == 1) {

            subActivitiesIn.get(0).setActualStartDate(date);

        }

    }

}



public void setActualFinishDate(Date date) {

    Date oldActualStart = this.actualStartDate;

    Date oldActualFinish = this.actualFinishDate;

    int oldPercentComplete = this.percentComplete;

    if (this.actualFinishDate != date) {

        if (date == null) {

            this.actualFinishDate = null;

            if (percentComplete == 100) {
```

```
                percentComplete = 99;

                setActualDuration();

            }

        } else if (HRBCalendar.compareDatesOnly(date, getProjectStartDate()) >= 0)
{

            if (!isSummery()) {

                if (actualStartDate == null) {

                    this.actualFinishDate = date;

                    percentComplete = 100;

                    this.actualStartDate = actualFinishDate;

                    this.EST =
(getProjectCalendar().getESTForActualStartDate(actualStartDate,
getProjectStartDate()));

                    setActualDuration();

                } else {

                    if (HRBCalendar.compareDatesOnly(actualStartDate, date) > 0) {

                        this.actualStartDate = date;

                        this.actualFinishDate = date;

                        this.EST =
(getProjectCalendar().getESTForActualStartDate(actualStartDate,
getProjectStartDate()));

                        percentComplete = 100;

                        setActualDuration();

                    } else {

                        this.actualFinishDate = date;

                        percentComplete = 100;

                        setActualDuration();

                    }

                }
```

```
                } else {
                    for (Activity act : subActivitiesIn) {
                        act.setActualFinishDate(date);
                    }
                }
            } else {
                JOptionPane.showMessageDialog(null, "Actual Finish Date you entered is
before project start date, Change the date!");
            }
        }
        this.firePropertyChange("percentComplete", oldPercentComplete,
this.percentComplete);
        this.firePropertyChange("actualStartDate", oldActualStart, this.actualStartDate);
        this.firePropertyChange("actualFinishDate", oldActualFinish,
this.actualFinishDate);
    }
  public void setActualFinishDate(Date date) {
        Date oldActualStart = this.actualStartDate;
        Date oldActualFinish = this.actualFinishDate;
        int oldPercentComplete = this.percentComplete;
        if (this.actualFinishDate != date) {
            if (date == null) {
                this.actualFinishDate = null;
                if (percentComplete == 100) {
                    percentComplete = 99;
                    setActualDuration();
                }
            } else if (HRBCalendar.compareDatesOnly(date, getProjectStartDate()) >= 0)
{
```

```
            if (!isSummery()) {
                if (actualStartDate == null) {
                    this.actualFinishDate = date;
                    percentComplete = 100;
                    this.actualStartDate = actualFinishDate;
                    this.EST =
(getProjectCalendar().getESTForActualStartDate(actualStartDate,
getProjectStartDate()));
                    setActualDuration();
                } else {
                    if (HRBCalendar.compareDatesOnly(actualStartDate, date) > 0) {
                        this.actualStartDate = date;
                        this.actualFinishDate = date;
                        this.EST =
(getProjectCalendar().getESTForActualStartDate(actualStartDate,
getProjectStartDate()));
                        percentComplete = 100;
                        setActualDuration();
                    } else {
                        this.actualFinishDate = date;
                        percentComplete = 100;
                        setActualDuration();
                    }
                }
            } else {
                for (Activity act : subActivitiesIn) {
                    act.setActualFinishDate(date);
                }
            }
```

```
            } else {

                JOptionPane.showMessageDialog(null, "Actual Finish Date you entered is
before project start date, Change the date!");

            }

        }

        this.firePropertyChange("percentComplete", oldPercentComplete,
this.percentComplete);

        this.firePropertyChange("actualStartDate", oldActualStart, this.actualStartDate);

        this.firePropertyChange("actualFinishDate", oldActualFinish,
this.actualFinishDate);

    }

public int getSumOfDurations() {

        int sumOfDurations = 0;

        if (isSummery()) {

            for (int i = 0; i < this.subActivitiesIn.size(); i++) {

                sumOfDurations = sumOfDurations +
subActivitiesIn.get(i).getSumOfDurations();

            }

        } else {

            sumOfDurations = getAdjustedDuration();

        }

        return sumOfDurations;

    }


    public double getExecutedQty() {

        if (!isSummery()) {

            return this.executedQty;

        }

        return 0;
```

```
    }


    public double getPlannedExecutedDuration() {
        double plannedExecutedDuration = 0;
        if (isSummery()) {
            for (int i = 0; i < this.subActivitiesIn.size(); i++) {
                plannedExecutedDuration = plannedExecutedDuration +
subActivitiesIn.get(i).getPlannedExecutedDuration();
            }
        } else {
            plannedExecutedDuration = (double) (getAdjustedDuration() *
this.getPlannedPercentComplete()) / 100;
        }
        return plannedExecutedDuration;
    }
    public double getExecutedDuration() {
        double executedDuration = 0;
        if (isSummery()) {
            for (int i = 0; i < this.subActivitiesIn.size(); i++) {
                executedDuration = executedDuration +
subActivitiesIn.get(i).getExecutedDuration();
            }
        } else {
            executedDuration = (double) (getAdjustedDuration() * this.percentComplete) /
100;
        }
        return executedDuration;
    }
public int setColumn(int column) {
```

```
        int initialColumn = column;
        int sameColumn = 0;
        boolean firstDtColumn = true;
        if (isSummery()) {
            if (isExpanded()) {
                this.column = initialColumn;
                Collections.sort(subActivitiesIn, new ComparatorEST());
                for (int i = 0; i < subActivitiesIn.size(); i++) {
                    if (subActivitiesIn.get(i).isVisibleOnMatrix() &&
subActivitiesIn.get(i).getRow() != -1) {
                        if (subActivitiesIn.get(i).getRow() == this.getRow()) {
                            if (firstDtColumn) {
                                firstDtColumn = false;
                                initialColumn = subActivitiesIn.get(i).setColumn(this.column);
                            } else {
                                initialColumn = subActivitiesIn.get(i).setColumn(initialColumn);
                            }
                        } else if (subActivitiesIn.get(i).getRow() != this.getRow()) {
                            if (sameColumn == 0) {
                                sameColumn = initialColumn;
                                initialColumn =
Integer.max(subActivitiesIn.get(i).setColumn(initialColumn), initialColumn);
                            } else {
                                initialColumn =
Integer.max(subActivitiesIn.get(i).setColumn(sameColumn), initialColumn);
                            }
                        }
                    }
```

```
                }
            } else {
                this.column = initialColumn;
                initialColumn++;
            }
        } else {
            this.column = initialColumn;
            initialColumn++;
        }
        return initialColumn;
    }

    public int getEndColumn() {
        int endColumn = 0;
        if (!hiden) {
            if (isSummery()) {
                if (isExpanded()) {
                    ArrayList<Activity> subActivitiesOnSameMatrixColumn = new
ArrayList<>();
                    ArrayList<Activity> subActivitiesOnDifferentMatrixColumn = new
ArrayList<>();
                    for (Activity subActivity : subActivitiesIn) {
                        if (subActivity.isVisibleOnMatrix() && subActivity.getRow() != -1) {
                            if (subActivity.getRow() == this.getRow()) {
                                subActivitiesOnDifferentMatrixColumn.add(subActivity);
                            } else if (subActivity.getRow() != this.getRow()) {
                                subActivitiesOnSameMatrixColumn.add(subActivity);
                            }
```

```
                }

            }

            int maxEndColumnFromSubOnSameColumn = 0;

            if (!subActivitiesOnSameMatrixColumn.isEmpty()) {

                maxEndColumnFromSubOnSameColumn =
Collections.max(subActivitiesOnSameMatrixColumn, new
ComparatorEndColumn()).getEndColumn();

            }

            int sumOfEndColumnFromSubOndifferentColumn = 0;

            for (Activity subActivity : subActivitiesOnDifferentMatrixColumn) {

                sumOfEndColumnFromSubOndifferentColumn =
sumOfEndColumnFromSubOndifferentColumn + subActivity.getEndColumn();

            }

            endColumn = sumOfEndColumnFromSubOndifferentColumn +
maxEndColumnFromSubOnSameColumn;

        } else {

            endColumn = 1;

        }

    } else {

        endColumn = 1;

    }

}

return endColumn;

}

public ArrayList<Resources> getActivityResources() {

    ArrayList<Resources> getActivityResources = new ArrayList<>();

    getActivityResources.addAll(this.activityResources);

    if (this.assignedCrew != null) {

        getActivityResources.addAll(assignedCrew.getCrewResources());
```

```
        }
        if (isSubActivity()) {
            getActivityResources.addAll(this.getSubToActivity().getActivityResources());
        }
        return getActivityResources;
    }

    public double getRequiredQunatityForResource(Date dateResourceRequired,
Resource resource) {
        if (this.actualStartDate == null && !isSummery()) {
            double requiredQtyOfResource = 0;
            for (int i = 0; i < getActivityResources().size(); i++) {
                if (getActivityResources().get(i).resource == resource) {
                    requiredQtyOfResource = getActivityResources().get(i).NoorQuantity;
                    break;
                }
            }

            if (isActivityExecutedOnDate(dateResourceRequired)) {
                return requiredQtyOfResource;
            }
        }
        return 0;
    }
  public int getTotalFloat() {
        return getLST() - getEST();
    }
```

```
    public boolean isContinuos() {
        if (isSubActivity()) {
            return subToActivity.isContinuos();
        } else if (isSummery()) {
            return continuos;
        }
        return false;
    }

    public void setContinous(boolean isContinous, JDialog dialog) {
        if (isSummery()) {
            this.firePropertyChange("continuos", this.continuos, isContinous);
            this.continuos = isContinous;
        } else if (isContinous && !isSummery()) {
            JOptionPane.showMessageDialog(dialog, "can't set this activity continous,
continuity is set for summeries only!");
        }

    }

    public void setSubTo(Activity activity) {
        this.subToActivity = activity;
        this.firePropertyChange("subToActivity", activity, this.subToActivity);
        if (this.getPredecessorActivitiesIn().contains(activity)) {
            this.removePredecessor(activity);
        }
    }
```

```
public ArrayList<Activity> getSubActivitiesIn() {
    return subActivitiesIn;
}

public ArrayList<Activity> getSubActivitiesInOfAllSummeries() {
    ArrayList<Activity> getSubActivitiesInOfAllSummeries = new ArrayList<>();
    ArrayList<Activity> allSummeries = getAllSummeries();
    for (Activity act : allSummeries) {
        getSubActivitiesInOfAllSummeries.addAll(act.getSubActivitiesIn());
    }
    return getSubActivitiesInOfAllSummeries;
}
public Activity getSubToActivity() {
    return subToActivity;
}
public Unit getUnitOfActivity() {
    return unit;
}

public boolean isSubActivity() {
    return subToActivity != null;
}
public ArrayList<Activity> getSubActivities() {
    return subActivitiesIn;
}
```

```
/** this class represents predecessor with the methods used to calculate the early start time
 * from the relationship set with a successor
*/
public class Predecessor implements Serializable{
 private static final long serialVersionUID = 21L;
    HRBProject.Activity mainActivity;
    String relationshipType;
public int getFloatBnPredecessor(){
   switch (this.relationshipType) {
           case "FS":
      return this.mainActivity.getLST()-this.predecessorActivity.getLFT();
           case "SS":
           return this.mainActivity.getLST()-this.predecessorActivity.getLST();
            case "SF":
            return this.mainActivity.getLST()-this.predecessorActivity.getLFT();
          case "FF":
                return this.mainActivity.getLFT()-this.predecessorActivity.getLFT();
      default:
              break;
}
return 0;
}
    public Date earlyStartDate(HRBCalendar cal) {
        Date date = null;
        switch (this.relationshipType) {
           case "FS":
```

```
            date = cal.getStartDatesFromPredFinishDate(leadLag,
predecessorActivity.getFinishDate());

            break;

        case "SS":

            date = cal.getStartDatesFromPredStartDate(leadLag,
predecessorActivity.getStartDate());

            break;

        case "SF":

            date = cal.getFinishDatesFromPredStartDate(this.leadLag -
mainActivity.getAdjustedDuration(), predecessorActivity.getStartDate());

            break;

        case "FF":

            date = cal.getStartDatesFromPredFinishDate(this.leadLag -
mainActivity.getAdjustedDuration(), predecessorActivity.getFinishDate());

            break;

        default:

            break;

    }

    return date;

  }


  public HRBProject.Activity getPredecessorActivity() {

    return predecessorActivity;

  }

 public int getLeadLag() {

    return leadLag;

  }

  HRBProject.Activity predecessorActivity;

  int leadLag;
```

```
    public void setRelationshipType(String relationshipType) {
        this.relationshipType = relationshipType;
    }
   public void setPredecessorActivity(HRBProject.Activity predecessorActivity) {
        this.predecessorActivity = predecessorActivity;
    }
    public void setLeadLag(int leadLag) {
        this.leadLag = leadLag;
    }
    public Predecessor(HRBProject.Activity mainActivity, HRBProject.Activity
predecessorActivity, String relationshipType, int leadLag) {
        this.relationshipType = relationshipType;
        this.predecessorActivity = predecessorActivity;
        this.leadLag = leadLag;
        this.mainActivity = mainActivity;
    }


    public int earlyStartTime() {
        int EST = 0;
        switch (this.relationshipType) {
            case "FS":
                EST = predecessorActivity.getEFT() + this.leadLag;
                break;
            case "SS":
                EST = predecessorActivity.getEST() + this.leadLag;
                break;
            case "SF":
```

```
            EST = predecessorActivity.getEST() + this.leadLag -
mainActivity.getAdjustedDuration();

            break;

        case "FF":

            EST = predecessorActivity.getEFT() + this.leadLag -
mainActivity.getAdjustedDuration();

            break;

        default:

            EST = predecessorActivity.getEFT();

            break;

    }

    return EST;

  }

}

/** this class represents successor activities

 * has methods for calculating late start date

*/

public class Successor implements Serializable{

 private static final long serialVersionUID = 24L;

   String relationshipType;

HRBProject.Activity successorActivity;

int leadLag;

HRBProject.Activity mainActivity;

   public Successor( HRBProject.Activity mainActivity,HRBProject.Activity
successorActivity, String relationshipType, int leadLag) {

      this.relationshipType = relationshipType;

      this.successorActivity = successorActivity;

      this.leadLag = leadLag;
```

```
    this.mainActivity = mainActivity;
    }
    public String getRelationshipType() {
        return relationshipType;
    }
    public void setRelationshipType(String relationshipType) {
        this.relationshipType = relationshipType;
    }
    public HRBProject.Activity getSuccessorActivity() {
        return successorActivity;
    }
    public void setSuccessorActivity(HRBProject.Activity successorActivity) {
        this.successorActivity = successorActivity;
    }
    public int getLeadLag() {
        return leadLag;
    }
    public void setLeadLag(int leadLag) {
        this.leadLag = leadLag;
    public Date lateFinishDate(HRBCalendar cal){
   Date date =null;
  switch (this.relationshipType) {
                            case "FS":
                           date =
cal.getFinishDatesFromSucceStartDate(leadLag,successorActivity.getLateStartDate());
                                    break;
                            case "SS":
```

```
                    date = cal.getFinishDatesFromSucceStartDate(leadLag +
mainActivity.getAdjustedDuration(),successorActivity.getLateStartDate());

                break;

                    case "SF":

                      date = cal.getFinishDatesFromSuccFinish(leadLag +
mainActivity.getAdjustedDuration(),successorActivity.getLateFinishDate());

                break;

                    case "FF":

                      date =
cal.getFinishDatesFromSuccFinish(leadLag,successorActivity.getLateFinishDate());

            break;

                    default:

                    break;

                }

       return date;

}

    public int lateFinishTime(){

   int LFT=0;

  switch (this.relationshipType) {

                    case "FS":

                      LFT = successorActivity.getLST() -leadLag;

                     break;

                    case "SS":

                      LFT = successorActivity.getLST() - leadLag +
mainActivity.getAdjustedDuration();

                      break;

                    case "SF":

                      LFT = successorActivity.getLFT() - leadLag +
mainActivity.getAdjustedDuration();
```

```
                        break;
                    case "FF":
                      LFT = successorActivity.getLFT() - leadLag;
                        break;
                    default:
                    break;
                }
        return LFT;
}
}
/** this class represents crews. when crews are created
 * crew name, crew productivity (which will be used for
 * automatic duration calculation), crew cost are also set.
 *stores the resource that composed the crew
 * @author TESFA
 */
public class Crew implements Serializable{
 private static final long serialVersionUID = 5L;
    int Id;
    String crewName="";
String unit="";
double crewProducivity;
double crewCost;
ArrayList<Resources>resourceInTheCrew = new ArrayList<>();
    public String getUnit() {
        return unit;
    }
```

```
    public void setUnit(String unit) {
        this.unit = unit;
    }

    public double getCrewProducivity() {
        return crewProducivity;
    }
    public void setCrewProducivity(double crewProducivity) {
        this.crewProducivity = crewProducivity;
    }

    public double getCrewCost() {
        return crewCost;
    }

    public void setCrewCost(double crewCost) {
        this.crewCost = crewCost;
    }

    public Crew(int id,String crewName,String unit,double crewProducivity, double
crewCost) {
        this.Id=id;
        this.crewProducivity = crewProducivity;
        this.crewCost = crewCost;
        this.crewName = crewName;
        this.unit=unit;
```

```
   }
 public void removeResource(Resource resource){
 for(int i=0;i<resourceInTheCrew.size();i++){
   if(resourceInTheCrew.get(i).resource==resource){
 resourceInTheCrew.remove(i);
 }
 }
 }
 public void addResource(Resource resource,double qty){
boolean resourceAddedbefore =false;
    for(int i=0;i<resourceInTheCrew.size();i++){
   if(resourceInTheCrew.get(i).resource==resource){
 resourceInTheCrew.get(i).NoorQuantity=qty ;
 resourceAddedbefore =true;
 break;
   }
 }
    if(!resourceAddedbefore){
     resourceInTheCrew.add(new Resources(resource,qty));
    }
 }
public void addUsingActivity(Activity a){
 for(int i=0;i<resourceInTheCrew.size();i++){
 resourceInTheCrew.get(i).resource.addUsingActivity(a);
 }
}
```

```
public void removeUsingActivity(Activity a){
 for(int i=0;i<resourceInTheCrew.size();i++){
 resourceInTheCrew.get(i).resource.removeUsingActivity(a);
 }
}
}
```