OBJECT-ORIENTED CONSTRUCTION SCHEDULING

Su-Ling Fan[1] and Ming-Teh Wang[2]

[1] *Ph.D. Student, Department of Civil Engineering, National Taiwan University*
[2] *Associate Professor, Department of Civil Engineering, National Taiwan University*

Abstract: Current network techniques presume that there is only one logical sequence of the activities in a project, therefore planners need to choose one particular logical sequence of the activities even if there is more than one possible sequence. This may request engineers expend many work hours to update the variable relationships between activities to keep a meaningful schedule, which effectively controls the progress of a project. This paper discusses the logic changes during the course of delivering the project. An object-oriented algorithm and computer system, called OERT and OERTSS respectively, have been developed to automatically update the logic changes providing the impact of logic change on the project completion date and the critical path thereby reflecting the progress of the project faster.

Keywords: logic changes, scheduling, object-oriented.

INTRODUCTION

As the complexity of a construction project increases, the need for planning and schedule control becomes more important. Network techniques have become the most widely used ones for project scheduling and control in construction fields over the last few decades.

Current network techniques presume that there is only one logical sequence of the activities. However, in reality there is sometimes more than one possible sequence of these activities. This alternative logic was termed ”soft logic” by Tamimi and Dichmann in 1988[12]. Current models do not differentiate between two types of logic; therefore planners need to choose one particular logical sequence of activities based on some original assumptions. Throughout the duration of a project, it is often necessary to modify logic relationships between activities in order to maintain an established completion date. As the complexity of a construction project increases, the associated need for updating increases, and the procedures become quite complex and time comsuming. This creates problems when the planner is unable to update changes fast enough for the field personnel and schedule control. In this paper the problem is said to be a soft logic problem.

The paper first discusses some important resutls of past research, followed by proposed solutions with the idea of using an object-oriented approach.

PAST RESEARCH

Many techniques have been developed for planning, scheduling, and monitoring construction projects. Bar charts and similar types of graphical displays are excellent tools for displaying schedule analysis. They are easy to prepare and review as compared to some other scheduling techniques. One of the main drawbacks of Bar charts type techniques is their inability of explicitly capturing the relationships between activities.

Critical Path Method (CPM) has been widely used as project planning tools in the construction industry. The explicit representation of activity relationships is one of its main advantages. However CPM does not deal effectively with repetitive cycles, randomness and probabilistic estimates, and network branch decision-making[8, 11].

The Line of Balance (LOB) scheduling technique is aimed at modeling repetitive construction projects and has been used since the 1950s. LOB is based on production curves. The slope of curve relates to the increase in units of production on the y-axis with the increment of time on the x-axis. The LOB technique shows the impact of delays on a specific activity, but not on the completion date of the project[1, 13]. To overcome the limitations as mentioned above, researchers have attempted to integrate LOB and CPM, (e.g., Schoderbek and Digman 1976；Rahbar and Rowing 1992；Russell and Wong； Suhail 1993)[11, 13].

PERT, the Monte Carlo Simulation Approach (MCSA) and probability networks, such as GERT, Q-GERT, R-GERT, P-QERT and VERT provide a more accurate representation of the duration of an activity than a CPM network [3, 7, 9]. Unfortunately, the soft logic problem remains unsolved. Although probabilistic networks provide a new method for capturing network branch decision making and loops in the logic, soft logic cannot be represented accurately within the probabilistic networks.

Tamimi and Diekmann termed the logic existing between the activities that are technically independent as “soft logic” in 1988[12], meaning those “interchangeable activities” could be accomplished at one time. However external factors, such as limited resources, may result in a limitation of total number of activities that can be done at one time. They developed a microcomputer program, SOFTCPM, to deal with the soft logic problem. Each interchangeable set of activities must be distinguished from the regular (non-interchangeable) activities in the SOFTCPM model. Each set of activities can consist of several subsets, and the number of subsets should equal the total number of activities that can be performed simultaneously. When the user specifies in a single data entry transaction which activity he wants to start, the SOFTCPM program performs an activity rearrangement (called topological transformation) by checking the following three requirements. First, the activity belongs to a certain set. Second, all its prerequisites must be completed; and third, there is at least one subset in which none of the activities is currently in progress. SOFTCPM has the advantage over previous techniques in providing a simplified updating procedure with minimum data entry. However, the requirement of making the number of subsets equal the total number of activities being performed simultaneously allows the model to only represent a very limited type of soft logic problems. Furthermore, the model performs topological transformation only on the activities that belong to the set that contains the activity which the user wants to start, but does not rearrange other sets of activities which may produce invalid schedule logic.

In 1992, Amr El-Sersy subdivided soft logic into three subtypes: SOFT, OR, and EXCLUSIVE-OR[13]. SOFT refers to the logic existing between the activities that can be scheduled either simultaneously or reversibly. OR refers to the logic existing between the activities that can be scheduled simultaneously, but not reversibly; while EXCLUSIVE-OR refers to the logic existing between the activities that can be scheduled reversibly, but not simultaneously. In practice, the later two subtypes of soft logic only exist in some very restricted cases with particular design or construction criteria. This paper adopts the definition of soft logic by Tamimi and Diekmann.

Amr El-Sersy also developed a SERSI model which generates possible alternatives to assumed schedule soft logic and constrains the generation of alternatives with three different user objectives: reducing project completion time, increasing network flexibility, and improving resource profile.

The SERSI model makes schedule updating easier and less time-consuming by providing users with the alternatives that satisfy the objectives. In order to prevent the problem of SOFTCPM, not rearranging other sets of activities that may make it produce invalid schedule logic, the SERSI model includes Dependency Fixed (DF) relationships. It forces the SOFT links from its follower activities to have the same states as the SOFT links from their predecessor activities. However, it might produce invalid logic as well. As an illustration shown in Fig. 1, link 300, 400, and 500 are termed as DF links which ensure that link 100 and 200, (101 and 201, and 102 and 202) have same state so that if activity “Excavation Area B” precedes activity “Excavation Area A”, then activity “Set Forms Area B” will precede activity “Set Forms Area A.” Providing two excavation crews and one carpenter crew, the link 100 should be ignored, but link 200 should not be ignored. Thus, in this case, SERSI will produce invalid logic too.

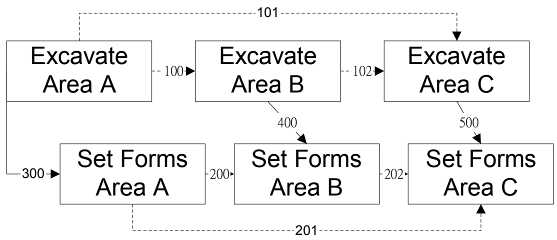


Figure 1. Example of Interdenpdent Soft Links

THE OERT Model

This paper proposes a model called OERT（Object-oriented Evaluation and Review Technique）, which utilizes the object-oriented modeling to deal with the soft logic problem.

Object-oriented modeling is a new way of problem solving for the abstraction problems that exists in the real world. Its fundamental construct is the object, which combines both data structure and behavior in a single entity. This is in contrast to conventional programming where data structure and behavior are only loosely connected. The object model encompasses the principle of abstraction, encapsulation, modularity, hierarchy, typing, concurrency, and persistence. Object-oriented modeling has been recognized for being able to bring benefits such as reusability, stability, reliability, faster design and programming, easier maintenances and etc[4, 6, 10].

The OERT model classifies the schedule logic into two types, fixed and soft. Fixed logic is the logic that exists between the activities which have one possible logic sequence due to physical constraints. Soft logic, as defined by Tamimi and Diekmann, is the logic existing between the activities which are technically independent and can start simultaneously in theory, but due to external factors, such as limited resources, may result in a limitation of total number of activities that can be done at one time.

In the OERT model, each activity is modeled as an object, which will be described by its attributes and behave according to its methods (or functions as termed in procedure programming paradigm). At first, the user defines the sequence of activities with fixed logic as network technique, and the activities with soft logic are grouped into a set. On the other hand, each activity in a set is given a priority number according to the users' preference or random assignment. During the schedule-generating phase, OERT establishes the relationship between the activities in the set automatically with the algorithm of finding out the precedence activity of soft logic with the given total number of activities that can be accomplished at one time. The total Number of Activities that can be accomplished at One Time is referred as NAOT). OERT produces project duration, early and late dates, float time, and critical path automatically as well. During updating phase, with the input of actual dates and the new NAOT (if the external factors have been changed), OERT rearranges the sequence automatically to keep the schedule workable, and gives the early and late dates, float time, and critical path to reflect the impact of logic change on project completion date and on the critical path.

To find out the precedence activity of soft logic named "Soft Precedence Activity (SPA)", an activity calculates the maximum date of early finish of its prerequisites, named as "Fixed Early Start (FES)." It then compares its FES with the FES of each activity in the same set. If NAOT equals one, the activity in the same set that has an equal FES with a higher priority (a smaller priority number) is the SPA, otherwise the activity whose FES is the latest one among the ones earlier than the FES of the activity of the course is the SPA. If the NAOT is more than one, the SPA found by the previous method is put in the list of the SPA candidates, then the activity object will ask the SPA to find out its SPA with the same method. The SPA found is then listed as an SPA candidate. This method will be repeated till the number of activities of the SPA candidates equals the NAOT, and the real SPA is the one where the fixed early finish (FEF) is the smallest within the candidates. Figures 2, 3, 4 and 5 show the complete algorithm of finding the SPA of an activity.

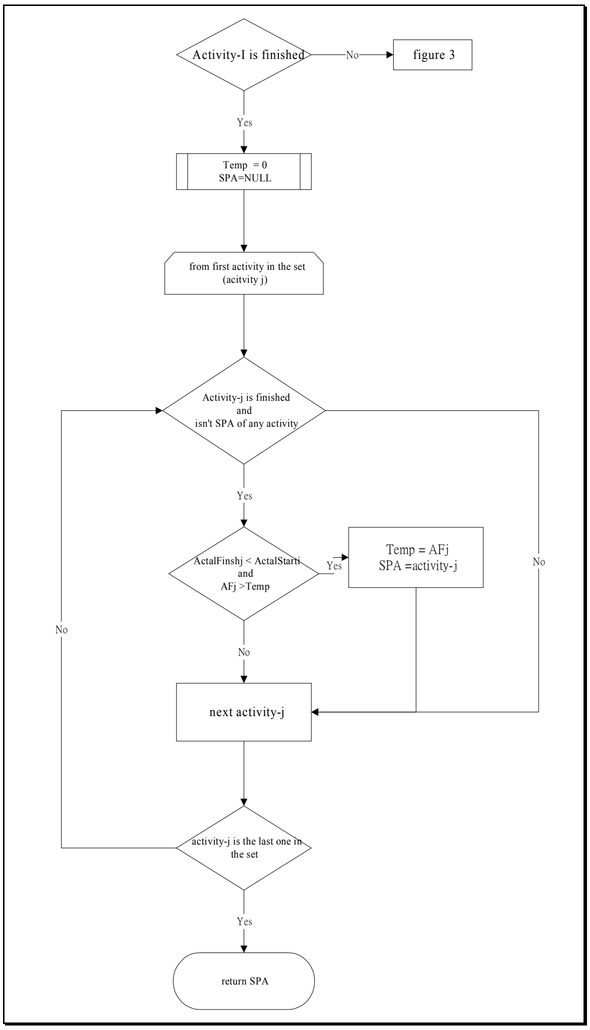


Figure 2. Finding the SPA of activity I (A)

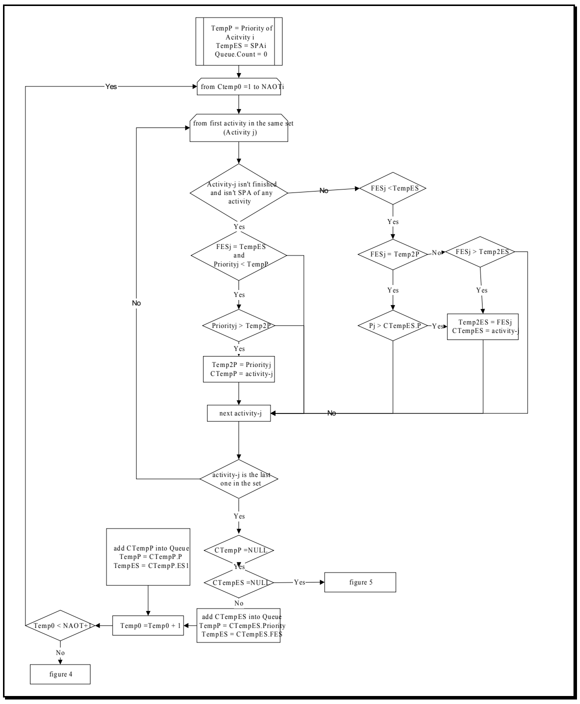


Figure 3. Finding the SPA of activity I (B)

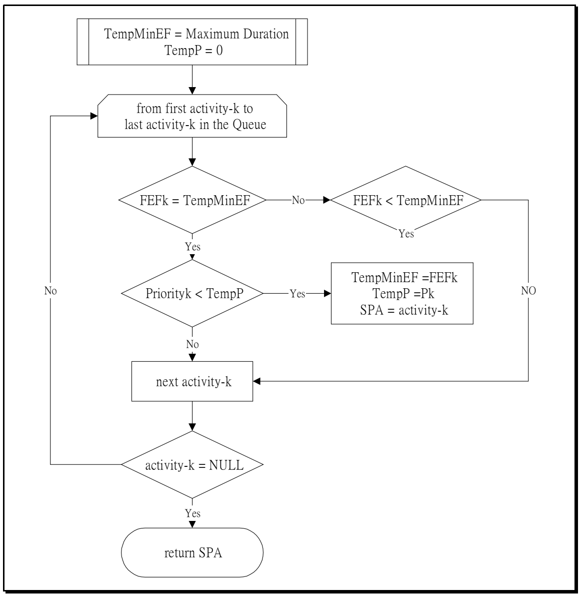


Figure 4. Finding out SPA of activity I (C)

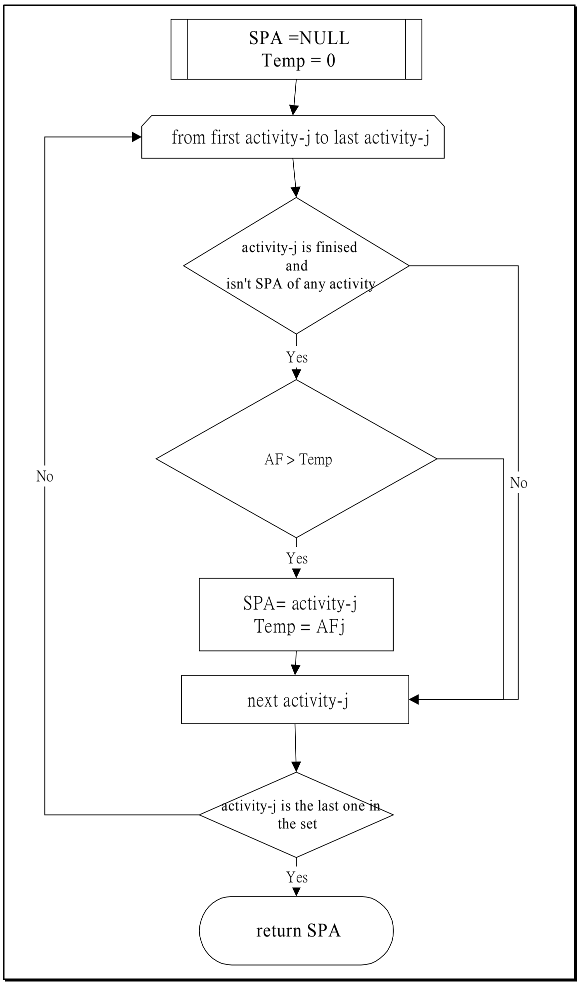


Figure 5. Finding the SPA of activity I (D)

4. COMPUTER IMPLEMENTATION

The OERT model's implementation named as OERTSS (Object-Oriented Evaluation and Review Technique Scheduling System) is facilitated by MS Visurla C++ with its MFC (Microsoft Foundation Class library). Figure 6 shows the main framework of OERTSS.

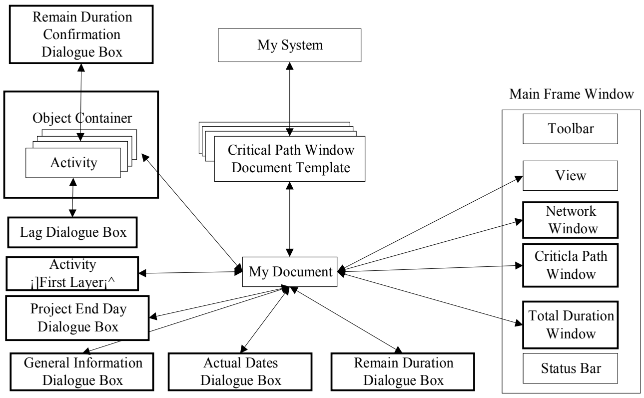


Figure 6. Framework of OERTSS

OERTSS has the following six menus: File, View, Input, Delete, Window, and About. The "File", "View", "Window" and "About" menus are similar to most window system with providing file open, close, save functions, etc. The "Input" menu provides functions for inputting the information of activities such as activity code, description, duration, fixed precedence activities, and activities in the same set, the NAOT, remaining duration, actual dates, etc. After input, OERTSS automatically generates project completion date, early and late dates, float time, and critical path as shown in Figures 7, 8, and 9.

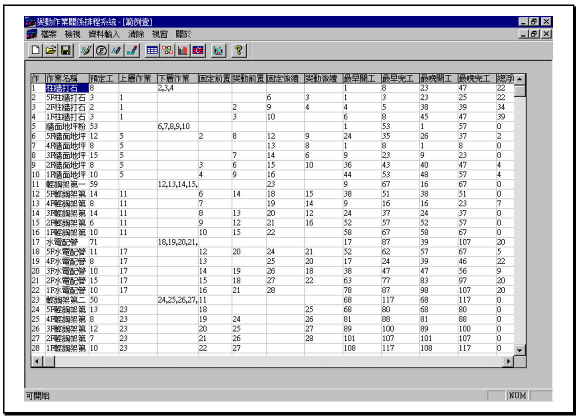


Figure 7. The table window of OERTSS

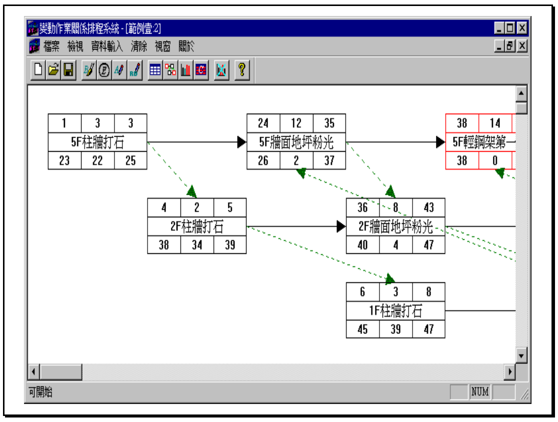


Figure 8. The network window of OERTSS

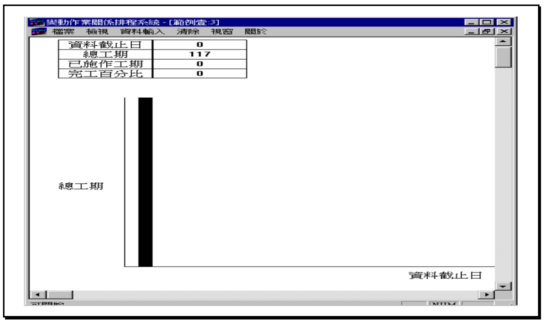


Figure 9. The record of project completion date

EXAMPLE PROJECT

An example is provided to clarify the capability of OERTSS. Assume that a project consisting of five floors is to be constructed. Table 1 shows the project description during the stage of interior finishing. There are 5 primary categories of interior finishing. The NAOT of each category is one. Only 5, 2, and 1 floors require column and wall chipping. After input, OERTSS automatically generates project completion date, early and late dates, float time, and critical path as shown in Figures 7, 8, and 9.

When the external factors have been changed, for illustration, if there is 1 more crew of the wall & floor finishing, and there are 2 subcontractors of piping. 5 and 4 floor is one subcontract, 1 to 3 is another subcontract. The user only needs to change the NAOT of the wall & floor finishing changed from 1 to 2, and 5 and 4 floor piping location arrangement is 1 set, and 1 to 3 piping location arrangement is another set. The logic changes are updated automatically. As figure 10 shows the updated analysis result after the modification, the project duration is reduced to 110 days.

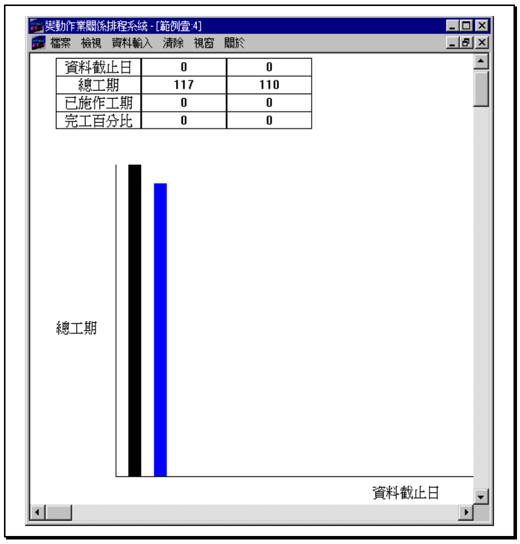
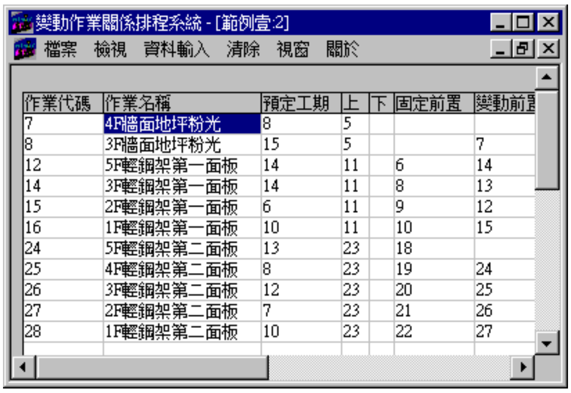


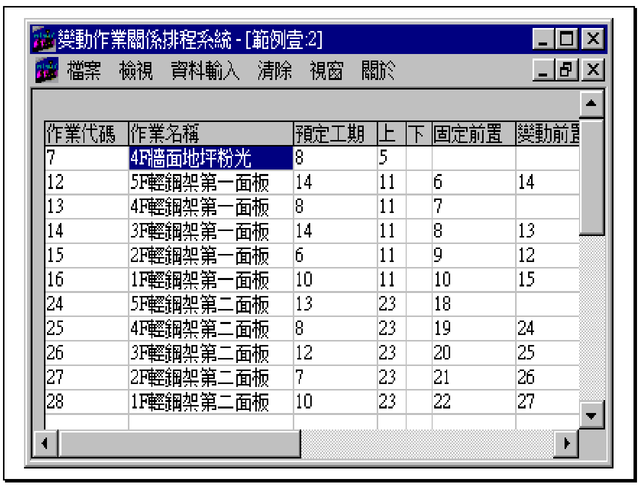
Figure 10. The record of project completion date

Figures 11 and 12 show the critical activities before and after the modification, respectively. As shown in these 2 figures, activity 8, "3 floor wall & floor finish," does not remain on the critical path, and activity 13, "4 floor Drywall first layer," became a critical activity.



作業代碼	作業名稱	預定工期	上	下	固定前置	變動前置
7	4F牆面地坪粉光	8	5			
8	3F牆面地坪粉光	15	5			7
12	5F輕鋼架第一面板	14	11		6	14
14	3F輕鋼架第一面板	14	11		8	13
15	2F輕鋼架第一面板	6	11		9	12
16	1F輕鋼架第一面板	10	11		10	15
24	5F輕鋼架第二面板	13	23		18	
25	4F輕鋼架第二面板	8	23		19	24
26	3F輕鋼架第二面板	12	23		20	25
27	2F輕鋼架第二面板	7	23		21	26
28	1F輕鋼架第二面板	10	23		22	27

Figure 11. Critical activities before the modification



作業代碼	作業名稱	預定工期	上	下	固定前置	變動前置
7	4F牆面地坪粉光	8	5			
12	5F輕鋼架第一面板	14	11		6	14
13	4F輕鋼架第一面板	8	11		7	
14	3F輕鋼架第一面板	14	11		8	13
15	2F輕鋼架第一面板	6	11		9	12
16	1F輕鋼架第一面板	10	11		10	15
24	5F輕鋼架第二面板	13	23		18	
25	4F輕鋼架第二面板	8	23		19	24
26	3F輕鋼架第二面板	12	23		20	25
27	2F輕鋼架第二面板	7	23		21	26
28	1F輕鋼架第二面板	10	23		22	27

Figure 12. Critical activities after the modification

6. CONCLUSIONS

Changes in network logic may cause significant overrun in project duration. These changes occur frequently in the networks that have soft logic problems. In order to keep the network up to date, many man-hours have to be spent when using current scheduling models. This paper proposes an object-oriented model, called OERTSS, to provide simplified procedures with very minimum data entry. Important logic changes will be updated immediately to reflect the impact on a project completion date and on the critical path of the project.

REFERENCES

[1] Al Sarraj, Z.M., *Formal Development of Line-of-Balance*, Journal of Construction Engineering and Management, ASCE, Vol. 116. No.4, pp. 411-424, (1990).

[2] Amr Hassen Ezzat El-Sersy, *An Intelligent Data Model for Scheduling Updating*, Ph.D. Dissertation, University of California, Berkeley, CA, (1993).

[3] Back, W.E., and Bell, L.C., *Monte Carol Simulation as Tool for Process Reengineering*, Journal of Management in Engineering, ASCE, Vol. 11, No.5, (1995)

[4] Booch Grady, *Object Oriented Design with applications*, Benjamin/Cummings, (1991).

[5] Carr, M.I. and Meyer, W.L., *Planning Construction of Repetitive Building units*, Journal of Construction Division., ASCE, Vol. 100, No.3, pp. 403-412, (1974).

[6] Coad Peter and Yourdon Edward, *Object-Oriented Analysis*, Prentice-Hall, (1991).

[7] Crandall, K.C. *Probabilitic Time Scheduling,* Journal of the Construction Division, ASCE, Vol. 102, CO4, pp.415-423, (1995).

[8] Jafari A.K,. *Criticism of CPM for Project Planning Analysis*, Journal of the Construction Division, ASCE, Vol.110, No.2, pp. 196-214, (1984).

[9] Nader N. Chehayeb and Simaan M. AbouRizk, *Simulation-Based Scheduling with Continuous Activity Relationships*, Journal of Construction Engineering and Management, ASCE, Vol. 124. No.2, pp. 107-115, (1998).

[10] Rumbaugh James, et.al., *Object-Oriented Modeling and Design*, Prentice-Hall, (1991) .

[11] Russel, A.D., and Wang, W.C.M., *New Generation of Planning Structures*, Journal of Construction Engineering and Management, ASCE, Vol. 119. No.2, pp. 196-214, (1993)

[12] Samer Tamimi and James Diekmann, s*oft logic in network analysis*, Journal of Computing in Civil Engineering, ASCE, Vol. 2, No.3, pp. 289-330, (1988).

[13] Suhail Saad A., and Neale Richard H., *CPM / LOB : New Methodology to Integrate CPM and Line of Balance*, Journal of Construction Engineering and Management, ASCE, Vol.120, No.3, pp. 667-684, (1994)

Table 1 Project description during Interior finishing

Code	Description	Duration	The upper layer	Fixed Precedence activity	Activity of the same set	NAOT	Priority
1	Column & wall chipping						
2	5 floor column & wall chipping	3	1		3,4	1	1
3	2 floor column & wall chipping	2	1		2,4	1	2
4	1 floor column & wall chipping	3	1		2,3	1	3
5	Wall & floor finish						
6	5 floor wall & floor finish	12	5	2	7,8,9,10	1	1
7	4 floor wall & floor finish	8	5		6,8,9,10	1	2
8	3 floor wall & floor finish	15	5		6,7,9,10	1	3
9	2 floor wall & floor finish	8	5	3	6,7,8,10	1	4
10	1 floor wall & floor finish	10	5	4	6,7,8,9	1	5
11	Drywall first layer						
12	5 floor Drywall first layer	14	11	6	13,14,15,16	1	1
13	4 floor Drywall first layer	8	11	7	12,14,15,16	1	2
14	3 floor Drywall first layer	14	11	8	12,13,15,16	1	3
15	2 floor Drywall first layer	6	11	9	12,13,14,16	1	4
16	1 floor Drywall first layer	10	11	10	12,13,14,15	1	5
17	Piping location arrangement						
18	5 floor piping location arrangement	11	17	12	19,20,21,22	1	1
19	4 floor piping location arrangement	8	17	13	18,20,21,22	1	2
20	3 floor piping location arrangement	10	17	14	18,19,21,22	1	3
21	2 floor piping location arrangement	15	17	15	18,19,20,22	1	4
22	1 floor piping location arrangement	10	17	16	18,19,20,21	1	5
23	Drywall 2nd layer			11			
24	5 floor Drywall 2nd layer	13	23	18	25,26,27,28	1	1
25	4 floor Drywall 2nd layer	8	23	19	24,26,27,28	1	2
26	3 floor Drywall 2nd layer	12	23	20	24,25,27,28	1	3
27	2 floor Drywall 2nd layer	7	23	21	24,25,26,28	1	4
28	1 floor Drywall 2nd layer	10	23	22	24,25,26,27	1	5