Beyond Current Scheduling Programs Using an Extended Relationship Database System

November 6, 2004

ABSTRACT

The core of CPM stands on the bedrock of two separate pillars; activities and relationships. Every day we hear of new additions to the functionality of activities, but what of relationships? Why can we not even produce a listing of relationship lags in sorted order?

Relationships in all scheduling programs are under-represented in both functionality and breath of usability. This paper investigates the current level of sophistication of relationship information and proposes extensive modifications that can be made to enhance the usability of scheduling programs. This system is called an Enhanced Relationship Database System (ERDS.)

With an ERDS, you can add relationship codes and notes, sort by any factor related to the predecessor or successor activity, filter out unnecessary relationships, maintain any number of scenarios, and update your schedule with the revised logic. You can reverse preferential logic and accelerate schedules using a relative degree of 'hard' and 'soft' settings with cost or disruption as a factor. You can even automatically add relationships to represent your resource leveling plan.

BACKGROUND

Physically, CPM schedules are just a collection of data about your project. To facilitate matching the various pieces of different types of data, the data is typically stored in several different files per schedule and related to each other by unique attribute that they all have in common. In this case, all of the files are typically linked to each other using the Activity ID Code. Your activity costs are related to the resources by activity codes. These in turn are related to other activity data such as duration and percent complete by Activity ID Code. Most every piece of data in your schedule is directly related to one activity ID code or another.

Relationships are a little different. They are associated with two Activity ID Codes; the predecessor activity and the successor activity. No one activity 'owns' a relationship. Relationships are shared by two activities. This fact makes relationships stand apart from the other databases. Even if you were to add all of the other database files together to describe an activity, you would still need to have a separate file for relationships.

Primavera Project Planner (P3) from Primavera Systems, Inc. keeps your data in a schedule relationship file with the following items, or “fields.”

- Predecessor Activity ID
- Successor Activity ID
- Relationship Type
- Relationship Lag
- Is this relationship a ‘driving one’? (Yes/No)

The last entry (Is this relationship a ‘driving one’) is not necessary to describe the relationship. This is extra information that Primavera chooses to store to add more insight into the nature of the relationship. In effect, this last entry extends the required data and adds functionality. You could say that in a small way, P3 uses an “Extended Relationship Database.”

Many people believe that much more is needed to manage modern schedules. Many feel that the schedule relationships should be extended in types of data as well as in functionality. This paper will present an entire system that extends the usability and increases insight into CPM schedule relationships.

EXTENDED RELATIONSHIP DATABASE SYSTEM

Relationships are much like activities. By this we mean that if you consider relationship lags, they can take-up time just as an activity would. You can say that relationships with lags are just ‘invisible activities.’ Unfortunately, most scheduling programs do not provide the same extensibility and functionality to relationships as they do to activities.

In this paper, we propose to create an extension to complement the existing scheduling program and data base. It will not take the place of the existing scheduling program, only complement it. We will call this an Extended Relationship Database System (ERDS.) Let’s look into why an ERDS is needed.

Extensibility

Over time, many new data fields have been added to schedule activities that are not absolutely required. Remaining Duration is an example of just such an addition. So are Actual Start and Finish dates. The early versions of CPM schedules did not have these features. Nowadays, we would be at a loss to perform our normal scheduling functions without these added, extended features.

In addition, scheduling programs like P3 [1] also provide additional custom data items where you can define the extra data that you want to track for each activity. This allows you to track such things as planned dates, interim costs, and newly created features such as Longest Path Value without having to wait for Primavera to include it in the program as an upgrade to the software.

Functionality

Just having this extra data is not enough. We need to be able to use it for sorting and filtering the database, grouping activities into sections, being able to print and export this data as needed.

Another issue of functionality is being able to display and manipulate the information directly. You can do this for activities but you cannot do this for relationships. In P3, relationships must always be observed in connection to the predecessor or successor activity. You cannot just display a list of all relationships; you can only show and manipulate the relationships for any given activity.

To report on relationships, you must list all activities and then display the relationships for each. You cannot ask for all relationships that have lags not equal to 0, or for all start-to-start relationships. The lack of these types of reports makes it very difficult for the today's Scheduler to properly review a schedule.

What types of extensions could be added?

Before you can actually use an extended relationship database, you need to create a presentation of it that does not depend upon the predecessor or successor activity. The ERDS needs to be oriented around the concept of independent relationships to be effective. This means that all of your information must be organized by independent relationship records.

A unique relationship consists of the following key fields,

- Predecessor Activity ID
- Successor Activity ID
- Relationship Type

The database may contain duplicate records sharing any of the three fields above, but not all three. P3 allows for each pair of activities to be able to have all four possible relationships at the same time. Note that the 'relationship lag' and 'driving' fields are not required to define unique records.

Note: In P3e and P3e/c [1], it is the internal task id and not the Activity ID Code that is unique. As far as the User can see, a current relationship database for P3e consists of the following required fields,

- Predecessor Project Name
- Predecessor Activity ID
- Successor Project Name
- Successor Activity ID
- Relationship Type

Now that we have a reconfigured system specialized in displaying relationships, what types of extensions could be added? The following is a partial list of what is possible. Many more possibilities will be developed later.

- Manipulate relationships,
- Print and export relationships,
- Maintain inactive relationships,
- Provide a History of Changes,
- Designate Hard versus Soft relationships,
- Estimate relative importance of a relationship,
- Manage Relationship Codes, and
- Maintain Relationship Notes.

What would you be able to do with these extensions?

Manipulate Relationships

Most current software packages are adept at manipulating activities. Why is this functionality missing with relationships? With a ERDS, you should be able to sort and filter on any of the following items,

- predecessor Activity ID,
- predecessor activity codes,
- predecessor resources,
- predecessor dates,
- predecessor Total Float or Free Float,
- successor Activity ID,
- successor activity codes,
- successor resources,
- successor dates,
- successor Total Float or Free Float,
- driving relationships, and
- any of the other extended relationship fields.

Specialized Review

The CPM database is different from other databases in that there are inherent principles that guide good CPM schedules. Such principles could be the basis for specialized reports about relationships. A sort list of such specialized reports would include,

- Review all relationships for proscribed requirements such as “no Start-to-Finish relationships,”
- Lags in daily schedules should be less then 5,
- All activities missing a finish relationship,
- Start-to-Start/Finish-to-Finish pairs.

Deactivate Relationships

Current software programs have only one type of relationships, active ones. There is no place for you to set-up and store alternate plans or scenarios except by making an entire different copy of your schedule.

This method of keeping various copies of the same schedule violates the normalization principle of relational databases. Over time, changes and status will be added to the ‘active’ schedule that will not be reflected in the ‘inactive’ scenario schedule. Very soon, the inactive schedule becomes useless as a tool because it no longer is an exact copy of the original schedule.

What if you could keep your alternate scenario relationships in the same schedule as your current plan? Then with the ‘press of a key’ you could switch one set of relationships for another and instantly change your sequencing of your work. Furthermore, it would be helpful to just be able to keep your deleted relationships on file just in case you change your mind or circumstances change.

Print and Export Relationships.

You should be able to use the sort and filter features to organize and print out the entire database for selected relationships or only just a formatted selection. Also you should be able to export all the relationship data to a spreadsheet for further manipulation, formatting, and printing. You should not need to export and re-format the information to a spreadsheet in order to do this.

How About Extended Functionality?

Once you open you mind to the possibilities available to you with an ERDS, you begin to see possible functionality that is not even related to activity manipulation. Some such functions would include the following topics.

Maintain Inactive Relationships

Why should you be limited to only keeping your current relationships? Why not keep inactive relationships that could be used in various contingencies? How about keeping an intermediate staging plan that would accelerate parts of the project if later authorized or required? How about another set of relationships that would describe the project with Stage 3 before Stage 2 instead of the other way around?

Designate if the Relationship is Hard or Soft

Some relationships represent real-world physical constraints. For example, you normally can't build the roof until the walls are built. Clough, Sears, & Sears [2] call this a physical relationship. Others call this required sequencing a "Hard" Relationship.

Other relationships may describe softer constraints that are not based upon physical constraints but are never the less real and important. In CPM in Construction Management, [3] O'Brien calls this "preferential logic." Relationships that depict crew timing from one activity to the next one requiring that same crew is such an example. A more common term for preferential logic is "Soft" Logic.

Expert schedulers concern themselves with more than just what predecessors are causing the timing of an activity to begin. They wonder if the relationship is physically required (Hard) or if this activity timing is caused by a discretionary logical constraint (Soft) that might just as well be removed for the good of the project.

Being able to isolate and identify Soft from Hard relationships from a schedule would be of large benefit. Of course, you would need to be able to annotate your findings for reference. Later, should a soft relationship become the focus of a delay or acceleration issue, it would be easy to evaluate if the soft relationship was still needed and investigate the results of eliminating that logical restraint.

Estimate the Relative Importance of a Relationship

One technique for accelerating schedules without reducing the work is to selectively delete relationships. If done correctly, this allows activities to begin earlier than they would have. First, we should only consider eliminating only soft relationships, as the hard relationships are physically required. But which of the soft relationships should we eliminate?

The secret is to know just what relationships to delete that will cause the largest results with the least disruption. Relationships could be evaluated based upon the size of the successor's cost, resource, or duration. This would give an indication as to which relationships could be deleted and have the smallest impact to the job.

We call this the 'weight factor' of a relationship. This is separate but linked to the hard/soft designation of a relationship. When a hard or soft designation is added to the weight factor, you have a combination that we call a Relationship Priority. Selective acceleration would then be investigated by eliminating the softest relationships with the greatest weight as a global change in your schedule.

Relationship Codes

Activity Codes are very useful. So are Relationship Codes. You can designate those relationships that interface between phases of construction or soft crew scheduling and quickly bring them together for analysis and modification. In effect, with relationship codes, you can treat a group of relationships in the same manner as you would do for one.

Add a group of inactive relationships that designate alternating phasing and code them the same. Then you can select all current phasing relationships and delete them in one step. Select the other group of alternating phasing relationships by their code and activate the bunch. In two steps, you have completely 're-wired' the entire schedule network!

Provide a History of Changes

P3 does not keep a history of the changes that you make. For the experienced professional, this is a serious oversight. While we are designing our ERDS, we should add in the automatic feature of remembering every addition, deletion, and modification that our system makes to the P3 schedule. We could print out a report or transfer this information over to a word processor for documenting our update.

While we are busy building our automated history of all changes, why not go one step further and add in the feature of being able to reverse any of our updates? Our program could read the history file and apply the reverse of that update to restore the P3 schedule back to the way it was before the update. Of course, we would have to document this reversal in our history file as well.

Relationship Notes

Relationships come and go over the course of a project. They are added to reflect a work plan or constraint. The lags are modified to respond to certain events. Sometimes the relationship is even deleted for acceleration purposes or to modify the work plan.

A history tells you what was done but nothing tells you why it was done. Was this relationship added to reflect a physical or preferential constraint condition? If preferential, then what was the reason, what resource was being managed? Why did you use a lag and what did it represent? Was the lag estimated, based on historical records, or a take-off? What caused you to change it? What was the relationship deleted and why?

These are all crucial pieces of information that must be determined before delay analysis can begin and it is never available. Imagine how better your documentation will be if these facts are documented at the beginning of the project and while the project is on-going instead of at the end.

Automated Resource-Leveled Soft Relationships

When P3 or P3e/c level a schedule using resources, certain activities may be delayed but no relationships are created to reflect this change. All of the Primavera programs mentioned here create a report that indicates which activities were directly delayed due to resource constraints. The P3 version of the report even identifies which resource caused the delay.

An ERDS can read these resource leveling reports and indicate to the viewer which successor Activity IDs were directly delayed by leveling and what resources caused the delay. A column is automatically added to the right of the successor resources column and any delayed successors are identified by delaying resource. Note: Currently P3ec does not report on the delaying resource, so only a “*” is indicated in the delaying column.

A new feature, called Resource Links has be added to analyze the Logic League relationship list and identify the most probable delaying predecessor activity for each delayed successor activity. With P3 this also entails checking to see that the predecessor activity also has a matching delayed resource. Once identified, Resource Links creates the new, soft relationship and codes this in the Hard, Code, and Notes sections to identify this added relationship properly.

The use of Relationship Code here is extremely useful. With each newly added soft relationship coded to the same value, the User can add all of them to the schedule with one function. The User can later remove these soft relationships in mass later when it is time the repeat the update/level process.

IMPLEMENTATION

A proper ERDS must be oriented to display and manipulate relationships independently of the predecessor or successor activities. Due to this lack of the necessary functionality, you cannot just implement an ERDS inside of a standard scheduling program. You need to develop an exterior program built along the necessary lines.

In the same vein, it is difficult or impossible to extend the relationship databases of a standard scheduling program to accommodate all of the new data that you will need. Existing relationship database also cannot deal with inactive relationships. All relationships in a schedule are active. An external relationship database is also needed.

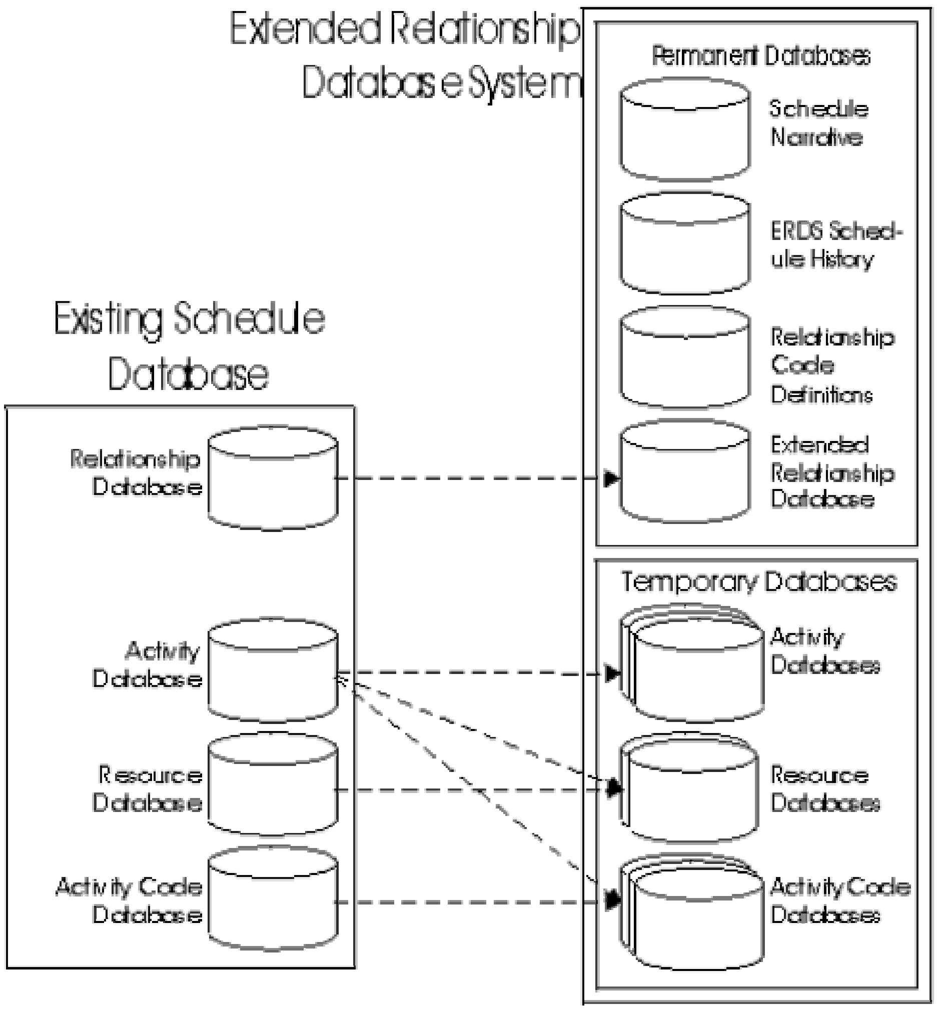
Creating and maintaining an external relationship database also brings its own problems. Chief amongst the issues to be resolved is maintaining synchronization between the two relationship databases. If a relationship is deleted, added, or modified between sessions of the ERDS, then the ERDS must automatically reflect this change. In our implementation of the EDRS, we validate the existing ERDS database against the schedule that it is complementing every time you start-up the ERDS. In a multi-user environment, this becomes even more complicated but we will not address that issue here.

Much of the information displayed in the ERDS is dependent upon the activity. Since the Activity ID is already a key data element of each relationship, this data is already present. To maximize speed and efficiency, instead of saving this redundant information in each extended relationship record, we will create little database in temporary memory related to Activity ID and then reference them when needed. Information directly related to Activity ID to be stored in temporary memory includes,

- Activity Description
- CPM Dates
- Planned (Resource) Dates
- Activity Codes
- Activity Resources
- Total Float

- Free Float

Because this information may change over time, it is read from the existing schedule every time that the EDRS is run. Also performed during start-up is the validation of the existing information stored in ERDS. Any relationships containing deleted activities are automatically marked as deleted. The status of the active and inactive relationships in the EDRS is matched with that in the existing schedule relationship database. Once complete, the ERDS looks like the following diagram.



To implement the above theories, we created an ERDS called, “Logic League [4].” It was necessary to create new software and not just extend the existing scheduling program for two reasons. First, scheduling programs do not have relationship extensibility built-in. Second, the display and manipulation of a relationship database required a new, different format from that provided by existing scheduling programs.

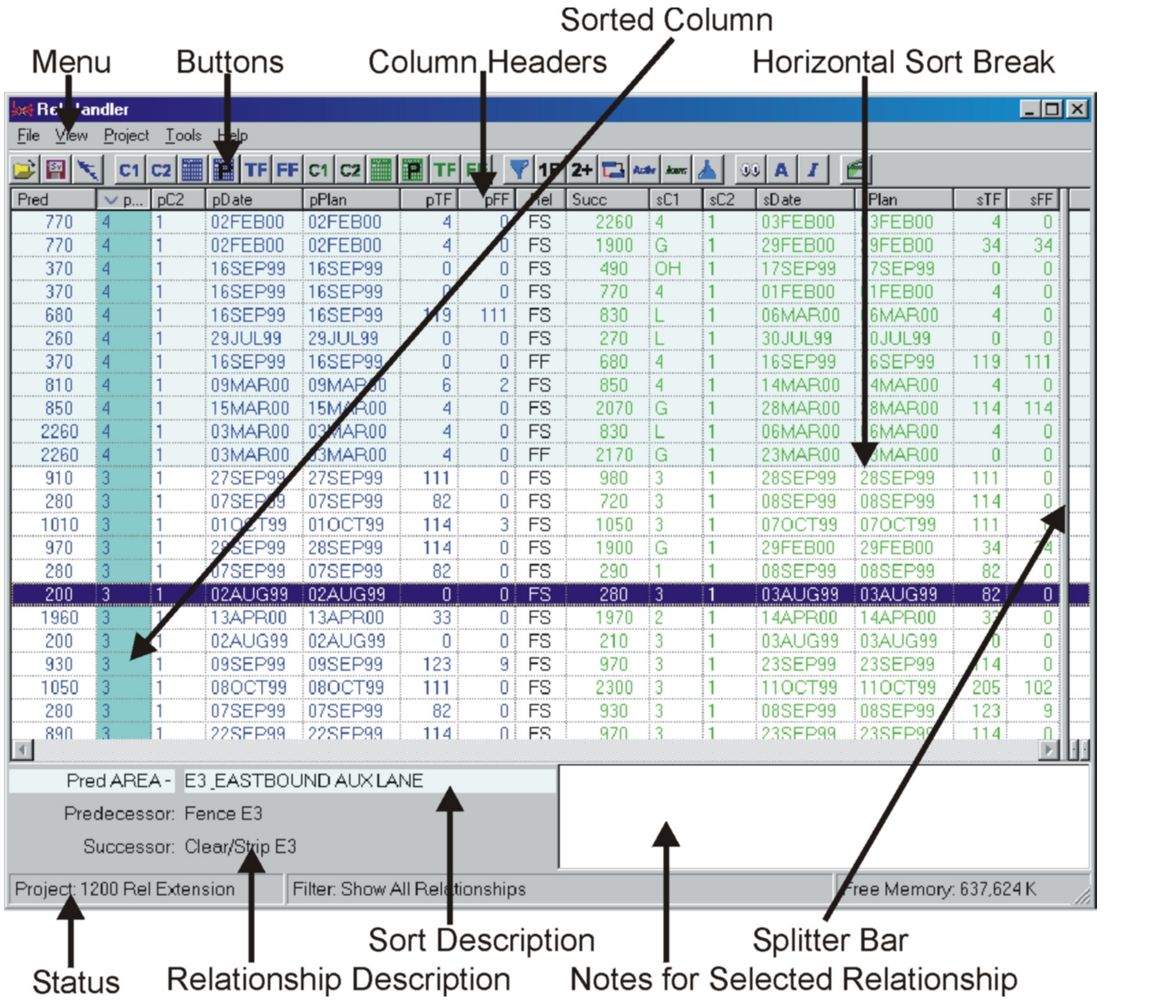
Logic League reads a Primavera Schedule and builds its own logical database that includes everything in Primavera and more. You can view and manipulate your relationships like you do with activities in Primavera. Save the extra information in a separate database and reuse it the next time you use Logic League.

Now you can keep ‘unused’ logic relationships and apply them when you want. Swap out entire logical sections of your schedule for another set with just one function. Special updating functions keep your Logic league database up-to-date with your Primavera schedule.

Besides logical information, other data available to you includes,

- Any two activity codes,
- Early and planned dates,
- Resource information,
- Total and free float.

With the above information, you can readily analyze the nature of each relationship. Below is a screen shot of a typical Logic League display.



Pred	p...	pC2	pDate	pPlan	pTF	pFF	Rel	Succ	sC1	sC2	sDate	Plan	sTF	sFF
770	4	1	02FEB00	02FEB00	4	0	FS	2260	4	1	03FEB00	3FEB00	4	0
770	4	1	02FEB00	02FEB00	4	0	FS	1900	G	1	29FEB00	9FEB00	34	34
370	4	1	16SEP99	16SEP99	0	0	FS	490	OH	1	17SEP99	7SEP99	0	0
370	4	1	16SEP99	16SEP99	0	0	FS	770	4	1	01FEB00	1FEB00	4	0
680	4	1	16SEP99	16SEP99	119	111	FS	830	L	1	06MAR00	6MAR00	4	0
260	4	1	29JUL99	29JUL99	0	0	FS	270	L	1	30JUL99	0JUL99	0	0
370	4	1	16SEP99	16SEP99	0	0	FF	680	4	1	16SEP99	6SEP99	119	111
810	4	1	09MAR00	09MAR00	6	2	FS	850	4	1	14MAR00	4MAR00	4	0
850	4	1	15MAR00	15MAR00	4	0	FS	2070	G	1	28MAR00	8MAR00	114	114
2260	4	1	03MAR00	03MAR00	4	0	FS	830	L	1	06MAR00	6MAR00	4	0
2260	4	1	03MAR00	03MAR00	4	0	FF	2170	G	1	23MAR00	3MAR00	0	0
910	3	1	27SEP99	27SEP99	111	0	FS	980	3	1	28SEP99	28SEP99	111	0
280	3	1	07SEP99	07SEP99	82	0	FS	720	3	1	08SEP99	08SEP99	114	0
1010	3	1	01OCT99	01OCT99	114	3	FS	1050	3	1	07OCT99	07OCT99	111	[illegible]
970	3	1	28SEP99	28SEP99	114	0	FS	1900	G	1	29FEB00	29FEB00	34	34
280	3	1	07SEP99	07SEP99	82	0	FS	290	1	1	08SEP99	08SEP99	82	0
200	3	1	02AUG99	02AUG99	0	0	FS	280	3	1	03AUG99	03AUG99	82	0
1960	3	1	13APR00	13APR00	33	0	FS	1970	2	1	14APR00	14APR00	33	0
200	3	1	02AUG99	02AUG99	0	0	FS	210	3	1	03AUG99	03AUG99	0	0
930	3	1	09SEP99	09SEP99	123	9	FS	970	3	1	23SEP99	23SEP99	114	0
1050	3	1	08OCT99	08OCT99	111	0	FS	2300	3	1	11OCT99	11OCT99	205	102
280	3	1	07SEP99	07SEP99	82	0	FS	930	3	1	08SEP99	08SEP99	123	9
890	3	1	22SEP99	22SEP99	114	0	FS	970	3	1	23SEP99	23SEP99	114	0

Pred AREA - E3_EASTBOUND AUX LANE
Predecessor: Fence E3
Successor: Clear/Strip E3

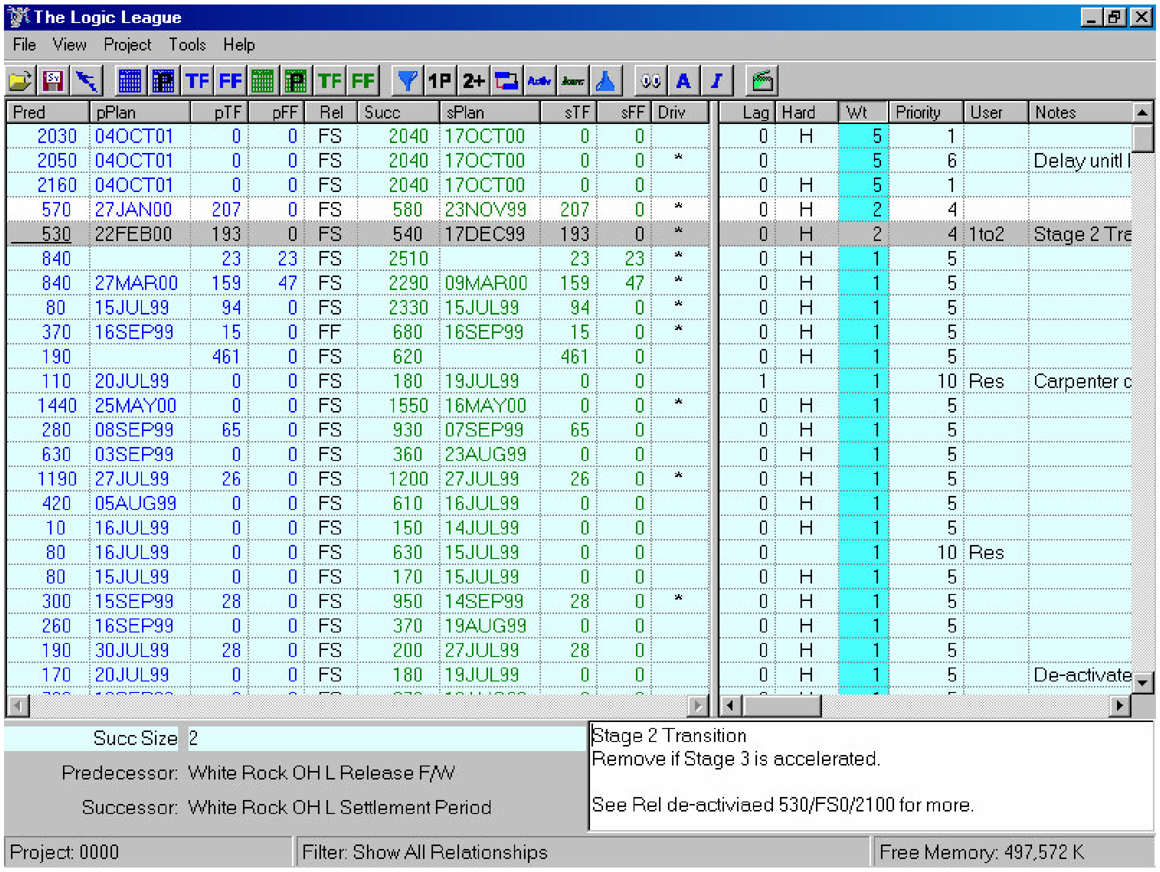
Project: 1200 Rel Extension | Filter: Show All Relationships | Free Memory: 637,624 K

Each horizontal line represents a single logical relationship between two activities. Predecessor information is displayed on the left in blue, followed by the relationship in black and then the successor information in green. The user-supplied information to the right of the splitter bar (not shown) includes hard & soft indicators, relative weights, priority values, relationship code, and notes.

You sort the database by clicking on the column headers. The current sort column is highlighted in dark blue. A very light blue is used to automatically color the entire line depending upon the sort column entry. Whenever the value changes, the line goes from light blue to white and back to light blue when that value changes again. We call this a Horizontal Sort Break.

The information in the grey at the bottom automatically changes when you change the currently highlighted line. This information is an expansion of the necessarily condensed information in the current line. It lists the current predecessor's activity description and the successor's activity description. Above this (in light blue) is a complete expansion of the sort column description and the current value in that line. In the example above, we are sorting on, "Pred Area" and the value, “3” displayed is defined in Primavera as, “E3, Eastbound Aux Lane.”

The figure below shows another screen of Logic League, this time with some of the resource and date information suppressed on the left of the splitter bar. Now the User-definable data is displayed to the right of the movable splitter bar. Everything to the right of the splitter bar may be edited by the User.



Pred	pPlan	pTF	pFF	Rel	Succ	sPlan	sTF	sFF	Driv	Lag	Hard	Wt	Priority	User	Notes
2030	04OCT01	0	0	FS	2040	17OCT00	0	0		0	H	5	1		
2050	04OCT01	0	0	FS	2040	17OCT00	0	0	*	0		5	6		Delay unitl I
2160	04OCT01	0	0	FS	2040	17OCT00	0	0		0	H	5	1		
570	27JAN00	207	0	FS	580	23NOV99	207	0	*	0	H	2	4		
530	22FEB00	193	0	FS	540	17DEC99	193	0	*	0	H	2	4	1to2	Stage 2 Tra
840		23	23	FS	2510		23	23	*	0	H	1	5		
840	27MAR00	159	47	FS	2290	09MAR00	159	47	*	0	H	1	5		
80	15JUL99	94	0	FS	2330	15JUL99	94	0	*	0	H	1	5		
370	16SEP99	15	0	FF	680	16SEP99	15	0	*	0	H	1	5		
190		461	0	FS	620		461	0		0	H	1	5		
110	20JUL99	0	0	FS	180	19JUL99	0	0		1		1	10	Res	Carpenter c
1440	25MAY00	0	0	FS	1550	16MAY00	0	0	*	0	H	1	5		
280	08SEP99	65	0	FS	930	07SEP99	65	0		0	H	1	5		
630	03SEP99	0	0	FS	360	23AUG99	0	0		0	H	1	5		
1190	27JUL99	26	0	FS	1200	27JUL99	26	0	*	0	H	1	5		
420	05AUG99	0	0	FS	610	16JUL99	0	0		0	H	1	5		
10	16JUL99	0	0	FS	150	14JUL99	0	0		0	H	1	5		
80	16JUL99	0	0	FS	630	15JUL99	0	0		0		1	10	Res	
80	15JUL99	0	0	FS	170	15JUL99	0	0		0	H	1	5		
300	15SEP99	28	0	FS	950	14SEP99	28	0	*	0	H	1	5		
260	16SEP99	0	0	FS	370	19AUG99	0	0		0	H	1	5		
190	30JUL99	28	0	FS	200	27JUL99	28	0		0	H	1	5		
170	20JUL99	0	0	FS	180	19JUL99	0	0		0	H	1	5		De-activate

Succ Size 2

Predecessor: White Rock OH L Release F/W

Successor: White Rock OH L Settlement Period

Stage 2 Transition
Remove if Stage 3 is accelerated.

See Rel de-activiaed 530/FS0/2100 for more.

Project: 0000 | Filter: Show All Relationships | Free Memory: 497,572 K

The white box in the lower right corner is the note for the currently selected relationship. You can type and edit this field as you wish; using wrap-around text or multi-line formatting as desired. When you are done, you can print the database or even save it to an Excel spreadsheet.

FUTURE ENHANCEMENTS

Many future enhancements are envisioned for Logic League. Below are just samples of ideas that probably may be implemented.

Remaining Lags

We plan to build in a function to calculate and display Remaining Lag. Currently, the data field known as, “Lag” actually represents “Original Lag.” It does not reflect the amount of lag remaining to be used. This is especially problematic for lags that are currently crossing the data date. These lags are neither at 0 days, nor are that at their original setting. You cannot calculate an in-progress schedule without knowing Remaining Lag. The original CPM did not have Remaining Duration, only (Original) Duration. It is time for CPM to include Remaining Lag.

Out-Of-Sequence Progress

Logic League could provide an option where you can automatically insert a finish-to-finish relationship from the active activity to the activity that has been started out of sequence. In effect, this will say the activity may have been able to start out-of-sequence, but it cannot finish until the preceding logic is complete. In addition, an appropriate note should be added explaining why and when the new relationship was created.

Note: This procedure is also known as “Option 4” as suggested by Fred Plotnick in “CPM in Construction Management” [3], page 154.

As-Built Function

Logic League should have an optional function that will look at all completed relationships and adjust the lag to match the As-Built condition. Perhaps there would be two functions, one for positive and one for negative lags. This function would allow for quick schedule update where the Scheduler wishes to reflect the actual construction sequence and model the early starts of activities. This feature would also be used to re-create an un-statused schedule that reflects the actual logic used.

Switch Activities

Upon occasion, it is necessary to switch the position of two activities. In this case, the simplest situation involves deleting three relationships and adding three relationships. Often in this situation, the lags are often ‘lost.’ Typically, switching two activities involves a lot more than three relationships. It is a simple matter for an ERDS to analyze and generate all necessary relationships for this special function.

CONCLUSION

An Extended Relationship Database System is desperately needed to unlock the potential hidden in relationships. This system is both possible and practical right now if the existing CPM scheduling program is just complemented and not duplicated. We have created and tested just a system called, “Logic League” for P3 and P3ec schedules. The possibilities for advanced functionality are just beginning to be defined but already have tremendous potential and appeal.

Ron Winter Consulting LLC
11100 Gingerwood Way
Rancho Cordova, CA 95670
www.RonWinterConsulting.com

[1] P3 Software by Primavera Systems, Inc., Three Bala Plaza West, Bala Cynwyd, Pennsylvania. Also manufacturers of Teamplay, P3e, and P3ec software (now called ‘IT Project Office,’ ‘Maintenance & Turnaround,’ ‘Engineering & Construction.’)

[2] Construction Project Management, Fourth Edition by Richard H. Clough, Glen A. Sears, and S. Keoki Sears, Copyright 2000 by John Wiley & Sons, Inc.

[3] CPM in Construction Management, Fifth Edition by James J. O’Brien & Frederic L. Plotnick, Copyright 1999 McGraw-Hill. ISBN 0-07-048269-1

[4] Logic League Software. Copyright 2004 by Ron Winter Consulting LLC, www.RonWinterConsulting.com.