Modeling Tough Scheduling Problems with Project Management Software

Alex S. Brown, PMP, Strategic Planning Manager, Mitsui Sumitomo Marine Management (U.S.A.), Inc.

Introduction

Despite year-over-year growth in the use and capabilities of Project Management software, many common scheduling problems remain difficult to represent and manage using these tools. Software alone cannot address fundamentally tough problems, including:

- resource leveling
- representing task dependencies -- hard and soft
- managing difficult-to-predict or quickly-changing work assignments
- making schedules easy to maintain during project execution

Solving these problems require analysis of a project's objectives, constraints, risks, and opportunities. They require a thinking human being to guide scheduling software.

Novice users of scheduling software have a difficult time with these types of problems. Their use of the software is driven by an urgent need to represent their schedule in a report, by any means possible. Advanced users of scheduling software, though, use the software to develop a model of their schedule. Eventually they will produce the same reports as a novice user, but their approach to the software is different.

Although techniques will vary with each scheduling software tool, some basic approaches are fundamental to building an accurate schedule. By running key scenarios through his or her software tool, a manager can help make progress from a novice to an expert user. Even expert users profit from these exercises, as a way to challenge old habits and try new techniques.

Novice Use of Scheduling Software

Novice users of scheduling software have a primary goal of communication. They have an urgent need to produce reports:

- Gantt charts
- Task lists
- Team member reports
- Executive reports

Their primary goal is to get the correct dates, duration, work, resources, and other project data into the tool and onto reports. Modern project management software allows the project manager to type desired values into a spreadsheet-like view to achieve this result quickly. The software often makes choices about dependencies, resource usage, and other factors, so reports appear on-screen quickly.

Communication is the critical factor in successful project management. Many managers never go beyond this "novice" phase; many never see the need. Even after years of experience as a project manager, many people's scheduling skills remain at this level. Symptoms include (Uyttewaal 2001, 24-26):

- Failure to maintain an up-to-date schedule during project execution; creating a schedule at the project start and never updating it afterwards
- "Delicate schedules" whose end-dates change dramatically each time actual work is applied to them
- Large amounts of time and frustration spent for each schedule change (new tasks, changed assignments, etc.)

Expert Use of Scheduling Software

Project managers with more expertise use scheduling software very differently. They create a model of the project in their software tool. They follow consistent steps to build up a schedule. They understand and override the default settings of their chosen software tool, manually specifying task dependencies and resource usage. Often they ignore predicted end-dates for tasks while entering key data. They wait until their schedule is complete and resource-leveled, then optimize the schedule to reach a desired end-date.

Expert schedulers run "what-if" scenarios on their schedules. They modify start and end dates for critical tasks; they ensure that the remaining project work moves in a realistic manner. They understand what their software will do if key tasks are actually done out-of-order. They regularly update their schedule based upon actual work completed, predicting a new end-date for the overall effort when necessary.

Every project manager comes up with his or her own habits when working with scheduling software. Scheduling software and experience drive these habits. Managers must develop a new mental map of the software. Experienced users view their software as a database manager, not a spreadsheet. They see their project files as a database of related facts, with many views to see the data from different angles. (Stover 2003, 7-8) They consult many views, including the network diagram and resource usage reports, even if they are never published.

Establishing Key Terminology

To allow users of different scheduling software tools to talk to each other, it is necessary to go back to basic definitions. The PMBOK provides definitions for standard terms. Different software packages use different terminology, but in some way they all represent network diagrams, task lists, Gantt charts, finish-to-start dependencies, and so on. This paper uses definitions from PMBOK.

To discuss common scheduling problems, this paper defines two key classes of projects:

- Parallel opportunities
- Serial dependencies

Real projects often contain a mix of these situations. Some projects follow one pattern for some phases and the other pattern for other phases. Scheduling these two types of projects, though, demand different approaches.

The network diagram for a parallel-opportunities project contains many tasks that could be done at the same time (see Exhibit 1). Software enhancement projects often follow this pattern; each software component could be designed or constructed independently. At some point all changes need to be integrated, but each change can be worked independently for a large part of the project. The number of available resources typically drives the end-date for these projects.

The network diagram for a serial-dependency project requires that tasks be done in a specific order (see Exhibit 2). Construction projects often follow this pattern, where walls cannot be erected until the foundation is poured and set, or where walls cannot be completed until wiring, pipe, and key inspections are complete. The total duration of the critical path typically drives the end-date for these projects.

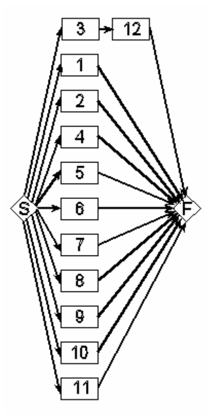

Exhibit 1
Parallel
Opportunities

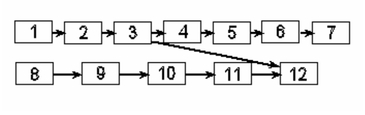

Exhibit 2 Serial Dependences

Workflow for Modeling a Schedule

By breaking schedule development into well-defined, discrete processes, the PMBOK suggests a workflow for building a schedule. In real life, building a schedule is often an iterative process, starting with a small, basic schedule, and building on it by adding phases or groups of tasks. Whether performed as a single pass or multiple iterations, experienced project managers perform these steps:

1. Build a Work Breakdown Structure (WBS)
2. Decompose the WBS into concrete activities
3. Define the sequence of activities (dependencies, predecessors, and successors)
4. Estimate task duration
5. Estimate task cost
6. Assign resources
7. Adjust cost and duration estimates based upon assigned resources
8. Perform initial resource leveling

9. Optimize the schedule
The final step, "optimize", may include changes to all of the information entered earlier. Activities might be grouped together, activities might be decomposed, resource assignments may change, and estimates of cost or duration may grow or shrink.

During this "optimization" step, the manager probes the schedule, assessing the quality of the model of the schedule. Changing assignments and estimates should adjust the overall end-date with minimal effort. The better the model, the easier it is to optimize. The more fragile the schedule, the more unpredictable changes occur with each adjustment.

Often modern schedules are tightly constrained. They are designed to produce the maximum output, just-in-time, for a minimum budget. Optimizing tightly constrained schedules is difficult. Allow extra time for creating such schedules.

In the rush to produce a schedule, it is tempting to bypass critical steps. For instance, if a task "must" begin on a certain date, it is tempting to ignore activity sequencing for any predecessor to that task and to simply set a start date for it. If the manager shortcuts the process, the software cannot warn the manager of scheduling conflicts.

Some Tough Scheduling Problems

Resource Leveling

One of the enormous benefits of scheduling software is that it provides the manager with multiple, synchronized views of the project. Gantt charts are effective for communicating dates and network diagrams are effective for communicating dependencies. In the 1960s and 1970s, managers who used critical-path techniques had to manually draw and update these diagrams by hand. Any changes to the project scope meant updates to multiple diagrams, charts and lists. Today's software automates this essential function. Users may edit data in one view, and the software will display up-to-date versions of all other views at the click of a button.

Theoretically, these advances should make it easy to keep a resource-leveled schedule. Indeed, modern software provides views that show the utilization of any resource in just about any unit, cumulative by date or for a particular period. With the entry of vacation calendars, holidays, resource availability over time, and other constraints, software can often show overallocation or underallocation of available resources automatically.

The key to accurate resource leveling is accurate, complete data. So many variables come together in this critical planning task:

- Resource start dates and availability
- Work estimates
- Duration estimates
- Activity sequence relationships
- Vacation schedules
- Holiday schedules
- Planning factors for unscheduled but known events like sick days
- Planning for schedule risk events

Resource leveling is the integration point for all these project elements.

Complexity increases with the use of scheduling software. It is possible to force a schedule to be resource-leveled strictly through the use of dependency relationships in most tools. Some tools offer automated systems to adjust resource allocations. Project managers should explore their tools thoroughly to understand the best techniques for their projects. Key questions for the project manager include:

- Is the method repeatable for future projects and for future reporting periods with this project?
- What if a key resource starts earlier or later than expected? How much data must be adjusted or re-entered?
- What if a single task takes more or less time than expected? How will the software respond?
- When there are two tasks that COULD start on the same date (but resource constraints will not allow them to), how does the software decide which one SHOULD start first? Does the software make decisions automatically? Can the project manager override any automated decisions easily?

Scheduling software is often designed for projects with serial dependencies. By adding "soft" dependencies to a schedule, it is possible to turn a project with many parallel opportunities into one with mostly serial dependencies. If a manager uses this technique,

- Where will the hard dependencies be documented?
- Will the software still accurately calculate key project measures, including critical path?
- When resources do work out-of-order, does the manager need to update the schedule? How will the software respond if the actuals do not match the planned activity sequence?

The better the manager understands his or her software tool, the less time is required for schedule maintenance.

Representing Task Dependencies -- Hard and Soft

Task sequencing is often challenging with scheduling software. Some software will add sequencing rules to a schedule, making all tasks sequential automatically; many expert schedulers disable these features.

PMBOK distinguishes between dependencies that MUST be enforced (hard or mandatory) and dependencies that SHOULD ideally be enforced (soft or discretionary) (PMBOK 2000, 68-69). Many software packages do not make that distinction, enforcing all dependencies strictly. Some resource-leveling strategies require the use of many soft dependencies, compounding the problem.

Managers have a few options to work around these common problems:

- Use the software to enforce ONLY hard dependencies, and attempt to meet soft dependencies as part of resource-leveling steps
- Create an initial network diagram with only hard dependencies and maintain it outside the main schedule (especially useful when the project's network diagram changes infrequently)
- Ignore the software's network diagram and create one by hand
- Keep a spreadsheet or diagram of hard dependencies and soft dependencies – compare the schedule in the software to these lists regularly and update the software as needed
- Create custom views or use custom fields in the software to track hard and soft dependencies

Task numbering is often an obstacle in creating and synchronizing lists of tasks and dependencies. Most software numbers all tasks sequentially, and adjusts the number of every task as tasks are added or deleted. Often there is a hidden field that does not change that uniquely identifies each task. Use this number to effectively cross-reference between a spreadsheet or diagram and the data in the scheduling software. Alternatively, create task names that are unique within the project, and use these names as an effective cross-reference.

Managing and Maintaining Changing Schedules

"Everything flows on and on like this river, without pause, day and night." –Confucius, standing by a river (Wilhelm 1987, lv)

The one unchanging principle that every project manager can count on is change itself. No matter how accurate the estimates, no matter how perfect the schedule, before the project is done, the schedule will change. Some tasks will start or end early, some will start or end late, and the project scope will expand and contract. Managing this change is the never-ending role of the project manager.

Managing schedule change is a requirement for many tough schedules. Fast-changing resource assignments, frequent changes to project scope, changes to budget, and changes to desired end-dates can make a project much more difficult to manage.

When creating a schedule for a brand new project, a manager should ask:

- What is likely to change?
- How much will it change?
- Will project changes come all at once, or a little with every reporting period?

The more the manager can anticipate change, the better he or she can plan for it. When creating the initial schedule, the manager can perform some "what-if" scenarios, seeing the effect of certain types of changes will be. Even if the exact changes cannot be predicted, the manager can develop scheduling techniques to address common sources of change:

- New tasks
- Old tasks no longer needed
- Resources are unavailable or new resources become available
- Resource start dates changing
- Changes to duration, work, and cost estimates

An expert scheduler can draw upon experience to address these situations. A novice needs to develop these techniques or learn from an expert. No matter what the experience level, though, any manager can benefit from identifying likely sources of changes and planning for them.

Modern software has opened new possible ways to model change, making uncertainty an explicit part of the project schedule. Software has been able to generate likely ranges of estimates based upon PERT estimates of likely, optimistic, and pessimistic outcomes for years. Monte Carlo simulation can be used to show the range of possible outcomes, and their relative probability. Many project managers do not need or use these tools, but for highly uncertain projects, they can be an essential feature of scheduling software.

Appendix: Scenarios

Sample scenarios help to illustrate, teach, and discuss abstract scheduling concepts. This appendix contains concrete examples to help managers learn and apply these principles. By experimenting on a small schedule, a manager can learn the pros and cons of different techniques.

In these examples, “duration” is the minimum possible duration for a task in days. “Work” is the number of work hours estimated to complete the task. Milestones have the abbreviation “(MS)” in their task name. Milestones have zero work and zero duration.

Basic Scenario

A manager has six tasks, each of which can be done simultaneously; there are no hard dependencies. The manager has three people available to perform the work. The task list appears as follows:

Task	Duration (days)	Work (hours)	Resource	Predecessor
1. Project Start (MS)	0	0		
2. Construct #1	5	20	A	1
3. Construct #2	5	20	A	1
4. Construct #3	2	16	B	1
5. Construct #4	2	16	B	1
6. Construct #5	5	16	C	1
7. Construct #6	2	16	C	1
8. Project End (MS)	0	0		2, 3, 4, 5, 6, 7

To start, assume all three resources are available 40 hours per week, and a week is five workdays.

Resource Leveling

Most software will show dates that are too optimistic after entering the raw data above. The use of project resources (A, B, and C) must be leveled, so they are used no more than 40 hours per week. Scheduling options include:

- Keep the dependency relationships as defined above
  - Manually “fix” start dates for each task to make the resources level over time
  - Allow all tasks to start on day 1, but change the allocation of each resource to create a level use of resources – in other words, adjust the time-per-day that each resource spends on each task until the end dates all match
  - Adjust the time-phased work assignments for each task until the resource usage for each person is level (procedure will vary with each software package)
- Introduce “soft” dependencies
  - Keep the work for “A” as is, since the two five-day tasks add up to 40 hours already
  - Make “Construct #3” a predecessor for “Construct #4”, so that “B” has a level set of work – 32 hours over 4 days
  - Review the work for “C” and find a way to reschedule it, possibilities include

    - Split "Construct #5" into two tasks – one with 8 hours work for one day, and a second with 8 hours of work for one day; put a mandatory delay of 4 days between them to keep the overall duration of five days, and schedule "Construct #6" between these two, new, one-day tasks
    - Spread the work for "Construct #6" over four or five days, so it becomes a level set of work that can be done simultaneously with "Construct #5"

Enter this scenario into your scheduling software a few different ways:

- Which method gets an accurate, level schedule the most quickly?
- Which method lets the manager set the start and end dates most flexibly?
- Which way do you typically use?
- Have you discovered any new techniques that might work better than your usual approach?

Representing Hard and Soft Dependencies

In the scenario above, use dependencies to force the schedule to be resource-leveled. The task list becomes:

Task	Duration (days)	Work (hours)	Resource	Predecessor
1. Project Start (MS)	0	0		
2. Construct #1	5	20	A	1
3. Construct #2	5	20	A	1
4. Construct #3	2	16	B	1
5. Construct #4	2	16	B	4
6. Construct #5a	1	8	C	1
7. Construct #6	2	16	C	6
8. Construct #5b	1	8	C	6+4days, 7
9. Project End (MS)	0	0		2, 3, 4, 5, 7,8

In your scheduling software, consider the following questions:

- The hard dependencies for the schedule are no longer documented in the schedule clearly. Where will you document them now?
- According to your scheduling software, what is the critical path? Do you agree with the software?
- Assume that "Construct #4" becomes a four-day task with 32 hours of work. The critical path will then run through "Construct #3" and "Construct #4". Many people would say that these soft dependencies could be broken, so the critical path should not change. Do you agree?

Managing Quickly-Changing Work Assignments

In the exercises above, you created several versions of the same schedule, each using different techniques and features of your scheduling software. Apply the following changes to each schedule:

- Resource "A" is only available for 20-hours per week, and you cannot change which tasks are assigned to "A". Forecast the new end-date.
- You now discover that you can substitute any resource for any other resource, but it will add an extra, eight hours to the work for any task that you switch. Optimize the schedule, with "A" at a 20-hour workweek.
- Assume that "B" volunteers to work 60 hours per week to get this project done. Optimize the schedule, following the two rules above.

After applying the changes to several schedules, answer some questions:

- Which schedule was easiest to change? Was it the easiest one to build initially or one of the others?
- While applying these changes, did your scheduling software unexpectedly shift dates? When? Why?

Making Schedules Easy-To-Maintain During Project Execution

Week one of the project is now complete. Team status reports show:

Task	Resource	Status	Actual Work	Remaining Duration (d)	Remaining Work (h)
1. Project Start (MS)					
2. Construct #1	A	Completed	16	0	0
3. Construct #2	A	Started	8	2	16
4. Construct #3	B	Not Started	0	2	16
5. Construct #4	B	Started	8	1	8
6. Construct #5	C	Started	8	3	8
7. Construct #6	C	Not Started	0	2	16
8. Project End (MS)					

Apply these actuals to different versions of the schedules, as created in the earlier exercises. Review the new end-dates and the resource utilization for each resource for the remaining work. Answer some questions:

- Was this a new experience, applying actuals and creating an updated forecast? If so, what new techniques did you learn?
- After applying actuals to different versions of the same schedule, how many different end-dates resulted? Which ones were the most accurate? Which ones were the easiest to change and to make realistic?
- While applying changes, did your scheduling software unexpectedly shift some dates around? When? Why?
- Resource “B” started “Construct #4” before “Construct #3”. How did your scheduling software respond to that?
- How does your software reschedule work that is started but not completed? Not started?

Summary

After running these exercises, review

- Which techniques did you usually use before? What are their strengths and weaknesses?
- Which of these tasks do you usually perform weekly? Monthly? Daily?
- Which scheduling techniques make schedule entry easiest? Which make maintenance easiest? Which strike a balance?
- Which techniques do you now plan to use in the future?
- What did you learn about your scheduling software that you did not understand before?

Future Development

These examples illustrate a schedule with parallel opportunities. Develop a sample schedule with serial dependencies, and examine the behavior of scheduling software when:

- Tasks are completed in the expected order
- Tasks are completed out-of-order
- Tasks with many dependencies end late
- Tasks with many dependencies end early
- Tasks use dependency relationships other than “finish-to-start”, including “start-to-start”, “finish-to-finish”, and “start-to-finish” (not all software supports all these relationships, and some software has flaws in their support for unusual relationships)

References

Project Management Institute. (2000) *A guide to the project management body of knowledge (PMBOK®)* (2000 ed.). Newtown Square, PA: Project Management Institute.

Stover, Teresa (2003). *Microsoft Project 2002 Inside Out.* Redmond, WA: Microsoft Press.

Uyttewaal, Eric (2001). *Dynamic Scheduling with Microsoft Project 2000.* Ottawa, ON: International Institute for Learning, Inc.

Wilhelm, Hellmut (1987). *The I Ching or Book of Changes: The Richard Wilhelm Translation* (3rd ed.) (C.F. Baynes, Trans. To English). Princeton, NJ: Princeton Univ. Press (Original translation from Chinese to German by Richard Wilhelm in 1923).