Abstract

There are a great number of different aspects to a schedule. Many organizations and experts have their opinions on what constitutes a well-built schedule. Regardless of these various outlooks, however, the question that should be the primary focus is: “Is the schedule executable?”

The basic construction of a schedule revolves around a proper approach to its development. The approach should be structured in a way that is simple to understand and yet provides enough detail for the experienced to glean the requirements used. This report discusses tried and true approaches that have been developed through years of analyzing and refining processes for schedule development. This approach provides the structure to implement proven best practices to building an executable schedule. We delve into this step-by-step approach by:

Developing the work breakdown structure (WBS)

Defining work packages

Defining activities

Defining logic

Defining resources and work

Defining timeframe

Analyzing the schedule

Developing the Work Breakdown Structure (WBS)

The most crucial part of any schedule is the proper development of the work breakdown structures (WBS). Without the WBS, the work to be accomplished will flounder and the schedule will be unorganized and for the most part not executable. Development of a proper WBS is the backbone for a proper schedule. There are several references available to use as guidelines for kick-starting the development of the WBS (e.g., PMI-WBS, MIL-HDBK-881). Remember that the definition of a project is a “unique” endeavor; therefore, while one WBS could be used as a starting point, I can assure you that the entire WBS cannot be used to fully describe your project.

There are actually only two basic types of WBSs used within industry today: the Functional and the Deliverable (product) types. Of course, there are hybrids of these, but they still rely heavily on either one or the other of these two basic WBS types.

Functional WBS

The functional WBS is used in organizations that are heavily involved in the functional aspect of business. Breakdowns such as System Engineering, Software Engineering, and Hardware Engineering are seen often. While these groups do have a large role to play in some projects, normally they do not have a deliverable that they can create or perform without some interfacing with another group (e.g., some requirements performed by System Engineering are necessary before design work can begin within other engineering groups). It can be difficult to ensure proper logic within a schedule when using the functional approach; however, it can be accomplished. The end result normally shows a critical path (explained later in this paper) that travels all over the schedule and is hard to trace from beginning to end. Exhibit 1 provides an illustration of a functional WBS.

Functional WBS

Exhibit 1: Functional WBS.

Deliverable WBS

The deliverable-oriented WBS provides for the most control over the project and the best basis for schedule development. However, this structure takes some coaching throughout most organizations, as they are accustomed to dividing up the work and “throwing it over the wall” among the groups. A deliverable-oriented WBS allows for a more cohesive team working on a single product or deliverable throughout. The main advantage (although cohesive teaming is a strong advantage as well) is that working in a deliverable-oriented WBS allows for a seamless traceability throughout the whole project, from the requirements in the contract vehicle to the lowest level item being worked. This ease in traceability ensures that the customer is getting everything that he or she expects (albeit not necessarily everything they thought they were getting—a topic for another paper.) Compare Exhibit 1 with Exhibit 2, which illustrates a deliverable WBS.

Deliverable WBS

Exhibit 2: Deliverable WBS.

Defining Work Packages

The work package is the lowest level of detail within the WBS. There are normally several work packages required in order to fulfill a single WBS element. Work packages must be structured so that all work can be allocated to complete the WBS element. Work packages should be couched in the “verb-noun” structure illustrating what action is accomplished on which item (e.g., Develop Requirements Traceability Matrix, or Program Sort Routine.) The work package description is key to understanding the work to be accomplished within the quadruple constraints of Scope, Quality, Time, and Cost. Normally earned value data is captured at this level of detail to ensure that the proper health of the project is being tracked.

Defining Activities

Activities are a further breakdown of the work packages. Don't get hung up on which term we use here to describe these activities: some software packages consider them to be “tasks,” and others consider them “steps.” What we are discussing here are the actual actions taken by an individual or group that can be measured to determine their current state subjectively and not objectively. It is imperative that the work defined here is something that can be measured. Let's run through a quick example of developing a document as shown in Exhibit 3:

Activity definition example

Exhibit 3: Activity definition example.

Note that while the “Wrong” column may seem measureable, when can you take credit for the different points? How do you know that you are 25% or 50% done with something? The “Right” column provides for specific measurements when accomplishments can be measured.

Defining Logic

Logic is the biggest driver in a properly constructed schedule. Planning and care must be taken to ensure that the logic is accurate and that false or soft logic is not used. Hard logic is the only valid form of logic to use within a schedule; any false or soft logic defeats the effectiveness of the tools used and will need to be modified during schedule execution. There are four standard types of logic used, with Finish-to-Start being the most prevalent and easiest to understand and use.

Types

There are many aspects to using the right logic type within a schedule. I have seen several organizations that do not understand the full intricacies of scheduling that put policies in place handcuffing the experienced scheduler (normally as a reaction to being bitten by poor scheduling.) These policies restrict the type of logic in some cases to Finish-to-Start only, while in more liberal cases they allow all except for Start-to-Finish. I almost agree with the latter policy, as the Start-to-Finish constraint can be tricky if you are not familiar with its use and the application (scheduling tool) being used. Each tool used (e.g., Primavera P3e, Deltek OpenPlan, MS Project) handles the logic with a little different algorithm, providing for varied results. The following examples are using only hard logic.

Every activity in the schedule will have at least one predecessor and one successor. The only exception to this is the very first activity (contract award, project start, or whatever you call it), which will not have a predecessor, and the very last activity (contract complete, project end …), which will not have a successor. To stop tying up the ends of the activities (both predecessor and successor) leaves those activities hanging as an isolated activity that will never show up on the critical path until the project's remaining duration is equal or less than that activity's duration.

When developing the logic, NO logic should exist on Summary or Hammock lines. This will override or confuse the logic on the activities within.

False Logic - logic used to arbitrarily modify the sequence of events in a schedule to bring about preconceived results in an activity's start and/or finish dates. (For example, Doc 1 must finish before Doc 2 can start when the same resources are working both documents but the documents do not relate in any other way to each other.)

Soft Logic – logic used to shift references of schedule start and/or stop dates based on perceived requirements within the activities. (For example, Rough Plumbing must finish before Rough Electrical can start. Although this is normal practice in construction, this is not hard logic.)

Hard Logic – logic used which provides valid links between activities. (For example, the foundation must be cured before the walls can be erected.)

Finish-to-Start

FS logic

Exhibit 4: FS logic.

This is by far the most popular and easily understood of the logic types in use. Most tools handle this logic type the same and provide for a smooth transition between tools when needed. Those unfamiliar with advanced scheduling techniques should stick with this type, as little can be done here to get you in trouble. The Finish-to-Start (FS) logic type (Exhibit 4) simply states that the succeeding activity cannot begin until the preceding activity has completed. An example of the FS constraint is that the foundation of a house must be finished before the walls can be built.

Finish-to-Finish

The Finish-to-Finish (FF) logic type, see Exhibit 5, is used to identify that a successor activity cannot finish until the predecessor activity is finish. This is normally used when both activities can be performed for the most part in parallel; however some piece of work must be accomplished before the other piece can be accomplished. The FF logic is normally accompanied by either a lead or a lag (discussed later) to separate the finish dates of the activities.

FF Logic

Exhibit 5: FF Logic.

An example of the FF logic is that a document cannot be finalized until all comments have been received and incorporated. The predecessor in this case is the comment period with the successor being the incorporation of comments. This indicates that once some comments have been received, work can begin incorporating or responding to those comments but the document is not complete until all comments are received and handled.

Start-to-Start

The Start-to-Start (SS) logic type (Exhibit 6) is used to signify that for a succeeding activity to begin, a preceding activity must first begin. This is normally accomplished when both activities can be performed for the most part in parallel; however some work must begin before work can begin on the other activity. The SS logic is normally accompanied by either a lead or a lag to separate th start dates of the activities.

SS logic

Exhibit 6: SS logic.

Typically the SS logic is used when material and labor are separate activities in a schedule, but there could be other instances of use. With that basis, think of construction and the pouring of concrete. The predecessor activity would be “Pour Concrete” with a SS to “Finish Concrete.” In fact, if the finish work does not begin as the concrete pouring starts you will have a mess on your hands.

Start-to-Finish

The Start-to-Finish (SF) logic type (Exhibit 7) is used very sparingly. The SF logic is not intuitive, but it is defined as the successor activity being required to start before the predecessor activity can finish. This logic will provide for some distinctly different results based on the tool being used. If you can't find a critical path, try changing any of these logic types that you may have in the schedule to one of the other types. This may be as simple as rethinking and realigning the current logic.

SF logic

Exhibit 7: SF logic.

It is best practice to not use this logic type; I mention it only to ensure that the type is known. Realistically, there is no valid use for this type of logic when hard logic used.

Leads/Lags

Leads and lags are a way to offset the logic that is set into place. Most tools define this usage as a lag, but allow the lead as a negative value of the lag. There are basically two rules of thumb when utilizing the lead and lag philosophy:

FS logic should not have lags applied as this normally would be a function of not accounting for an activity. This is risky in that not accounting for an activity could impact the work within the schedule. Hiding the fact that some activity is taking place could be a hindrance in maintaining the schedule. For instance, pouring concrete footer with a FS logic to erecting walls includes a 3 day lag. This illustrates that the cure time for the concrete is not accounted for in an activity and is contained within the logic.

Most tools will allow percentages to be applied for leads and lags. Use percentages versus duration for the lead or lag. The reasoning behind this is that if durations are used and the durations change or the work changes, the lead or lag may not be accurate any longer. For instance, while we would normally place an FS on a design build type of activity, we would normally accept an 80% solution of the design before beginning the building of our software product. This is accomplished with an FS and a 20% lead (FS-20%). If the design is 30 days as originally planned and the lead is defined as 6 days, and then the work is updated for design to be only 15 days, we would be trying to build the program with only about a 50% design solution. This would be very risky.

Defining Resources

Definition of resources is a multistep process. Ensure that these steps are followed to provide for the most executable schedule. There are subject matter experts (SMEs) from whom you will be getting information who will question this approach only because they have not been exposed to a structured process to build the schedule. Most of these SMEs find that they have been involved with projects that deviated greatly from their original schedules; but again, this is because these schedules were not built in a structured way.

Define the Resource Type

When reviewing the tasks for work to be accomplished, the number of resources (both human and nonhuman) must be ignored. The only thing to worry about at this step is ensuring that you identify the types of resources required to accomplish the activity. Do you have the right expertise assigned? There are assumptions made when assigning the resource, but then again, how valid are the assumptions? For instance, when identifying a programmer, you assume that they have the computer and tools required when placed on your project, therefore the computer and tools are not identified as additional resources. But what if the organization requires the programmer to lease or procure those types of resources, and he or she cannot? Now you have a programmer, but nothing they can work with. Don't forget about all tools that may be required to accomplish the activity at hand.

Ensure that you identify those resources that are consumable, renewable, labor, and nonlabor. Tools can aid in monitoring and control of the project later on by reporting to these differing resource types and determining what can be done based on the resource type.

Define the Amount of Work for Each Resource

For all resources, you will need to define the amount of work required. Again, ignore the quantity of resources at this time, only the amount of work which needs to be accomplished. For instance the activity may require 40 hours of work to accomplish something, but that could be one person for one week, or five people for one day as well as many other combinations. However, at this point, you are just identifying the amount of work to accomplish, and not the number of resources required to accomplish the work.

At this point, the tools used for scheduling will consider all resources as a single (100%) resource for the activity, which will automatically calculate the duration of the activity.

Defining Durations

The last step in building your schedule is to define the durations of the activities. This will enable you to determine how many of each type of resource are required as well as the time it will take to complete the project. Note that a great many SMEs will tell you they cannot tell you how long something should take until they know how many people they have to work on something, or that they can't tell you how many people they need until they know how long they have to complete the task.

Resource Efficiency Factor

The main focus here is to ensure that you have a plan of attack to determine some intermediate milestones, inch-stones, or other points in the schedule that are not necessarily a requirement date, but rather a preferred expectation for dates. In addition, unless you are in a truly project-oriented organization, you cannot assume a resource is available 100% of the time. This assumption will always come back and bite you. What is a company priority today (your project) may not be that way throughout the life of your project. I always plan resources in a nonproject-oriented organization at only about 75% efficiency. Basically, I plan on 6 hours of work for every 8 hours I need. Face it, people are pulled away from work to attend meetings, answer calls, or simply because they need to step away from their desks from time to time. Planning in this manner will assist in producing a more executable schedule. If you achieve better efficiency than 6 hours of work for every 8 hours scheduled, you can consider yourself lucky, and yours will be among the few projects that will actually perform ahead of schedule. For truly project-oriented organizations, that number may approach closer to 90% utilization, but never 100%.

Three-point estimates

How many of us are willing to go to our boss and inform him or her that “the plan for project execution is complete and I have a 50/50 shot at accomplishing the plan”? But each time you work a schedule with only a single point for your duration, those are exactly the odds you are facing. Many studies have been conducted and empirical data has proved that a single-duration schedule will be executed on-time and on-budget only approximately 50% of the time. Placing a three-point estimate on durations provides for a couple of key advantages to your schedule execution:

Three-point estimates by performing a simple PERT (Program Evaluation and Review Technique) ((Optimistic + 4 times the Most Likely + Pessimistic)/6) calculation increase the project execution plan up to about 73% based on empirical information.

Three-point estimates are required in order to perform a proper risk management analysis of the schedule as well as running a Monte Carlo simulation against the schedule (the simulator needs to know boundaries for each activity.)

Analyzing the Schedule

Now the schedule is complete. The next thing is to see if what you have planned actually makes sense. There are several aspects to this analysis. One is simply to ensure you are meeting your contractual obligations. Once that is ensured, then we use other techniques to determine that the schedule is valid. There are many methods used to analyze the schedule, but the critical path method is by far the most dominant method and is the best understood, and is therefore deemed the best practice

Critical Path

The critical path is defined as “longest path through the project” (Project Management Institute, 2008), which means that there may or may not be slack in the schedule, but that the schedule's longest path to completion is through the critical path. With this in mind we must analyze the schedule to ensure that our proposed critical path actually makes sense to us from a programmatic view. Does it line up with the thinking of our SMEs that this is really the “longest pole in the tent?” There may be tuning to the resources required, the durations, or even the logic, as we go through the analysis of the critical path. There may even be places where constraints (discussed later) may be required in order to ensure that the critical path is planned correctly.

One thing to remember is that most of the tools use zero (0) as the definition of the critical path for slack. This is adjustable and in many cases should be adjusted to ensure the critical path is meaningful. Of course changing this value may provide for a multimodel (more than one) critical path. There is nothing wrong with this, just more to manage, and this normally provides a strong point of review and management of the plan.

There are other aspects to the path which must be analyzed to ensure an accurate planning schedule is developed. Associated with the critical path are the near critical path and the risk path, as discussed below. Near Critical Path

While the critical path is the longest path through the schedule, attention needs to be given for any task that may become a real candidate for a change to the critical path. These are viewed as items with a certain amount of total slack. What is determined to be on the near critical path actually depends on the project, but as a general rule of thumb this would be anything that has less slack than half of the reporting period. In other words, if you are planning to report status on your project in a biweekly fashion (10 days), then the near critical path would be defined as any path with less than 5 days of slack. Of course, this number will vary according to the project.

Risk Path

Although related to the critical path, the risk path is truly a different way of looking at the schedule. The risk path is viewed as the riskiest path through the schedule. A specific path, when defined, may have 50% total slack of the project, yet if one risk occurs, that path becomes the critical path. In this instance, the path would not have shown up on the critical path or the near critical path, yet it now “comes to bite us” due to the risk that has occurred. This must be reviewed when we look at the schedule, and then we must determine how to handle it within our schedule.

A good example of the risk path would be that we have a decision in our plan that we will procure (buy) a specific item to complete the project. One risk we have identified is that the provider of the item may go out of business. If that provider actually does go out of business, we will need to develop (make) the item which is needed for the project.

Constraints

Constraints are normally defined as placing a date constraint on an activity that alters the actual logic placed within the schedule. Best practice would state that a properly constructed schedule would have zero (0) constraints in the baseline schedule. Of course, in actual practice, this will not always be the case. If one is following the guidelines defined under the logic section, there may be items that simply do not fall into place once the schedule is developed. If you must use constraints, do so very carefully and with caution.

Hard Constraints

Hard constraints are defined as those that drive an activity to a specific date. These constraints normally contain the word “must” in their definition (e.g., “must start on,” “must finish on”). These constraints, while available, should not be used, as any other logic in the schedule would then be driven by this constraint and the critical path or critical chain would become invalid. If this is used in the schedule, any usage in excess of 0.1% of total logic would define the schedule as invalid. While keeping to this percentage can be used as a guideline, certain auditing organizations will fail a schedule if any hard logic is used.

Soft Constraints

Soft constraints are a bit more forgiving, yet are still restrictive to the algorithm in the scheduling tools. These types of constraints do constrain the activity to a point, yet still allow it to move based on the type used. Soft constraints are of the type “finish no later than,” which allows a task to finish earlier, but does not allow the task to be scheduled to finish later. Each one of the soft constraints has this same type of feature in that it will constrain one end of the activity based on its structure. Note that all activities have a constraint, the default being “as soon as possible.” This type of constraint is what is normally used and provides for the cleanest schedule; however, the drawback is that management and others may not be aware that what you are actually providing is the quickest schedule possible, an early start schedule. When you cannot finish the base schedule in that time frame, many question why the schedule has slipped, when in fact you have stated (by default) that this is your earliest schedule possible. That noted, care should be taken when using any soft constraint other than “as soon as possible,” as the possible scheduling algorithms may provide results that are not anticipated. Another popular constraint type is just-in-time (JIT) scheduling, when we use “as late as possible.” Of course, this leaves you no room (slack) for that activity and could result in missed deadlines. While these constraint types can be useful, care should be taken and anything other than “as soon as possible” should not exceed about 8% of the total types in use.

References

Department of Defense Handbook. (1998) MIL-HNBK-881, Washington DC; Department of Defense.

Project Management Institute. (2008). A guide to the project management body of knowledge (PMBOK® guide)—Fourth edition. Newtown Square, PA: Author.

Project Management Institute. (2008). Practice standard for work breakdown structure—Second edition. Newtown Square, PA: Author.

This material has been reproduced with the permission of the copyright owner. Unauthorized reproduction of this material is strictly prohibited. For permission to reproduce this material, please contact PMI or any listed author.

© 2009, Elden F. Jones II, MSPM, PMP, CMII

Originally published as a part of 2009 PMI Global Congress Proceedings – Orlando, Florida