Project Management Series: Part 2 – Scheduling & Managing Resources

INTRODUCTION

Steven M. Barger, Greenbrier & Russel:

Good day, and welcome to . . .
The Project Management Series,
Part Two, Scheduling and Managing Resources.

In Part 1 of this course, we discussed "Defining and Planning the Project." This included setting goals and determining scope - which are very important in starting the project off right.

We showed how the project manager must balance time versus cost versus scope.

In this course we are going to continue that discussion by going over scheduling and managing resources.

SYLLABUS

Today, we will to go through:

> building the project team,

> developing estimates,

> defining tasks and deliverables,

> determining those tasks'
  dependencies,

> putting it all together: developing the
  work plan using task estimating and
  resource leveling, and

> how to refine the work plan when
  everything is not quite in sync.

BUILDING THE PROJECT TEAM

One of the things that we need to do in order to manage our resources is to build the project team.

The first step to building the successful project team is to make sure that you have a handle on all of the roles that you, as project manager, will be managing.

Those roles include yourself - the project manager - the architects, the project sponsor or executive sponsor and almost, quite literally, a "cast of thousands."

The Project Manager

Let's start off by talking about the project manager.

We already know that the project manager is the one who manages project goal trade-offs of time, cost and scope. We certainly can't forget that. He or she also allocates people resources, recommending the people to bring onto the project team. It's advantageous to the project manager if he or she can handpick his or her team.

The project manager must have some general skills similar to a samurai. These include: credibility, sensitivity and leadership.

Credibility means that project managers must be believable and trustworthy.

They should have expertise in the business functional areas, and they should have expertise in the technical architecture.

Remember, though, that you don't have to be an expert - as a project manager - in the technical architecture. You should be familiar with it. This is good, because it's difficult to find people who are technically proficient in each of the new technologies that we work with today.

Second, the project manager should have sensitivity.

This involves being politically correct. It also involves setting and communicating expectations to both users and the team. And, it certainly involves managing the inevitable conflicts, and overcoming the obstacles that you're going to face day-to-day as you run a project.

Project management skills also include leadership, which can be viewed as maintaining focus. Project managers need to focus on the high-priority tasks. They also need to be comfortable that they can track the progress of those tasks, and compare estimates to actuals.

They have to have confidence - they have to exude confidence from the pores, if you will. Confidence is contagious. If you, as the project manager are confident, your team will be confident as well, especially in embarking on a new journey.

They should have charisma - and maybe just a little bit of charm.

They should have authority, because there are going to be some hard decisions to make, and the decisions will likely get funneled up to them. And as we said before, the project manager is the one that must able to get the job done.

The Architects

Another role we need to discuss is that of the architects.

While the project manager is responsible for designing the team, the architects are ultimately responsible for designing the system. The architect's roles can be broken up into four types: systems architect, application architect, technical architect and data architect. Each of these has unique responsibilities within your project.

Their titles may change. Roles may be shared by two or more people. But these are the types of roles you are likely to see in the new technologies.

Systems architects are just like the chief architect in a building construction project. They have total responsibility for the product and they oversee the other architects. They must understand the impact one area of the project has on another, and they must coordinate all the other architects.

Application architects, on the other hand, could also be referred to as lead application developers. They are the ones who are closest to the users.

They are responsible for the user interface, the look and feel of the system. They are responsible for the applications tools suite: "Are we going to be using the development tools of PowerBuilder or Visual Basic, or maybe some other new tool?" They are also responsible for the applications code.

Technical architects are responsible for the technical infrastructure, and for communications. They have to determine what changes need to be made to the infrastructure to handle the new throughput.

They have a big hand in selecting the hardware - if that hasn't already been done - to help determine if there's enough horsepower in the selected platform. They also need to take a look at the system software, ranging from the mundane such as back-up and recovery systems to the system services for n-tier processing.

The next type is the data architect. Sometimes referred to as lead data analysts, they need to identify and select the appropriate database whether it be DB2, Oracle, Sybase, Informix or some other. This is no easy task in today's changing database market, with database vendors constantly leapfrogging one another.

They are responsible for data partitioning. That means determining where the data goes - what data goes where.

They have to determine how to access the data. Will it be SQL? Will it be stored procedures? Will they write some sort of system services to access the data?

Finally, they are responsible for data modeling. What will the database structure look like?

Architects, in general, must have experience in the appropriate technologies. This is certainly easier said than done, especially as new technologies keep moving forward.

The architects must have architectural vision to inspire a project to great success.

The architects must help the project manager anticipate possible failures, and build contingency plans in case those failures come to light.

Finally, the architects must communicate. Probably as much as 90% of their time will be spent doing this, to keep everyone in sync.

The Project Sponsor

Another role that the project manager needs to consider is that of the project sponsor.

Now, the project sponsor may already have been chosen for your project. They are sometimes referred to as executive sponsors.

The executive sponsor serves as a protector for the project, warding off budget cuts and unwarranted changes.

Project sponsors also serves as overseers - keeping the goals in sight. They serve as enforcers, reminding the project manager of critical tasks that must be completed in order to keep the project on track.

They also serve as cheerleaders - rallying the troops, especially through the tough times.

And, they empower the project manager, which may be an informal process or a formal project chartering.

You need to know who your project sponsor is because lots of projects have many different project sponsors, and it may not be clear to you.

The "Cast of Thousands"

Another aspect that project managers need to consider is all of the other roles - the "cast of thousands." These roles are many and diverse, especially as we get into new technologies.

These people will include business analysts who perform your analysis. They'll include GUI design experts to build your user interfaces. They'll include data modelers to model the data. They'll include database analysts to look at performance for whatever your selected database is.

They will include database developers, which is kind of a new twist. These folks are responsible for building the triggers and stored procedures, and they'll have to have some business knowledge in order to carry that out.

They will include, certainly, your application developers or programmers. They'll include middleware experts, operating system experts. They will include object designers and object librarians if you're pursuing an object-oriented approach.

They'll include quality assurance and testing analysts. They might include a "Webmaster." if you're doing an Internet or intranet project.

The project manager must coordinate the roles of all of these people.

Managing Resource Availability

Another part to building the successful project team is to ensure that you can obtain the necessary resources. This means that you must pick the resources at the right time, certainly. And, you need to look for people with the right skills.

Resource availability becomes a very big hurdle for the project manager, so let's take a little closer look.

One way you could manage resource availability is to determine how you are going to mix the experts with the novices - because certainly not everyone on your team is going to be an expert. Take a look at your staff's capabilities. Examine the required expert skills and see if they are hard to obtain in your market.

Have a process for verifying how you'll determine whether people are experts or not. You might consider using a maximum of two novices for each expert - or maybe fewer.

Consider using consultants. Many consulting companies are technology-focused. They can help mentor your staff, and bring those skills in house.

Don't overlook vendors. They're often very eager to spend time with you - to help build a successful client relationship, and to prove that their products work successfully.

Don't allocate your plan until the resources are booked. This is a very big issue - especially as we head towards the year 2000 - because many resources are already overbooked.

DEVELOPING ESTIMATES

Okay, now that you have a handle on the resources that you need, you'll have to estimate the cost of the project, and supply that estimate to management and users.

Let's begin by talking about what an estimate is. It's an approximation of the costs of the resources needed.

This includes the people who will perform the tasks on your project. It includes hardware. Be sure to consider both the development and production environments, especially if they don't exist today. It will include things like system software: your development tools, databases and the like.

Just as you made decisions regarding allocation of resources through the phases, you'll have to consider things such as additional design work. Will that save construction as you move forward on your project?

Always give your estimates as a range. As a rule of thumb, we like to say something like, "I'm ninety percent confident that the cost will be between 1.2 and 1.4 million dollars."

Also, consider planning for contingencies up front, especially for projects that are using significant amounts of new technology.

Estimating Techniques: Top-Down

Let's talk a bit about some of the techniques that you can use to estimate your project.

The first is top-down. This generally is a directive from above, and I think we're all pretty familiar with that.

The next is a little bit trickier. It's parametric - based on prior, similar projects, which may or may not have been in your environment.

The third is bottom-up. These estimates are developed at the detail level and summarized up to a total estimate cost.

Top-down estimates (for various tasks) are typically provided for you, so that you can estimate the total project cost. On the other hand, if you do them, they can be done early on in the process, to give management and users a ballpark figure of the costs.

Top-down estimates imply that, given constraints of time and cost certain functionality will follow. In other words, "Here's how much time and money you have. What can you produce?"

Top-down estimating is less costly to produce. It's also less accurate. You might use top-down estimating when there's a limited amount of detailed information, when you have previous projects that are similar in size and scope, and when you have a staff that has the required expertise, so that there are not a significant number of surprises.

Parametric Estimating

The next technique is what we call parametric estimating.

I've already mentioned that it's going to be a little trickier, because it can use a mathematical model to predict cost. It is based on previous projects, so it uses historical information. It uses parameters from the complex, such as function points or lines of code, to the more mundane or easy, for instance the number of windows.

An example of this might be if you have, let's say, 20 windows, and you've estimated that each window in your system will take approximately one month to design, code and test. Then you know that you have about 20 months of effort for this project.

The thing about parametric estimating is that the accuracy is only as good as the knowledge about each of the parameters.

If you were off on that one-month-per-window development time, for example, by a factor of two, that would throw your entire estimate off by that same factor.

Parametric estimating is used when you have a parametric estimating tool available. Well, where are those?

They often exist in the methodology, or in your project management scheduling tools. But parametric estimating requires accurate historical information regarding previous similar projects. Your project's size must be similar to the historical project's. If your new project is significantly larger, the model may not translate well.

Parametric estimating should be used when you have many repetitive tasks.

Bottom-Up Estimating

The third type of estimating technique is the bottom-up estimate.

This is where you estimate the cost of each individual task from the work breakdown structure or the work plan. What you do is summarize those costs to obtain a project total. This allows for easy incorporation into the detailed project budget - once you've fixed your budget.

Bottom-up estimating is likely to be very accurate. It's also more costly to produce because it takes longer. You want to consider using bottom-up estimates when the technology is known and accurate estimates are required.

Estimating "Sanity Check"

Now, once you've completed your estimating process, it's good to conduct a "sanity check." You want to check your estimates against, first of all, basic guidelines.

One rule of thumb you might consider is to allow between 12% and 15% (of the total budget) for your project management tasks. Don't forget these project management tasks. They are often assumed to be part of your project, and you want to make sure that you include cost and time estimates for them as well.

You might expect your construction budget to range from 150% to 200% of the design costs. You might also anticipate system tests to be between 30% and 40% of this amount.

Remember that what you're doing is checking the estimates you've come up with against basic guidelines, and these guidelines may vary for your shop.

Another sanity check factor is to ensure that you've allowed for contingencies. Either itemize the contingencies into each part of the project or build them into the phases and tasks.

But, make sure you don't double dip. Make sure you don't have people who are estimating bottom-up - building in a 20% or 30% contingency - and then you, as the project manager, add a contingency for the entire project. That's double dipping.

You might consider between 5% and 20% percent for contingencies. This really depends on how confident you are in your estimate.

It also depends on the amount of new technology. Has anyone in your shop ever done this before? Does your team have experience? Has anyone in the country ever done this before?

DEFINING TASKS AND DELIVERABLES

Now that you've defined the estimates, and you've got a pretty good handle on them, let's make a start towards building a project work plan.

We'll do this by defining the tasks and deliverables so that you can properly manage the project through the execution phase.

But before we start, you'd better confirm that everyone has a clear understanding of what the work breakdown structure levels are.

Consider a possible scenario in which your project breaks down into phases, which contains activities. Tasks fall under activities, and may or may not be broken into subtasks.

You'd be surprised at how many projects have different names for these levels. Make sure that you get clear understanding across your entire team.

Decompose the Work Breakdown Structure

The first step in defining those tasks is decomposition of the work breakdown structure, or WBS.

You want to do segregate the elements in the work breakdown structure further and further, to get to a greater sense of detail.

You identify the elements and tasks that are manageable so that they can be planned, so that you can estimate them at a very detailed level, so that they can be scheduled through, let's say, the course of several weeks, not several years.

You want to make them so they can be controlled. By controlled I mean that you can track them and compare what you planned to do to versus what is actually happening.

Some rules of thumb when you're doing the decomposition: first of all, it's hard to know when to stop. You might consider continuing until the task duration is no greater than around 15 days.

Some of your tasks might be more especially if you have a pretty good handle on what that task can do and there are not going to be a lot of technological surprises.

Another rule of thumb you might consider: get to the point where you can easily identify one or two roles that are required to perform the work necessary to complete that task.

If you have, let's say, four or five roles assigned to a given task, it's going to be very difficult to manage the time for all those people on one task.

WBS: Templates and Budgets-vs.-Actuals

One of the ways you can help define your tasks is to use templates. This is simply historical information from previous projects.

If, for example, you've built a WBS on a prior project, you probably already have the tasks and the estimates and the deliverables defined - either in the WBS or in your project work plan - and you'll find that the tasks are often usable from those previous projects.

Another way it can help is if the budget-to-actuals are tracked. This would occur during the project execution phase that we'll get into it a little bit.

This can assist you greatly in estimating. This is especially true if you have good historical information, and you're doing a project that's very similar.

WBS: Who'll Be Doing What ?

A step that will help you create the work breakdown structure (WBS) is to identify the person who is responsible for each area - for each task - on the work breakdown structure. Have each person who will actually perform a task drive down and specify the elements in the work breakdown structure.

This gives you a lot of advantages. First off, it buys you a lot of commitment and buy-in from the people who will actually be performing the tasks. It certainly helps improve your estimates, and should give you better confidence in them.

Finally, it improves your estimating accuracy, because you're not going to overlook any important details.

Define the Deliverables

Now that you've defined your tasks, it's time to define the deliverables - not necessarily a really exciting task, and something that people often overlook.

You need to ask: "What is the output for each task?" You must be able to define it. We suggest that each deliverable have its own unique name.

You should definitely consider including sample deliverables, so that your people understand the level of detail that you're trying to get.

The deliverables should be explained in detail. You should have supporting information and some examples.

A deliverable can be either a service or a product. For example, a service might be "LAN wiring is complete for the application developers." A product might be a document, such as a detailed design spec or a three-tier partitioning document.

The last thing about deliverables is to obtain sign-offs from users for all of the appropriate deliverables.

Make those decisions early on - which deliverables will require sign-offs - and get those sign-offs as they're completed.

DETERMINING TASK DEPENDENCIES

Now that you've defined the tasks, and you've identified the deliverables, it's time for some of the really fun stuff in project management.

It's time to determine the task dependencies - basically, a two step process.

First, we'll look at task sequencing.

Second, we'll look at the dependency relationships that exist between the various tasks.

In addition, we'll look at techniques for diagramming those task dependencies. These include Gantt charts and project network diagrams.

Task Sequencing

Identifying the inter-task dependencies is the big step in task sequencing.

You need to ask questions like: "What actions are sequentially dependent on which tasks? You need to remember that not all tasks have dependencies, so it's likely you'll end up with a work plan where many tasks can be performed in parallel.

The next step in determining task sequencing is to identify roles.

You'll need to consider the availability of people for certain roles. Be especially cognizant of the fact that there will be critical roles on your project - perhaps a three-tier architect or an object modeler - for which resources (people) may not be readily available at all times.

Finally, you'll need to ignore the calendar when you're determining task sequencing We're going to talk about this in just a little bit.

Four Types of Dependency Relationships

The next thing to do in determining task sequencing is identify the dependency relationships. There are four types of dependency relationships.

The first is finish-to-start. Second is finish-to-finish. The third type of dependency relationship is start-to-start and the final, and certainly the least common, is start-to-finish.

The first type of dependency relationship is the finish-to-start. Definitely the most common, it's probably what you're familiar with.

Finish-to-start means that the "from" task must finish before the "to" task can start.

Envision a "from" task, which is "install workstation PC," and a "to" task, which is "install the system software."

This relationship implies that the "install workstation PC" task must finish before the "install system software" can start.

The second type of dependency relationship is the finish-to-finish. This means that the "from" task must finish before the "to" task can finish.

Consider an example: our "from" task is "determine software" and our "to" task is "obtain development software."

The "determine software" task must finish before the "obtain development software" task can finish.

The third type of dependency relationship is the start-to-start. This means that the "from" task must start before the "to" task can start.

Consider an example. The "from" task is "install workstation PC" and the "to" task is "install PC videoconferencing."

This means that the "install workstation PC" must start before the "install PC videoconferencing" task can start.

The start-to-start dependency relationship is not very common.

The final type of dependency relationship is the start-to-finish. This says that the "from" task must start before the "to" task can finish.

We almost need to pause here because this is a little bit different way of looking at a dependency relationship.

Consider an example in which the "from" task is "determine GUI standards" and the "to" task is "obtain GUI development tool."

This means that "determine GUI standards," which is the "from" task, must start before "obtain GUI development tools" can finish. So we have a degree of overlap here.

The start-to-finish dependency relationship is rarely used, and it's somewhat difficult to understand.

Diagramming Task Dependencies

We'll finish up the task dependency section by examining two techniques for diagramming task dependencies.

The first is the Gantt chart. These are fairly familiar to all. They show the tasks on a horizontal time scale. They are really easy to read and pretty intuitive and easy to understand.

They can show planned and actual progress, and you can easily resequence using the Gantt chart editor.

You can clearly show dependencies from one task to another.

Another type of diagramming tool is the project network diagram. It's a schematic display of the project's activities and dependencies. It's used pretty extensively in government work.

It helps most by determining the critical path. We can define the critical path as a set of tasks that, if delayed will delay the project completion, or if accelerated will enable an earlier project completion.

Project network diagrams - outside of the government area - are not used much, but can really help you get a handle on what the critical path tasks are in your project.

Tools for Defining Task Dependencies

I don't recommend doing this task dependency and task sequencing by hand. Instead, I recommend that you look at some scheduling software. Most complex projects use project-scheduling software to perform the task sequencing and the task dependencies.

You might want to take a look at Microsoft Project, ABT's Workbench products, and others.

This can really help simplify your task sequencing, so that you can move towards completing your project work plan.

DEVELOPING THE WORK PLAN

Once you've completed your task sequencing and dependencies determination, you have a good start towards developing your work plan.

Let's look at the final major deliverable that you need to produce.

Estimating Task Duration

What you need to do next is estimate task duration, and if you thought you could forget about bottom-up estimating, it's here again. You need to perform bottom-up estimating in order to estimate task duration, and you'll do this for each of the work periods - whether you do it in hours, days or weeks.

You'll need to consider your effort versus your elapsed time, sometimes known as duration. Let me give you an example. "Order workstations" takes less than one day of effort, but "acquire workstations" can take many days elapsed time.

These are simple concepts, but often confusing to implement. Make sure you have these clear before you proceed.

In order to estimate task duration, the next thing you need to do is identify the named resource, and then use this as a base for your estimates.

At this point, you should identify those resources (people, hardware, etc.) by name. For example, an expert might take 5 days to complete a task while a novice would take 20, and your task estimates would vary as a result.

The next thing you need to do in estimating your task duration is to document all your assumptions.

You probably wouldn't believe how many times those assumptions are made and agreed on by all the people involved, and then not documented - and when you come back later, it's difficult to understand where that estimate may have come from.

I suggest that you always give your estimates as a range, for example, "I'm 90% confident that the time for these tasks will be between 5 and 6 days."

This also allows a little bit of slack for scheduling subsequent tasks.

You should definitely consider using project-scheduling software, because it's easy to enter and change the estimates, and will automatically perform the summary for you.

Putting All the Elements into Your Work Plan

Once all of the tasks are estimated, and your resources are identified, you're finally ready to develop the work plan. You do this by pulling together all the pieces that we've talked about so far: the work breakdown structure, the task dependencies, the estimates of cost and the estimates of task duration.

You will do the next couple of steps more or less at the same time - these are definitely not sequential steps.

You'll determine the start and end dates for each task. Take a look at the calendar, and see what can be accomplished within this time frame - and be realistic.

In developing the work plan, you also need to consider the availability of each of the named resources. Certainly people are a resource, but don't forget about hardware. Will the hardware that you need in order to perform some of your tasks be available?

A lot of people forget about things like development tools. Will they be ready for when your coders need to start the design and coding steps?

You also need to determine the maximum resource utilization per day for each of those resources. For people, we like to use a rule of thumb of 6 to six-and-a-half hours per day. This is realistic and allows for things like e-mail, breaks, vacations, sick time and holidays. You might also want to note if there are some significant holidays or vacations coming up. You may want to block that out.

Now, this whole process is iterative, and you're going to continue to do this until the work plan is complete. The tough part is knowing when to stop.

One of the ways you can do this is to stop when you know, and believe, that you have a schedule that is realistic to your people. This implies that you've developed the plan with the full involvement of the project team - so that they can believe it's realistic and you can obtain their commitment and buy-in.

Once you've done this, you might find that you need to smooth out the resource usage. We call this resource leveling. For example, John might be needed for 80 hours one week but only 10 the next.

Take a look at each task to see how you can distribute his time so that he's not working an 80-hour week and then taking off the next week.

Most project scheduling software tries to level, but it really can't. This ends up being a little bit of a manual effort, but what you'll end up with is a much better plan as a result.

REFINING THE WORK PLAN

Well, by now, you're pretty proud that you've created your preliminary work plan - only to find out that it's still not quite right. Because, by comparing your work plan to your original cost or time constraints, you'll find that it doesn't quite fit.

This is where you have to come in and refine the work plan just a little bit. You do this by comparing your work plan to the budget, or the preliminary estimate, that you had done early on in the process.

If your work plan is not in sync with the budget or the preliminary work plan - preliminary estimate - you need to cut the scope by taking a very hard look at what you've included in it to date.

This is where the scope and goals estimates and priorities that you had done early on can really come in handy - to highlight what things you might consider cutting.

You might take a look at refining your estimates. But I will caution you: don't just cut estimates for the sake of cutting. Base your refinements on additional information you've gathered.

You might consider adding resources. Resources, if you have them available, may shorten the duration of some tasks, but at the same time add costs to the entire project.

Two of the more advanced methods for refining the work plan are the critical path method and fast tracking, which we'll start to discuss now.

The Critical Path Method

The critical path method, or CPM, involves the project manager trading time for cost, or vice versa. Normally it's the first - the project manager has to trade out some time by adding cost.

There are basically two approaches to the critical path method.

The first is normal CPM. This is the realistic estimate of duration that you've built into your work plan to date. It includes all the leveling and resource assignments that you've done.

The second is crash CPM. This involves expediting by adding resources, such as putting your people on overtime for a certain period, or adding staff temporarily in certain places to get you over the hump.

Normal CPM

Let's define normal. Normal is the plan that you've created to date, and normal is characterized by having slack time. No matter how you've resource-leveled, no matter how you've done your dependencies, there is always going to be slack time - and slack time is the time between the tasks that are not on the critical path.

You'll always know these tasks, because they can be delayed without penalty to other tasks or projects. They're the ones you don't worry about as much.

Slack time in itself - and don't let others in management hear you say this - allows a little bit of additional contingency.

I wouldn't necessarily recommend that you build a contingency into the slack. But as a project manager, you always know it's there.

Consider an example of a normal project network diagram, as follows. First, tasks on this critical path are identified as one-dot-one-dot-one, one-dot-one-dot-two and one-dot-one-dot-five.

This is somewhat of a standard nomenclature for the work breakdown structure, so we're using those terms here.

You should feel free to translate the tasks into real task names.

In this example, one-dot-one-dot-one on your critical path takes 5 weeks. One-dot-one-dot-two has a 2-week duration. And one-dot-one-dot-five has a duration of 5 weeks.

In addition, one-dot-one-dot-three and one-dot-one-dot-four are tasks that you're performing on your project, and they're done in parallel. In this example, there is slack time between one-dot-one-dot-four and the remainder of the critical path.

Crash CPM

Crashing involves adding resources to tasks on the critical path in order to compress the time. Basically, we're looking to cut the duration.

Consider making the previous example a crash project. As a result of crashing the critical path becomes one-dot-one-dot-one, one-dot-one-dot-two and one-dot-one-dot-five, which is the same critical path. But the duration on some of these tasks will have been shortened.

One-dot-one-dot-one has been shortened (from 5) to 3 weeks. One-dot-one-dot-two is already at its shortest possible duration, and one-dot-one-dot-five has been shortened (from 5) to 4 weeks.

The shortened duration of the crash project is now 9 weeks, versus 12 weeks for the original, normal project plan. On a project network diagram, this shows up very well.

Now crashing does not work for all tasks. I'd like to say that it takes one elephant 22 months to have a baby elephant, but you can't really crash this process by having 22 elephants take one month.

What's more, this adds costs - possibly from overtime or from adding resources - and may decrease efficiency.

As a result of crashing, the critical path may change each time a task is crashed. In the example, there was some change in the critical path. But we ended up with the same critical path.

Crashing is also risky. It removes the contingency that's contained in the slack time. Above all, it assumes your most optimistic estimates, which really may not reflect reality.

Crashing is iterative. This means that a total crash schedule is really unrealistic.

If you continue to crash to the shortest possible duration, it may not be feasible to carry that off.

One of the most famous examples of a crash project was the NASA Apollo 11 flight. President Kennedy had set the goal of getting to the Moon before 1970. NASA realized in about 1967 that, in order to make that happen by June or July of 1969, they were going to have to crash that project, and it's certainly the biggest crash project you can imagine.

Fast Tracking

Another method you can use when your work plan is not quite in sync with the budget or your preliminary estimates is fast tracking.

Fast tracking involves scheduling in parallel, phases, activities or tasks that would normally be done in sequence.

An example of this might be starting to code before your design is complete. Fast tracking may involve additional rework, because, for instance, you're making some assumptions if you start to code when the design is not quite done, and this substantially increases your risk.

Baselining: "Freeze" Your Work Plan

The final step, once your work plan is defined and refined, is to set a baseline against which your changes will be controlled.

This is known as freezing your work plan. You freeze task estimates. You freeze resource assignments. You freeze task dependencies.

Now, ideally, any changes that you make to your plan from then on will involve going through the standard change management process.

The final result of all the hard work that you put into creating your work plan will be a road map for your project.

And if you've done all the elements that we've talked about well, it will be a road map to success.

At that point, the project's budget can be accurately set, and you're ready to implement your plan.

SUMMARY OF "SCHEDULING AND MANAGING RESOURCES"

In this course, we learned how to plan for all required resources.

We learned how to develop estimates to the necessary accuracy.

We talked about defining tasks and deliverables, and their dependencies.

We showed how to develop the work plan by pulling together the tasks, resources and dependencies.

And we refined the work plan by cutting scope, refining estimates or adding resources.

In our next course, we're going to learn about how to manage the project.

FOR ADDITIONAL INFORMATION

If you need additional information, please check out . . .

www.gr.com.

Or feel free to e-mail us at . . .

sbarger@gr.com, or . . .

rskibski@gr.com.

Thank you very much for your time.

Computer Channel, Inc.
www.compchannel.com