CHAPTER 13
THE REAL SCHEDULE AND BUDGET

In many cases, the ERP schedule and budget are not worth the paper on which they are printed. Many project managers shoot themselves in the foot by prematurely committing to a plan that only sets the wrong expectations. Others throw darts to come up with a go-live date, or falsely assume the quote from the consultants is the gospel.

The problem is once unrealistic expectations are cast, they are not going away. Could this be why so many ERP projects fail? In fact, some argue many projects are not really failures, just a failure to manage expectations!

For most senior managers, the project schedule and budget are usually their biggest concerns. Providing the best possible answers is part of a project manager's survival guide, but there are also other implications.

Unrealistic time and cost commitments drive poor project decisions. Many project managers cut corners in an attempt to catch up to a plan that was never agreed upon or feasible from the start. On the other hand, most teams will rally around a schedule if they had input in developing it and believe in the plan.

Three Stages of Estimating

Estimating the project schedule and budget usually occurs in three stages: 1) Developing early estimates before a decision to proceed with an ERP software evaluation, 2) When securing project funding after evaluating software and consultants, and 3) Formulating a baseline schedule and budget during the planning phase which will be used to manage the project. Each stage of estimating goes deeper into the details since there is more information available as the project progresses.

As with all estimates, always include your list of assumptions, constraints, and risks. If the project takes too long and cost too much, and this list is not documented, management will not recall all the things that drove the previous estimates.

Early Estimates

The project manager should anticipate that senior management would want at least high-level estimates before making any type of financial commitment to the project. There are many ways to develop estimates, but it will never be a science. Below are my general guidelines for a project timeline based on company revenue or the number of users:

- Smaller companies (under $100 million, or under 125 users) = 9 to 16 months
- Medium companies ($100 million to $250 million, or 125 to 250 users) = 16 to 24 months
- Larger companies ($250 million to $1 billion, or 250 to 600 users) = 24 to 36 months
- Very large companies (over $1 billion, over 600 users) = 36 to 48 months

Many factors will affect the actual timeline, but for the purpose of ballpark estimates, the four major assumption areas driving the ranges above are: 1) Project team time commitments (full or part-time), 2) The number of modules to install, 3) The software rollout strategy (Big Bang or phased), and 4) The amount of software modifications expected.

A quick, yet fairly accurate way to estimate total project cost is to back into the numbers based on the cost of the ERP software. For example, the cost of ERP software historically represents roughly 20% of the total project cost. Start the cost estimating process by getting two or three initial quotes for software packages. The packages should be within the software tier associated with your company size (see Chapter 5). Next, calculate the average package cost, and multiply this number by four or five (depending on how conservative you want to be) for the estimated total project cost.

The total estimated cost can be broken down into the major implementation cost categories by using some commonly observed percentages. These include:

- Consulting Services (PM, application, technical) = 50%
- ERP Software = 20%
- Application Development Services (programming) = 12%
- Hardware & System Software = 10%
- Education and Training = 8%

Project Funding Estimates

Undoubtedly, some type of project funding request is necessary before purchasing software and hiring consultants. Those who must approve the funding will want to know the project schedule and cost. This will require more research and getting further into the specifics of the project.

In fact, one should also review the next section of this chapter (developing the baseline estimates) prior to completing the funding document, particularly the list of potential budget items.

When managed correctly, the bidding process for selecting ERP software and consultants can yield a much better understanding of the project. Therefore, gather as much input as possible from all ERP software vendors and consulting firms quoting the project, not just from those selected.

As outlined in Chapter 6, first inform all vendors that planning assumptions should accompany all quotes and that low-balling schedule or consulting cost will not necessarily result in winning the bid. This requires that each vendor get more into the project specifics. Go as far as to ask each vendor for a detailed project plan, and then take what you can get.

In addition, all vendors want the project to be funded. They will usually go out of their way to provide additional information beyond what they are selling. Of course, during the bidding process all of the above is free. Take advantage of this while it lasts.

Second, at this point the consulting cost estimates are mainly a by-product of the project timeline. Have each firm quote consulting hours by project phase in weekly buckets for each consultant over their entire estimated timeline. Beyond scope and other assumptions, this requires them to think through the activities within each phase, the level of effort involved, and their responsibilities versus those of the client.

It is also important to work with all consulting firms to reconcile differences in their quotes. If all firms are really quoting the same project, timeline and consulting costs should not vary significantly. This can (and should) be accomplished without sharing quote information between vendors. Inform the firms suspected of low-balling that they are underestimating the project. Next, let them make the adjustment they see fit, not simply by matching another firm's bids.

Also, remember that you (not the vendors) control what is submitted to

management in the funding request. Even when insisting on valid quotes for consulting services, quotes can be so far off that adjustments upward of 50% are not unusual.

A good technique to estimate the overall timeline and budget is to start with the longest quoted schedule and cost (regardless of the firm selected) and compare against the guidelines previously mentioned for making early estimates. You might find it necessary to adjust upward.

Since at this point the ERP package and associated technology options are known, the IT group should be able to obtain quotes from third party vendors based on the technologies they will likely select. Early quotes for hardware and system software are usually in the ballpark (if there is a good estimate of the number of servers required).

For the project-funding request, always be conservative with the return on investment (ROI) since there is a human tendency to overstate benefits and underestimate costs. In the end, if the project cost is significantly underestimated, the project manager will get the blame.

Remember, the approved funding sets the *limit* on the total project budget, not individual line items; so provide some buffer. On the other side of the equation, it is my experience that senior management is usually more forgiving of projects that take longer than stated in the funding document. Therefore, the focus should be on the total cost.

Finally, if senior management has concerns about the project cost, by all means, revisit the numbers, but always change the project assumptions to reflect how the lower cost was justified. This might go as far as changing the project objectives and scope. If you cave now to unreasonable demands and reduce the costs without changing the assumptions, you will regret it later.

Baseline Estimates

Early estimates are necessary, but the project manager needs a handle to manage activities and expenditures on a daily basis. Unlike other plans or estimates prepared previously, the *baseline* schedule and budget represent the detail required to manage deliverables.

The baseline is also a tool to measure progress and actual costs for specific line items. This is why it is called the baseline since it is the yardstick by which all actuals or changes are measured.

The baseline schedule should be aggressive, yet achievable. It is prepared during the latter stages of the planning phase. At this point, we know much more about the project. The objectives, scope, and software rollout strategy are known. The implementation teams are now in place, and the software consultants are engaged to perform preliminary analysis and to assist with planning. In addition, senior management and project team education and software training and the *As-Is* analysis should be complete or well underway. In other words, for the first time, we have more information and all the project resources are in place to develop a real schedule and budget.

The major phases and deliverables outlined in Chapter 7 are a starting point for developing the schedule, plus sample or template ERP project plans are easy to obtain.

When using a consulting firm, project scheduling is an area in which their project management expertise should be leveraged to the hilt. One of these areas is how to use project scheduling software, when this knowledge does not exist internally. Finally, get educated on project scheduling concepts, principles, and tools when necessary.

The Detail Schedule

The right scheduling is a "top-down, bottom-up" approach, with several iterations. It is top-down in that it begins by *decomposing* (breaking down) the major phases and deliverables into specific tasks and task *dependencies* (i.e., Task A must complete before Task B can begin).

When breaking down the project work, at some point activities should be organized around each software module and business process. The reason is that current process analysis, the *To-Be* processes, configuration set up, testing, etc., are associated with modules and eventually business processes. It is best to plan the project the same way.

The scheduling process is bottom-up in the sense that the dates to start and complete major phases and deliverables are derived by the associated lower level tasks dependencies and *durations* (elapse time required to complete a task). Therefore, the project master schedule is not pulled out of thin air. It should be based on what must occur to complete each deliverable and thus the entire project.

Once the project management team constructs a first-cut schedule, they

validate it for structural integrity. This ensures all tasks are included, dependencies are properly linked, resources are assigned, and task durations pass the test of reasonableness.

During validation of the schedule, pay special attention to the tasks on the *critical path*. This set of related tasks, when added together, determines the total project duration. For items on the critical path, it is prudent to revalidate tasks durations and dependences. Most project scheduling software will identify items on the critical path.

Next, determine the workloads placed on each team member by time-period and compare these to their committed project hours for the period. Adjust the schedule or task assignment accordingly. As imperfect as it is, most scheduling software contains some type of rough-cut capacity planning capabilities.

After working the schedule within the project management team, review the draft with the entire project team. This is an interim sanity check to gather more input prior to finalizing. It is an important step because those implementing the plan must believe and support it. It is best to get team "buy-in" early, rather than first presenting the schedule to the team after it is final. In the latter case, the project manager could become the only one committed to making it happen.

The Critical Path

Part of finalizing the schedule is to work the critical path to complete the project sooner. One way to compress (crash) the critical path is to shorten the task durations by adding more resources. Nevertheless, continuing to add more resources does not necessarily continue to reduce the time to complete a task proportionally. Eventually, the law of diminishing returns applies.

Another technique is to look for the *earliest possible start date* for items on the critical path. It may be possible to start some tasks sooner than the "hard" serial task dependencies suggested in the existing plan. Usually, there are critical activities that can run somewhat concurrently with others on the critical path, thus reducing the overall timeline. The best way to do this is to break up a single task into multiple tasks and then start the first one earlier.

For example, let us assume Task 2 can only start after Task 1 is complete (according to the current plan). Upon further analysis, Task 2 may really con-

sist of Task 2A and 2B. It may be possible to run Task 2A concurrently with Task 1. Task 2B is now dependent on the completion of Task 1 and 2A. We have now taken the time for task 2A off the overall project timeline.

Finally, another technique is to decouple task dependencies for a few major items on the critical path and simply *force* them to start earlier. For example, software development is normally on the critical path (this is one reason to minimize software modifications). Nevertheless, it might be possible to get a jump on the design of a few high priority or difficult data conversions, interfaces, or software mods prior to the "official" design phase (of course, if the steering team approves the mod).

If you take this approach, treat each piece of major development work as a separate project, but linked with the overall project. These "mini-projects within the project" might have dedicated resources.

Of course, there are risks in starting something too early and encountering more rework than it was worth. The key is to find the "sweet spot"—when these types of tasks can start early without causing excessive rework.

Use of Slack Time

Once the critical path is set, another step in finalizing the plan involves using *slack time* effectively. Tasks not on the critical path have "slack." To a certain extent, it does not matter when these start or finish as long as completed before they become critical.

All ERP projects have slack activities, so use this time to your advantage by scheduling non-critical items when most convenient to the team, or to smooth the workload when resources appear overloaded.

Protect the Schedule

Murphy's Law says what can go wrong, will go wrong. Nevertheless, there is such a thing as being proactive to increase the likelihood that important events materialize as planned or to mitigate known project risk. The final step in building the schedule is to "protect" it by adding additional steps to help ensure that activities occur without major problems.

Often times, these additional steps are easy to identify and perform. For example, prior to an important software demonstration with stakeholders, do a quick "dry run" with only the team present to make sure the system and the

team is prepared for the demo. Again, it is easy to identify proactive tasks to add to the plan by asking: "What can go wrong when performing this important task?"

Once the project management team is satisfied with the baseline schedule, a second review should occur with the rest of the project team to make any final adjustments prior to the project kick-off meeting.

Scheduling Mistakes

Building a schedule is always somewhat subjective, but the goal is to make it a good predictor of reality. Without attention to the items below, a schedule can be grossly understated.

1. **Inadequate Task Definition**
   When developing the plan, breaking project tasks down to extreme levels adds little value in the end. However, thinking through what must occur within each activity helps isolate certain steps that should be treated as separate tasks.

   A classic example of this is software development. Creating a single task for a software development project called "XYZ Programming" does not reveal the true scope of work. Developing software requires more than physically writing programs. It also involves analysis, specifications, and unit testing. In addition, these activities involve more than just the software developer. Most software development also requires the participation of the application consultant and functional analyst.

   Another example of poor task definition is failure to include tasks that are necessary to *transition* between projects phases (i.e., wrap up the work on one phase and prepare for the next phase). As an example, it is not realistic to assume the team can complete the first round of conference room pilot testing on Friday and immediately jump into round two testing on Monday.

   The final major pitfall in this area is not recognizing *planned rework*. Planned rework is different from unnecessary rework, since planned rework should be anticipated and can be of benefit to the project. However, unplanned rework is a project manger's worst enemy.

For example, the purpose of a design review is to gather feedback and input from stakeholders outside the team. Therefore, an expected outcome should be improvements in the system design considering the feedback. The process of updating the system design or set up to reflect the changes is rework, but it is "good" rework. Make sure you account for this type of rework in the schedule.

2. **Tasks Dependencies Are Not Well Defined**
   Almost every task should have at least one *predecessor* task (something that must complete first). Most project scheduling software identifies tasks with no predecessors.

   When it is difficult to identify a predecessor task for an item, the activity should be broken down further in order to build a definitive dependency relationship with other tasks in the schedule.

   Second, a task can have more than one predecessor. Since the accumulation of related tasks comprises the critical path, a single task not properly linked as a predecessor can result in an incorrect critical path. You do not want the critical path determined by simple mistakes when setting up task dependencies.

3. **Task Durations Assume Execution Is Flawless**
   The duration of each task is based on what needs to be accomplished, the estimated level of effort, and applied resources. Certainly, avoid building unnecessary fluff in the schedule. However, when implementing ERP, there are learning curves, unknowns, and meeting cycles, and as a result, not all decisions are made quickly. Account for these realities in task durations.

4. **Theoretical Versus Actual Hours**
   Even when the implementation team plans to allocate the agreed upon hours per week to the project, usually not all of this time is actually applied or 100% productive.

   Of course, always account for vacations and holidays when determining the amount of resources used for scheduling purposes. But it is not practical to include every conceivable item in the

schedule or outside the project that consumes project resources. For example, there are project status meetings, coffee breaks, and sick days. In addition, team members at times will be pulled away from the project to address a crisis within the business.

As a result, for each resource, plan for less than the average number of hours per week theoretically committed to the project. This may seem trivial, but if resource availability is overstated by only 3-4 hours per week for each resource, this number can really add up over the course of a project. For this reason alone, the actual timeline could exceed the schedule by months.

The Master Schedule

The master schedule (i.e., schedule of deliverables) specifies the start and completion dates for the project, each phase, each deliverable, and who is responsible for the deliverable. This schedule is a summarized version of the detail schedule in that it only depicts the highest level of the plan.

Again, no detail schedule will unfold exactly as planned, but when done with the due diligence; minor omissions, errors, or inaccuracies at lower levels tend to cancel each other out at the highest level. In this case, the master schedule is still an accurate picture. This is what is important since the ultimate goal of any project manager is to hit the phase and deliverable start and completion dates, not necessarily each individual task in the detail plan.

This does not mean the detail plan becomes irrelevant once the master schedule is published. The detail is important later in planning weekly activities, assigning action items, gauging progress, making adjustments, and simulating the effect of proposed scope or resource changes.

The last step in preparing the plan is presenting it to the steering team for approval. The master schedule is typically the only schedule in which the steering team is concerned (although most executives want to know if there is detail to support it). In fact, having the detail as back up during sr. management's review of the master schedule reduces the chance that they will seriously challenge it.

The Final Budget

If the cost estimates in the project-funding document provided some room for the unknown, the final budget will probably not exceed those estimates. Thus,

budgeting at this point should be a process of fine-tuning the numbers as more decisions are made during the planning phase.

When finalizing the budget, the biggest advantage is we now have a real project schedule with assigned resources. Better yet, the steering team and other major stakeholders have reviewed the schedule and they support it. This allows for revisiting the estimated number of hours per week for each consultant, and lock down the final consulting budget.

One way to fine-tune the consulting hours is to "ramp up and ramp down" the hours to better reflect how you plan to use each consultant at different stages of the project. For example, during the planning phase, project management consulting hours per week should be at their highest. Once the project is launched, the hours should ramp down considerably and stay relatively flat until the cutover phase begins.

How low one can go with project management consulting hours depends on what the project manager (assigned from within the company) is capable of doing, and the level of support required. Once the planning phase is complete, project management consulting could be perhaps one day per month or one day a week. If a PM consultant is required more than two days per week, one may question whether the client project manager is capable performing the job.

In terms of application consulting hours, fewer hours should be planned for completing the preliminary analysis, the *As-Is* analysis, and when the project team is attending software training.

Preliminary analysis runs concurrently with the planning phase and this time is necessary for consultants to become oriented to your business and the project. However, it is not an efficient period during most projects since the project is still somewhat unstructured. So be careful not to overkill consultant hours during this time. A couple days per week during the preliminary analysis phase should be enough.

As discussed further in Chapter 17, most organizations can conduct the current process analysis on their own. An application consultant should be able to review the completed process maps, ask questions, and provide feedback without attending these meetings. This should take a consultant only a few hours versus setting through many meetings.

If the consultants are to conduct the team software training, this cost is part of the education and training budget. If the team attends training not provided

by the consultants, during this time the consultants may not have anyone to work with or anything productive to do. If so, do not schedule them during this time.

During the prototype, design, and construction phases, application consulting hours progressively ramp up since more support is typically required during this time. How much it ramps up depends on how quickly the project team can learn the software. Once formal testing begins, consulting hours should again ramp down since testing should always be the primary responsibility of the organization.

Hours should increase for all types of consultants during the cutover phase so they can assist with final preparations. If the system is truly ready for go-live, consulting support after cutover should be heavy for only two weeks or so. Hours should then ramp down, and end completely within about four to five weeks.

In establishing a weekly budget for application consultants, as a rule 40 hours a week on a routine basis should be avoided since this means consultants are camping out on the project. On the other end, eight hours per week is not an efficient use of consulting time either. Time is wasted, as the consultant gets reoriented to the project every week. The hours scheduled for a consultant for most weeks should be zero or 16-32 hours.

Finally, the consulting budget is just a budget. The project manager will determine the *actual* schedule to be released to each consultant as the project progresses. This is important because if you give consultants all the budgeted hours, they will burn every minute of it whether it is necessary or not.

List of Budget Items

Figure 9 below lists budget items to consider at any stage of estimating:

Figure 9 – Potential Budget Items

EDUCATION AND TRAINING	SYSTEMS HARDWARE	CONSULTING
Sr. Management Education Industry Practices Education ERP Software Project Team Training IT Systems Administration Training IT Application Development Training Implementation Tools Training Books / Societies / Seminars	ERP Servers 3rd Party Bolt-on Application Servers Internal Network Upgrades Telecom Service Upgrades PC's Printers Data Collection Devices Mobile Devices Storage Access Network (SAN)	ERP Readiness Assessments ERP Software Selection Industry Practices Consulting ERP Project Management ERP Application Consulting ERP Software Technical Consulting 3rd Party Systems Technical 3rd Party Bolt-on Applications Go-Live Readiness Assessment ERP Change Management
APPLICATION SOFTWARE		
ERP Software Package 3rd Party Bolt-on Applications		
IMPLEMENTATION TOOLS	**SYSTEMS SOFTWARE**	**CONTRACT PROGRAMMING**
ERP Application Development Tools ERP Application Set Up Tools 3rd Party App Development Tools 3rd Party Report Writers/Data Query Project Scheduling Software Process Mapping Software Other Implementation Tools	ERP Package Proprietary Middleware Operating Systems Data Base Software Web Services Software Client / PC Software 3rd Party Middleware / Integration EDI Software Forms / PDF Management Job Scheduler Data Back-Up / Recovery	Data Conversions Reports Interfaces Software Modifications
		ANNUAL SUPPORT FEES
		ERP Software Maintenance All Systems Software Maintenance Hardware Maintenance Outsourced IT System Infrastructure Outsourced ERP Application Support ERP Software-as-a-Service Implementation Tools
ADDITIONAL STAFFING	**OTHER**	
Project Team IT Department Temporary Staffing to Backfill Temporary Staffing to Load Data	Facilities and Equipment Upgrades Travel and Living Expenses Hardware Shipping Cost Project Team Incentive Bonus Sales Tax Contingency Factor	

Note: Some of the software and tools listed above may be included with the ERP software package.