Project Management Institute

Practical Advice for Software Implementation Project Managers

By Steve Reichenbach, PMP

How is it possible that an effective project manager with over 10 years of experience leading software projects could burn through three months of effort for a "simple project" that was supposed to take three weeks? Well, it happened to me, but it doesn't have to happen to you, if you are smart enough to learn from my mistakes!

> "When the project was completed, we had used over 300 hours ... more than 2,000% what had been anticipated!"

Risk management is a topic receiving a lot of deserving attention in the project management community. The moment that anything unexpected has a dramatic negative impact on one of our projects, we become very receptive to learning what we could have done differently to avoid or at least to have anticipated such problems. This is the essential nature of a risk management plan: to spend some time during development of the project plan to think about what might go wrong and to take action to prevent or detect those problems. While other authors have done a fine job of explaining risk management theory, I hope to add to the discussion by offering some practical advice based on my own experience specifically to equip project managers who take on a software implementation project.

My nightmare software implementation project began in February 2008 and was expected to be completed before the end of the month. Our company had been using a third-party document management system for about five years without installing any updates. The project was to add some user licenses and to install an upgrade from version 3 to version 6. As the project manager, I interviewed the vendor and created a project schedule listing the major tasks involved in the project and estimated the duration of each task.

The vendor was handling all of the software upgrade work, using a remote desktop access provided by our production support team. The vendor estimated that the upgrade work would take 12 hours to complete. When the project was finally completed in April 2008, we had used over 300 hours ... more than 2,000% what had been anticipated!

Both the vendor and the project manager underestimated the complexity of the upgrade project. As a result the project:

- Was completed over budget
- Used more resource time than allocated
- Was not completed on time
- Produced deliverables that were buggy
- Frustrated the system users
- Damaged the reputation of the vendor and the project team

One of the root causes of the project problems was the lack of time spent in the planning phase. As the project manager, I trivialized the project, saying to myself, "This will be a fast and easy one!" I stuck to the bare minimums in the project plan (scope, schedule and resource plan). I spent less time on the "simple" project, and focused more time on other projects. The other project stakeholders shared my optimism about the project, and I believed them.

- I believed that the task list was complete
- I believed that the level of effort estimates were correct
- I believed that the requirements were well understood

As a project manager, I like to be the "can-do" person with a positive attitude and enthusiasm. This positive trait, however, must be balanced with a more suspicious attitude when you start work on the risk management plan. Effective risk management requires less optimism, and therefore can require a great deal of effort for people who are normally very positive. Not only do you have to force yourself to think about what may go wrong, you also need to *believe* that these things probably will go wrong—otherwise you won't be watching out for them.

Software Application Risks

So, here's a list of the specific issues that surfaced during my software implementation project:

1. We purchased a new system sight unseen, without requiring the vendor to provide a demonstration that proved it would actually work.

The software application we were upgrading featured two methods for importing new documents: FTP and scanning. We needed both options to work, as some of the documents were being scanned and indexed off-site, and other documents were being scanned locally. As a global company, it was not practical for all sites to ship their documents to a single location for scanning.

During the planning phase, we discussed this need with the vendor. For some reason the vendor didn't realize that we were the first customer in their history to try to implement *both* methods in the same system. Knowing only that both methods worked, the vendor assumed incorrectly that setting up both options in the same system would be no problem.

Solving this problem required developing significant database and application modifications, and single-handedly cost the project two weeks of delay.

2. We allowed unqualified resources to work on the project.

The application database in the old version was Microsoft Access, and in the upgraded version would be Microsoft SQL Server 2005. The vendor indicated that they would convert the data from the old database to the new one. Within the first few minutes of the conversion effort it became obvious that the vendor's developers were much more familiar with Microsoft Access and were not very familiar with SQL Server 2005.

Some of the problems we experienced could have been identified by the vendor, if more rigorous quality assurance methods had been in place. I incorrectly assumed that every vendor does a good job with quality assurance. In our case the application had been tested with Microsoft SQL Express (a free download) but had never been tested with Microsoft SQL Server 2005.

As the project manager, I should have stopped work and called for help from our company's DBA (database analyst) resources. Because asking for help would have slowed down the project (which was already late), we spent about a week letting the vendor struggle with the database work. In retrospect, we should have reached out for help much sooner.

3. The key project resources worked only four hours per day due to time zone differences.

Struggling with time zone differences is not a problem just for global companies. In our case, because the vendor was in California and we were in (relatively nearby) Texas, I hadn't anticipated that a two-hour difference would have so much impact. Unfortunately, even this slight time difference had a *tremendous* impact.

The upgrade was a small deal for the vendor, so they had no incentive for working nonstandard hours. Our resources also considered the upgrade to be a low-priority project and insisted on working regular hours. Every day at 11:00 a.m. CST we would start work until lunch time at noon CST. We would work another hour at 1 p.m. CST until noon PST (yep, lunchtime in California). At 3 p.m. CST we would work for another two hours. The failure to negotiate working times during the initiation of the project resulted in a 50% reduction in the availability of key project resources.

4. The project team was allowed to show up late for meetings.

Adding insult to injury, the project team did not strictly follow the schedule outlined above—for example, showing up late after lunch without any prior warning. On some days, the vendor simply did not show up.

5. The vendor failed to reset expectations when the original plan was not met.

It was frustrating that when the project began to run very late, the vendor never engaged us to adjust the original plan. As the project manager, I constantly had to request meetings to revise the required level of effort for each task, tracking the time spent and estimated time remaining. During each follow-up discussion the vendor's estimates remained very short in duration. I heard "that will be done tomorrow" every day for about a month before the vendor admitted that their estimates were too optimistic.

> "Ask the following 20 questions when planning a software implementation project."

6. The vendor didn't bother to ask for help when problems arose.

Eventually I requested and received help from our own production support team. The vendor never asked for help because they didn't want to appear incompetent. To save face, they would struggle with problems for days without asking for help. When I was forced to reschedule the project (when it became clear we were going to be very late), I realized that the vendor had not been asking for needed help and reached out for support.

The reaction from our production support team was very positive. Although they had hoped to be more involved up front, they were still very glad to become involved in the project at this point. The participation of a company developer and a database administrator really helped move the project forward to completion.

Lessons Learned

During the closeout phase of the project, the project team (including the vendor) met to discuss what went right, what went wrong, and what we would do differently next time.

The classification of this project as a "simple upgrade" combined with the decision to move forward without rigorous analysis of the business needs led to poor planning and contributed to an underestimation of the effort needed.

Lesson learned: Take the time to ensure that the business needs are clearly defined, even for "easy" projects!

The original time estimate for the installation of the upgrade was 12 hours, a total level of effort of 24 hours of company time plus 12 hours of vendor time. Actual effort was over 200 hours.

Lesson learned: Use the Program Evaluation Review Technique (PERT) estimation technique combined with risk analysis to produce better estimates.

Ask the following 20 questions when planning a software implementation project.

Ask the vendor:

1. May I talk with some of your recent customers/references whose needs are similar to our needs?
2. What, if anything, are we doing in this project that is unique or that has never been done in quite this same way before?
3. How much experience do you have with the technology we will be using?
4. Can I see a demonstration of the system setup the way we will be using it?
5. What resources should be available to help, in case we run into problems or have questions?
6. Can the system requirements document be improved to make it clearer?
7. What hours during the day will our resources be available to work on the project?
8. What milestones should we use to check our progress against the plan, and to detect problems sooner?
9. What are the rewards for finishing on schedule?
10. What are the rewards for delivering a working system that meets all original expectations?
11. What three problems typically happen during a project like this, and describe how you would like to address them if they arise?
12. What other commitments have you made that may distract resources from this project if it takes longer than expected?
13. What problems might prevent us from being able to complete (this task) on time? *(repeat question for each task)*
14. Can you restate (this task) for me, to ensure we understand it the same way?
15. Dependencies: What needs to be done before we can complete (this task)?

Ask yourself, as the project manager:

16. Have I planned enough slack time in the project to accommodate unforeseen problems?

17. Do I need a risk management plan?
18. Do I have an effective way to communicate confidence level?
19. Do I have a generally understood method for reporting if I am ahead of schedule or behind schedule?
20. Am I being too optimistic?

PERT

The Program Evaluation Review Technique (PERT) provides a time estimate, or level of effort estimated to complete a task. The primary advantage of using this technique is that it provides the project manager with a confidence level, and it forces the project resources to consider the impact of project risks when estimating the level of effort.

- Use PERT to prevent a five-week task from being planned as a 12-hour task
- Use PERT when a project resource doesn't want to provide an LOE estimate
- Use PERT to measure your confidence that project tasks will be completed on time

$(O + P + 4M)/6 = T$

O = Optimistic
"If everything went better than expected, how soon might we complete this task?"

P = Pessimistic
"If almost everything went wrong, how long might it take to complete this task?"

M = Most Likely
"In your (expert) opinion, how long does this task normally take to complete?"
T = Time estimated to complete the task: use this estimate for the project plan

Standard Deviation

The standard deviation is an expression of the confidence level of the time estimate, based on the variation of the pessimistic and optimistic estimates. Generally speaking, the higher the standard deviation, the less confidence we have in the estimate.

Low standard deviation doesn't always mean the estimate is "good." The estimator could be confidently wrong!

$(P - O)/6 = S$

About the Author

Steve Reichenbach is currently a project leader with Walmart ISD in Bentonville, Arkansas. Steve began leading software development and implementation projects in 2000 at RealPage in Carrollton, Texas where he developed and implemented multifamily property management systems. Steve has also served as an IT program manager at McAfee in Plano, Texas. Steve has a bachelor of science degree in Business Management from LeTourneau University and is a certified Project Management Professional (PMP) and has earned the ITIL Foundation certification.