Managing Successful Software Projects:

Estimating and tracking the right way

Proper estimating and tracking are keys to successful software development projects. Without them you may be fated to manage a late project. With them, you will find the ultimate key to happiness and acclaim (well maybe not … but you will enjoy your work quite a bit more).

Software development is an area of project management that is notorious for late and unsuccessful projects. I have managed software development projects for more than 15 years and have found some key tips to delivering successful software projects.

Software projects today are being managed differently from the way they were managed in the past. Formal processes are more widely used and the level of experience of the managers involved is better than ever before yet many projects are still delivered late or over budget or both. In the year 2008, 51 percent of big corporate software development projects surveyed were defined as "challenged" by the Standish Group of research advisers in Dennis, Mass.

Perhaps one reason is that software development poses some unique problems. Two areas where development projects often get into trouble are estimating and tracking.

Estimating Faults

Many software development projects fail due the lack of proper estimates. If your estimates are way under, as is the case in many software projects, then there is probably no way to meet your original timelines and budgets. Software projects are often very difficult to estimate for a wide variety of reasons.

Why is estimating software projects so darned hard? Here's why:

- Detail: Writing software is about handling a large number of details: everything from the way the main program does its calculations, to writing code to change the look of the buttons on the screen, to writing code that displays the many hundreds of possible errors in a readable manner to the user. When estimating the time to create a piece of code it is very easy to overlook some of this detail.
- Uniqueness: Though attempts have been made to make software code reusable, this effort has had only limited success. Most software projects and most modules within those projects are new to the developers writing them and often new to the world. Most software developers spend their careers writing code to do very new things every project. On a construction project, if you had only put in windows before could you give a good estimate on building a staircase?
- Software developers are optimists: Estimates by developers are usually too optimistic. You can trust the estimates of some experienced developers and software architects but I have never run into a software person who pads his estimates too much. Often the opposite is the case.
- Unexpected problems will occur: In software, bugs caused by a mistake in single line of code can take days to track down. Other problems can occur when a piece of third party software doesn't perform as expected and you can spend weeks trying to find out if it is a bug in your code or in the vendor's code. These types of problems can and do occur, are hard to plan for and can invalidate your project schedule.
- The attitude that planning and process are a waste of time: This attitude was prevalent in software development a few years ago. Some of the comments heard include:
    - Software is creative and we can't interfere with the process
    - Paperwork takes away from development time

- The schedule is too tight for us to spend time properly designing the solution
- The documentation is the code (it isn't)

Of course, avoiding proper planning only results in late projects or worse a project that doesn't meet the needs of the end user and ends up being scrapped.

Software Project Problem areas

Estimating

- Detail: It is difficult to get detailed enough
- Uniqueness: almost every project is new.
- Software developers are optimists: check their estimates
- Unexpected problems will occur
- Many in the software world will believe planning and process are a waste of time

Tracking

- Difficult Bugs will occur so you better know about them and fix them.
- Gold plating and tangents: are developers focusing where they should?
- Drift: if the project is trending toward lateness, you should know about it early

Tracking Mishaps

Even with good estimates and planning, a project can run into problems during execution. Here are some problems to watch for:

- Difficult Bugs: Developers will always run into bugs. I have seen a developer spend the better part of two days trying to track down a problem caused by a single letter spelling mistake.
- Gold plating and tangents: Some staff will start working on a task and quickly use up half the time on the task building the world's greatest configuration utility, a top of the line help system or a module in that sexy new computer language. None of this work may be necessary, planned for, or good for the project. Ensure developers know their deliverables on a weekly basis and don't spend their time doing unnecessary work.
- Drifting Away: Not monitoring closely enough can result in your dates slowly drifting to the point where your project will be inevitably late. Nearing the end of the project, it will be too late to take action and get it back on course.

Solutions

Those are some issues to watch out for. Is there anything a project manager can do to avoid these problems? Here are some tips:

Estimate right the first time!

- Ownership has its advantages - have the developers own the estimates in project plan. The best way to ensure you have an accurate schedule that the team believes in is to base it on their estimates. Require them to breakdown and list all their tasks and then estimate the effort. The team lead will also be involved in this process. When the planning is done, get them to agree to the dates. This process will give you better estimates to work with and will motivate the team to meet their deadlines.

- Detailed design - ensure that each developer has considered design. This doesn't have to be a long process but it needs to be detailed enough to show that the developer has thought through everything that needs to be done. Even a small amount of pre-planning and design will improve estimates and improve the quality of the product.

- Design review meetings - get the team together in a room to discuss the proposed design. What else needs to be done and the possible pitfalls should also be discussed. You will be surprised at the ideas put forth and the problems avoided.

- Bottom up is good! - the greater the level of detail, the less chance work will be missed in your planning. Work should be broken down to approximately 1-5 day long tasks. If your project is too large to make that practical, break down the project plan into smaller sub-project plans and delegate the estimating. If no one has gone to that level of detail, there is a good chance work will be missed. Put all your detailed tasks together to build the big plan.

- Top down is bad! – trying to get a plan to fit into dates defined from above without doing detailed estimating can get you into big trouble.

- Previous experience... – it is always better to learn from someone else's mistakes then from your own! If you have a similar project that was competed within your organization in the past or if you have access to records from a similar project elsewhere you may use this information as a basis for comparison or for creating your estimates. Best of all is if some of your team has done a similar project in the past. Do what you can to get these people or even consider hiring outside consultants with the right experience. Often, this type of information or experience is not available internally.

- Formal analysis – Formal methods such as function points exist for estimating software projects and are ways of estimating based on an analysis of the requirements for a project. Doing function point analysis requires training but can provide estimates before a detailed design is available and even before the team is assembled. It is not always a substitute for detailed design and planning. However, it may be worthwhile for high risk projects.

Track Religiously!

- Track Weekly – meet at least every week to discuss progress and to update your project plan. It will quickly become clear if a developer is having problems or is behind schedule. Group meetings are great for this; developers will not like to report they are late in front of the team and so are motivated to keep on schedule. If your project plan isn't being updated with actuals every week, why not?

- Have some contingency, darn it - for software projects always ensure you have adequate buffers built in. I sometimes add contingency to individual estimates from junior developers. Developers often don't factor in interruptions, meetings and phone calls when estimating. In addition, I add an overall contingency to the project as a whole. Even with this added contingency, which may even sound like overkill, you are very likely to find your project pushing the deadlines.

- Don't rush it - the biggest mistake starting software project managers make is to agree to compress the timelines. This doesn't work in most project management fields and is especially problematic in software development. Rushing through the writing of code is a risky exercise. Small mistakes made in a rush or by a tired developer can be costly. You may find the code written when a developer has stayed an extra six hours late one night ends up taking him or her two days to rewrite during the testing phase. An occasional late night or weekend can help get a project back on track but save it for emergencies; you shouldn't include it in your up front planning or over do it.

- See a problem, take action! – if dates start to slip, get help for that developer. If you're tracking weekly, you will know early enough to take action. Ensure you know if someone is stuck on a problem and try to help (asking questions can do wonders for expanding developer's thought processes even if you don't have the technical knowledge to give the solution). Get senior team members to help resolve tricky bugs. Get the team lead or a senior developer to help out.

- Stop the drift: If you start to see that your estimates are consistently off, you need to take action immediately. Otherwise, the chances of you delivering on time are very, very low. Renegotiating delivery dates or removing some functionality is much easier to do early in the project than towards the end of the project. Late in the game, capital may have already been spent on the project launch and changing it will be very difficult.

Avoiding the pitfalls:

Estimating

- Give the team Ownership of the dates
- Do Detailed design
- Use Design review meetings
- Estimate bottom up
- Take advantage of previous experience
- Consider Formal analysis

Tracking

- Track Weekly
- Have enough Contingency
- Don't rush it!
- See a problem, take action!
- Take early action on drift

Following these suggestions will improve the consistency of software development projects. Though other things can go wrong and other areas need to be managed, following these tips will maximize your chance of delivering a highly successful software development project.

It has worked for me: I have been consistently delivering success software development projects for many years now. Just follow the steps and you can too!

Pedro Serrador, MBA, PMP, P.Eng. is a project and program management consultant and speaker to large international corporations. He specializes in technically complex and high risk projects, vendor management engagements and tailoring and implementing project management methodologies.

He holds an Hons. BSc in Physics and Computer Science from the University of Waterloo, Canada and an MBA from Heriot-Watt University, Edinburgh, Scotland. He is an accredited professional engineer in computer engineering and has been a PMP since 2000.

pedro@serrador.net
www.serrador.net