Project Management Best Practices

21 ways to take the pain out of project management.
Tips to Manage Every Project Like a Pro

Managing software projects is difficult under the best of circumstances. As a project manager, you must balance competing stakeholder interests against the constraints of limited resources and time, evolving technologies and challenging demands. It’s like a juggling act with too many balls in the air at once.

Unfortunately, many new project managers receive little training in how to do the job. Learning survival tips from experienced and effective project managers can save you from learning the hard way. Jama Software has teamed up with requirements expert Karl Wiegers to provide you with project management best practices across five key categories:

- Lay the Foundation
- Plan the Project
- Estimate the Work
- Track Your Progress
- Learn for the Future

Remember, there are no “silver bullets.”

Every project is different and even the “best” practices are situational. Study this list to selectively and thoughtfully apply best practices only where you know they will add value to your project.
Lay the Foundation
1 Define success criteria.

At the beginning of the project, make sure all stakeholders share a common definition of success. Start by identifying your stakeholders and their interests and expectations. Then, define clear and measurable business goals such as:

- Increase market share by a certain amount by a particular date
- Reach a specified sales volume or revenue
- Achieve certain customer satisfaction measures
- Reduce expenses by retiring a high-maintenance, high-cost legacy system

Remember, business objectives must define your overarching goals. It won’t matter if you deliver on schedule and on budget if those factors don’t clearly align with business success.

2 Identify drivers, constraints and degrees of freedom.

Your project must balance its functionality, staffing, budget, schedule and quality objectives. Define each of these five project dimensions as either a constraint within which you must operate, a driver strongly aligned with project success or a degree of freedom you can adjust within some stated bounds.

An over-constrained project leaves you no way to deal with requirement changes, unexpected staff outages and/or risks that materialize or other occurrences — so take time up front to get the clarity you need.

Keep in mind

Business goals should imply specific project success criteria and always be measurable and trackable.
3 Define product release criteria.

Early on, decide what criteria will indicate whether your product is ready for release. Examples of possible release criteria include:

- No open high-priority defects
- Performance goals achieved on all target platforms
- Required functionality is fully operational
- Specified legal and regulatory goals are met
- Customer acceptance criteria is satisfied

Whatever criteria you choose, make sure they are realistic, objectively measurable, documented and aligned with what “quality” means to your customers.

4 Negotiate achievable commitments.

Despite pressure, never make a commitment you know you can’t keep. Engage in good-faith negotiations with customers, managers and team members to agree on goals that are realistically achievable.

When project realities (such as staff, budget or deadlines) change or unanticipated problems arise, it’s a good idea to renegotiate commitments right away. No one likes change, but it’s better to realign commitments than to wait for inevitable disappointment.
5 Write a plan.  
The hard part of planning isn’t writing the plan. It’s going through the actual planning — thinking, negotiating, balancing, asking, listening and thinking some more. To help guide you, consider these elements of a useful plan:

- Staff, budget and other resource estimates
- Team roles and responsibilities
- Assumptions, dependencies and risks
- Target dates for major deliverables
- Metrics to collect and analyze

Some plans may be more complex while others may require only a page, but it’s essential to always formulate and write a plan.

6 Decompose tasks to inch-pebble granularity.  
Inch-pebbles are miniature milestones that break large tasks into multiple smaller tasks to help you estimate them more accurately, reveal activities you might not have thought of and allow for more fine-grained tracking.

7 Develop worksheets for common large tasks.  
If your team frequently undertakes certain common tasks — implementing a new class, executing a system test cycle or performing a product build — consider developing checklists and planning worksheets for them. Not only will they help you avoid overlooking an important step, they’ll also help every team member identify and estimate the effort associated with each instance of the larger task he or she must tackle.

8 Plan to do rework after a quality control activity.  
Despite our best efforts, almost all quality control activities find defects or other needed improvements. Therefore, your project schedule should include rework as a task after every quality control event. Base your estimates of rework time on previous experiences, and if you have no rework, great — you’re ahead of schedule!
9 **Manage project risks.**

If you don’t identify and control project risks, they’ll control you. Yet simply identifying possible risk factors isn’t enough. You also have to evaluate the relative threat each one poses so you can focus your energy where it will do the most good. To properly manage each potential risk, select mitigation actions to reduce either the probability or the impact.

10 **Plan time for process improvement.**

Your team members are already swamped with their current project assignments. If you want them to rise to a higher plane of software development capability, though, you’ll have to invest in process improvement. That means setting aside dedicated time from your project schedule for improvement activities. It may not pay immediate dividends, but it’s an important strategic investment in your organization.

11 **Respect the learning curve.**

The time and money you spend on training should be thought of as an investment in sustained project success. You may pay a price in terms of a short-term productivity loss when you first try to apply new processes, tools or technologies, so make sure managers and customers understand this learning curve as an inescapable consequence of working in a rapidly changing, high-tech field.

---

**Perspective**

Process improvement is like highway construction: It slows everyone down a little bit, but after the work is done, the road is a lot smoother and the throughput greater.
Estimate the Work
12 **Estimate based on effort, not calendar time.**

People generally provide estimates in units of calendar time, but it’s better to estimate the effort (in labor-hours) associated with a task. Consider that a 20-hour task might take 2.5 calendar days of nominal full-time effort, or two exhausting days. Or it could take a week if you have to wait for critical information or get interrupted. If you track how you actually spend your time at work, you’ll know how many effective weekly project hours you truly have available.

13 **Don’t over-schedule multitaskers.**

There’s simply too much friction and task-switching overhead associated with multitasking. Some people multitask more efficiently than others, and even thrive on it. But if certain team members thrash when working on too many tasks at once, set clear priorities and help them do well by focusing on just one or two objectives at a time.

14 **Build training time into your schedule.**

In the high-tech field of software development, all practitioners should devote themselves to ongoing education, both on their own and the company’s time. Arrange training when you can, as the half-life of new technical knowledge is short.

**Did you know?**

Typically, the effective project time is only perhaps 50 to 60 percent of the nominal time team members spend at work. Of course, that’s far less than the assumed 100 percent effective time on which so many project schedules are planned.
15 Record estimates (and how you derived them).
When you prepare estimates for your work, write down those estimates and document how you arrived at each of them. Understanding the assumptions and the approaches used to create an estimate will make them easier to defend and adjust when necessary. It will also help you improve your estimation process. Train your team in estimation methods too, and develop procedures and checklists that everyone can use.

16 Use estimation tools.
Many commercial tools are available to help you estimate entire projects. Based on equations derived from large databases of actual project experience, these tools can give you a spectrum of possible schedule and staff allocation options. They’ll also help you avoid the “impossible region” — combinations of product size, effort and schedule where no known project has been successful.

17 Plan contingency buffers.
Projects never go precisely as planned. Therefore, it’s prudent to incorporate contingency buffers at the end of phases, dependent task sequences or iterations to accommodate the unforeseen. Use project risk analysis to estimate the possible schedule impact if several of the risks materialize. Then, build that projected risk exposure into your schedule as a contingency buffer. Some may see this as mere “padding.” To help persuade skeptics, point to unpleasant surprises on previous projects as a rationale for your foresight.

Are you sure?
If you choose to discard contingency buffers, you are tacitly absorbing all the associated risks and assuming that all estimates are perfect, no scope growth will occur and no unexpected events will take place.
18 **Record actuals and estimates.**

Unless you record the actual effort or time spent on each project task and compare them to the estimates, your estimates will forever remain guesses. If you write down what actually happened today, that becomes historical data tomorrow.

In addition to effort and schedule, consider estimating and tracking the size of the product in terms of requirements, user stories, lines of code, function points, GUI screens or other units that make sense for your project. This will give you more relevant context into time and effort so you can create right-sized estimates in the future.

19 **Count tasks as complete only when they’re 100 percent done.**

We give ourselves a lot of partial credit for tasks we’ve begun but not yet fully completed. But if someone asks you whether or not a specific task is complete and your reply is, “It’s all done except…,” then it’s not done! Don’t let people “round up” their task completion status. Instead, use explicit criteria to determine whether an activity truly has been completed.

20 **Track project status openly and honestly.**

Run your project from a foundation of accurate, data-based facts, rather than from the misleading optimism that can arise from the fear of reporting bad news. Use this information to take corrective actions when necessary and celebrate when you can. Remember, you can only manage a project effectively when you really know what’s done and what isn’t, what tasks are falling behind their estimates and why and what problems, issues and risks remain to be tackled.

---

Use responsibly!

The cardinal rule of software metrics is that management must never use the data collected to either reward or punish the individuals who did the work. The first time project leaders do that will be the last time they can count on getting accurate data from teams.
Learn for the Future
21 Conduct project retrospectives.

Retrospectives (also known as postmortems and post-project reviews) provide an opportunity to reflect on how the last project, phase or iteration went and to capture lessons learned that will help enhance future performance. During such a review:

- Identify things that went well, so you can repeat the successes
- Look for things that didn’t go well, so you can change approaches
- Think of events that surprised you and include them as risk factors in your next project
- Ask yourself what you still don’t understand about your projects, so you can continue to learn and improve

You’re Off to a Great Start — Now Keep Going!

These 21 project management best practices won’t guarantee a great outcome. They will, however, help you get a solid handle on your project and ensure you’re doing all you can to make it succeed in an unpredictable world.

Learn how the Jama Product Development Platform can transform the way you work so you can deliver better business outcomes.
ABOUT KARL WIEGERS

Karl has provided training and consulting services worldwide on many aspects of software development, management and process improvement. He has authored five technical books, including Software Requirements, and written more than 175 articles. Prior to starting Process Impact in 1997, he spent 18 years at Eastman Kodak Company. His responsibilities there included experience as a photographic research scientist, software applications developer, software manager and software process and quality improvement leader. Karl has led process improvement activities in small application development groups, Kodak’s internet development group and a division of 500 software engineers developing embedded and host-based digital imaging software products.

ABOUT JAMA SOFTWARE

Jama Software brings innovative analytics, solutions and insights to companies creating complex products and mission-critical software systems. With expanded product and service capabilities, the Jama Product Development Platform empowers large enterprises to accelerate development time, mitigate risk, slash complexity and verify regulatory compliance.

Representing the forefront of modern development, its rapidly growing customer base of more than 600 organizations — including SpaceX, NASA, Thales and Caterpillar — use Jama Software to streamline processes and bring complex products to market. Through Predictive Product Development, Jama equips its customers to make the most of their revenue potential and achieve ongoing competitive advantages.