jama build great products™

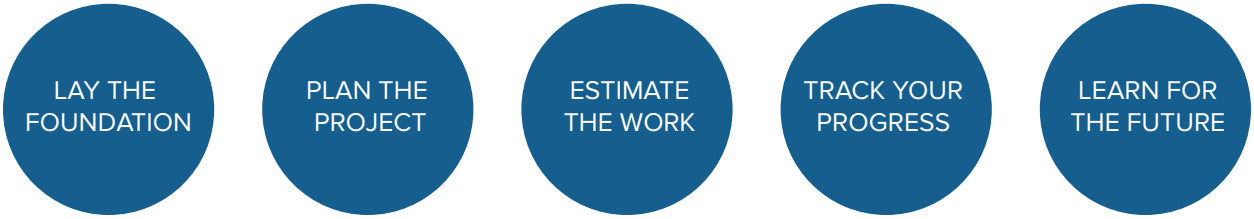
PROJECT MANAGEMENT BEST PRACTICES

Requirements expert Karl Wiegers introduces 21 valuable practices that can help both rookie and veteran project managers do a better job with less pain.

Managing software projects is difficult under the best circumstances. The project manager must balance competing stakeholder interests against the constraints of limited resources and time, ever-changing technologies and challenging demands from high-pressure people. Project management is 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. Anyone can learn to draw a Gantt chart, but effective project managers also rely on the savvy that comes from experience. Learning survival tips from people who've already done their tours of duty in the project management trenches can save you from learning such lessons the hard way.

This white paper, adapted from the book Practical Project Initiation: A Handbook with Tools (Microsoft Press, 2007), by requirements expert Karl Wiegers, is organized into five categories:



When initiating a new project, study this list of practices to see which ones would be valuable contributors to that project. Build the corresponding activities into your thinking and plans. Recognize, though, none of these practices will be silver bullets for your project management problems. Also, remember that even "best" practices are situational. They need to be selectively and thoughtfully applied only where they will add value to the project.

LAYING THE FOUNDATION

1 Define project success criteria.

At the beginning of the project, make sure the stakeholders share a common understanding of how they'll determine whether this project is successful. Begin by identifying your stakeholders and their interests and expectations. Next, define some clear and measurable business goals. Some examples are:

- Increasing market share by a certain amount by a particular date
- Reaching a specified sales volume or revenue
- Achieving certain customer satisfaction measures
- Saving money by retiring a high-maintenance legacy system

These business goals should imply specific project success criteria, which again should be measurable and trackable. These goals could include achieving schedule and budget targets, delivering committed functionality that satisfies acceptance tests, complying with industry standards or government regulations or achieving specific technology milestones. The business objectives define the overarching goal. It doesn't matter if you deliver to the specification on schedule and budget if those factors don't clearly align with business success.

Not all of these defined success criteria can be your top priority. You'll have to make some thoughtful trade-off decisions to be sure that you satisfy your most important priorities. If you don't define clear priorities for success, team members can work at cross-purposes[1].

2 Identify project drivers, constraints and degrees of freedom.

Every 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[2].

Bad news: not all factors can be constraints and not all can be drivers. If you are given a defined feature set that must delivered with zero defects by a specific date by a fixed team size working on a fixed budget, you will most likely fail. An over-constrained project leaves the project manager with no way to deal with requirement changes, staff turnover or illness, risks that materialize or any other unexpected occurrences.

Scenario: a senior manager and a project leader debate how long it would take to deliver a planned new large software system. The project leader's top-of-the-head guess was four times as long as the senior manager's stated goal of six months. The project leader's response to the senior manager's pressure for the much shorter schedule was simply, "Okay." A better response would have been to negotiate a realistic outcome through a dialogue:

- Does something drastic happen if we don't deliver in six months (schedule is a constraint), or is that just a desirable target date (schedule is a driver)?
- If the six months is a firm constraint, what subset of the requested functionality do you absolutely need delivered by then? (Features are a degree of freedom.)
- Can I get more people to work on it? (Staff is a degree of freedom.)
- Do you care how well it works? (Quality is a degree of freedom.)
- Can I get more funding to outsource part of the project work? (Cost is a degree of freedom.)

3 Define product release criteria.

Early in the project, decide what criteria will indicate whether the product is ready for release. Some examples of possible release criteria are:

- There are no open high-priority defects.
- The number of open defects has decreased for X weeks and the estimated number of residual defects is acceptable.
- Performance goals are achieved on all target platforms.
- Specific required functionality is fully operational.
- Quantitative reliability goals are satisfied.
- Specified legal, contractual, or regulatory goals are met.
- Customer acceptance criteria are satisfied.

Whatever criteria you choose should be realistic, objectively measurable, documented and aligned with what "quality" means to your customers. Decide early on how you will tell when you're done, track progress toward your goals and stick to your guns if confronted with pressure to ship before the product is ready for prime time[3].

4 Negotiate achievable commitments.

Despite pressure to promise the impossible, 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. Negotiation is required whenever there's a gap between the schedule or functionality the key project stakeholders demand and your best prediction of the future as embodied in project estimates.

Principled negotiation involves four precepts, as described in Getting to Yes by Roger Fisher, William Ury and Bruce Patton (Penguin USA, 1991):

- Separate the people from the problem.
- Focus on interests, not positions
- Invent options for mutual gain.
- Insist on using objective criteria.

Any data you have from previous projects will strengthen your negotiating position, especially because the person with whom you're negotiating likely has no data at all. However, there's no real defense against truly unreasonable people. Plan to renegotiate commitments when project realities (such as staff, budget or deadlines) change, unanticipated problems arise, risks materialize or new requirements are added. No one likes to have to modify his or her commitments. But if the reality is that the initial commitments won't be achieved, let's not pretend that they will right up until the moment of disappointing truth.

PLANNING THE PROJECT

5 Write a plan.

Some people believe the time spent writing a plan could be better spent writing code, but I don't agree. The hard part isn't writing the plan. The hard part is doing the planning—thinking, negotiating, balancing, asking, listening and thinking some more. Actually writing the plan is mostly transcription at that point. The time you spend analyzing what it will take to solve the problem will reduce the number of surprises you encounter later in the project. A useful plan is much more than just a schedule or task list. It also includes:

- Staff, budget and other resource estimates and plans
- Team roles and responsibilities
- How you will acquire and train the necessary staff
- Assumptions, dependencies and risks
- Target dates for major deliverables
- Identification of the software development life cycle that the project will follow
- How you will track and monitor the project

- Metrics that you'll collect and analyze
- How you will manage any subcontractor relationships

Your organization should adopt a standard software project management plan template, which each project can tailor to best suit its needs. You might start with the project management plan template available at www.ProjectInitiation.com. Adjust and shrink this template to suit the nature and size of your own projects.

If you commonly tackle different kinds of projects, such as major new product development projects as well as small enhancements, adopt a separate project plan template for each project class. The project plan should be no longer or more elaborate than necessary to make sure you can successfully execute the project. One page might suffice in some cases. But always write a plan.

6 Decompose tasks to inch-pebble granularity.

Inch-pebbles are miniature milestones (get it?). Breaking large tasks into multiple small tasks helps you estimate them more accurately, reveals work activities you might not have thought of otherwise and permits more accurate, fine-grained status tracking. Select inch-pebbles of a size that you feel you can estimate accurately. Inch-pebbles that represent tasks of about 5 to 15 labor-hours is a good start. Overlooked tasks are a common contributor to schedule slips. Breaking large problems into smaller bits reveals more details about the work that must be done and improves your ability to create accurate estimates. You can track progress based on the number of inch-pebbles that the team has completed at any given time, compared with those you planned to have done by that time.

7 Develop planning 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—develop activity checklists and planning worksheets for these tasks. Each checklist should include all of the steps the large task might need. These checklists and worksheets will help each team member identify and estimate the effort associated with each instance of the large task he must tackle. People work in different ways, and no single person will think of all the necessary tasks, so engage multiple team members in developing the worksheets. Tailor the worksheets to meet the specific needs of individual projects. They help avoid overlooking an important step in my rush to finish the project.

8 Plan to do rework after a quality control activity.

Some project plans assume every test will be a success that lets you move on to the next development activity. However, almost all quality control activities, such as testing and peer reviews, find defects or other improvement opportunities. Your project schedule should include rework as a discrete task after every quality control task. Base your estimates of rework time on previous experience. If you collect a bit of data, you can calculate the average expected rework effort to correct defects found in various types of work products. And if you don't have to do any rework after performing a test, great; you're ahead of schedule on that task. This is permitted in all fifty states and in many other countries. Don't count on it, though.

9 Manage project risks.

If you don't identify and control project risks, they'll control you. A risk is a potential problem that could affect the success of your project. It's a problem that hasn't happened yet—and you'd like to keep it that way. Simply identifying the 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.

If you don't identify and control project risks, they'll control you. A risk is a potential problem that could affect the success of your project.

Risk exposure is a combination of the probability that a specific risk could materialize into a problem and the negative consequences for the project if

it does. To manage each risk, select mitigation actions to reduce either the probability or the impact. You might also identify contingency plans that will kick in if your risk control activities aren't as effective as you hope.

A simple risk list doesn't replace a plan for how you will identify, prioritize, control and track risks. Incorporate risk tracking into your routine project status tracking[4].

10 Plan time for process improvement.

Your team members are already swamped with their current project assignments. If you want the group to rise to a higher plane of software development capability, though, you'll have to invest in process improvement. This means you'll need to set aside some time from your project schedule for improvement activities. Don't allocate 100 percent of your team's available time to project tasks and then wonder why they don't make any progress on the improvement initiative.

Process improvement is like highway construction: It slows everyone down a little bit for a time, but after the work is done, the road is a lot smoother and the throughput is greater.

Some process changes can begin to pay off immediately, but you won't reap the full benefit from other improvements until the next project. Process improvement is a strategic investment in the organization. Process improvement is like highway construction: It slows everyone down a little bit for a time, but after the work is done, the road is a lot smoother and the throughput is greater.

11 Respect the learning curve.

The time and money you spend on training, self-study, consultants and developing improved processes are part of the investment your organization makes in sustained project success. Recognize that you'll pay a price in terms of a short-term productivity loss—the learning curve—when you first try to apply new processes, tools, or technologies. Don't expect to get fabulous benefits on the first try, no matter what the tool vendor or the consultant claims. Make sure your managers and customers understand the learning curve as an inescapable consequence of working in a rapidly changing, high-tech field.

ESTIMATING THE WORK

12 Estimate based on effort, not calendar time.

People generally provide estimates in units of calendar time. It's preferable to estimate the effort (in labor-hours) associated with a task and then translate the effort into a calendar-time estimate. A 20-hour task might take 2.5 calendar days of nominal full-time effort, or two exhausting days. However, it could also take a week if you have to wait for critical information from a customer or stay home with a sick child for two days. Translate effort into calendar time on estimates of how many effective hours one can spend on project tasks per day, any interruptions or emergency bug fix requests, meetings and all the other places into which time disappears.

If you track how you actually spend your time at work, you'll know how many effective weekly project hours you have available on average. Tracking time like this is illuminating. Typically, the effective project time is only perhaps 50 to 60 percent of the nominal time team members spend at work, far less than the assumed 100 percent effective time on which so many project schedules are planned.

13 Don't over-schedule multitasking people.

The task-switching overhead associated with the many activities we are all asked to do reduces our effectiveness significantly. Excessive multitasking introduces communication and thought-process inefficiencies that reduce individual productivity. One manager claimed that someone on his team had spent an average of eight hours per week on a particular activity, so therefore she could do five of them at once. Forty hours per week divided by eight is five, right? In reality, she'll be lucky if she

can handle three or four such tasks. There's just too much friction associated with multitasking.

Some people multitask more efficiently than others, even thriving on it. But if certain of your 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 the schedule.

Estimate how much time your team members spend on training activities each year and subtract that from the time available for them to work on project tasks. You probably already subtract out average values for vacation time, sick time and other assignments; treat training time the same way.

Recognize that the high-tech field of software development demands that all practitioners devote time to ongoing education, both on their own time and on the company's time. Arrange just-in-time training when you can schedule it, as the half-life of new technical knowledge is short unless the student puts the knowledge to use promptly. Attending a training seminar can be a team-building experience, as project team members and other stakeholders hear the same story about how to apply improved practices to their common challenges.

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 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 the team in estimation methods, rather than assuming that every software developer and project leader is naturally skilled at predicting the future. Develop estimation procedures and checklists that people throughout your organization can use.

The Wideband Delphi method is an effective group estimation technique. This technique asks a small team of experts to anonymously generate individual estimates from a problem description and reach consensus on a final set of estimates through iteration. Participation by multiple estimators and the use of anonymous estimates to prevent one participant from biasing another make the Wideband Delphi method more reliable than simply asking a single individual for his best guess[5].

16 Use estimation tools.

Many commercial tools are available to help project managers 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. The tools incorporate a number of "cost drivers" you can adjust to make the tool more accurately model your project, based on the technologies used, the team's experience and other factors. You can compare the estimates from the tools with the bottom-up estimates generated from a work breakdown structure. Reconcile any major disconnects so you can generate the most realistic overall estimate.

17 Plan contingency buffers.

Projects never go precisely as planned. The prudent project manager incorporates budget and schedule contingency buffers at the end of phases, dependent task sequences or iterations to accommodate the unforeseen. Use your 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. An even more sophisticated approach is critical chain analysis, a technique that pools the uncertainties in estimates and risks into

> If a manager elects to discard contingency buffers, he has tacitly absorbed all the risks that fed into the buffer and assumed that all estimates are perfect, no scope growth will occur and no unexpected events will take place.

a rational overall contingency buffer[6].
Your manager or customer might view these contingency buffers as padding, rather than as the sensible acknowledgment of reality that they are. To help persuade skeptics, point to unpleasant surprises on previous projects as a rationale for your foresight. If a manager elects to discard contingency buffers, he has tacitly absorbed all the risks that fed into the buffer and assumed that all estimates are perfect, no scope growth will occur and no unexpected events will take place. Sound realistic to you? Of course not. Better to deal with reality—however unattractive—than to live in Fantasyland.

TRACKING YOUR PROGRESS

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. It's really not more complicated than that. Each individual can begin recording estimates and actuals, and the project manager should track these important data items on a project task or milestone basis. In addition to effort and schedule, you could estimate and track 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.

19 Count tasks as complete only when they're 100 percent complete.

We give ourselves a lot of partial credit for tasks we've begun but not yet fully completed: "I thought about the algorithm for that module in the shower this morning and the algorithm is the hard part, so I'm probably about 60 percent done." It's difficult to accurately assess what fraction of a sizable task has actually been finished at a given moment.

One benefit of using inch-pebbles (see Practice #6) for task planning is that you can break a large activity into a number of small tasks (inch-pebbles) and classify each small task as either done or not done—nothing in between. Project status tracking is then based on the fraction of the tasks that are completed and their size, not the percentage completion of each task. If someone asks you whether 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 is completed.

20 Track project status openly and honestly.

An old riddle asks, "How does a software project become six months late?" The rueful answer is, "One day at a time." The painful problems arise when the project manager doesn't know just how far behind (or, occasionally, ahead) of plan the project really is. Surprise, surprise, surprise.

If you're the PM, create a climate in which team members feel it is safe for them to report project status accurately. Run the 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 project status information and metrics data to take corrective actions when necessary and to celebrate when you can. 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.

An old riddle asks, "How does a software project become six months late?" The rueful answer is, "One day at a time."

The five major areas of software measurement are size, effort, time, quality and status. It's a good idea to define a few metrics in each of these categories. Instilling a measurement culture into an organization is not trivial. Some people resent having to collect data about the work they do, often because they're afraid of how managers might use the measurements. 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 you do this will be the last time you can count on getting accurate data from the team members.

LEARNING FOR THE FUTURE

21 Conduct project retrospectives.

Retrospectives (also called postmortems and post-project reviews) provide an opportunity for the team to reflect on how the last project, phase or iteration went and to capture lessons learned that will help enhance your future performance. During such a review, identify the things that went well, so you can create an environment that enables you to repeat those success contributors. Also look for things that didn't go so well, so you can change your approaches and prevent those problems in the future. In addition, think of events that surprised you. These might be risk factors to look for on the next project. Finally, ask yourself what you still don't understand about the project, so you can try to learn how to execute future work better.

It's important to conduct retrospectives in a constructive and honest atmosphere. Don't make them an opportunity to assign blame for previous problems[7]. It's a good idea to capture the lessons learned from each retrospective exploration and share them with the entire team and organization. This is a way to help all team members, present and future, benefit from your experience.

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

References

1. Chapter 4 of *Practical Project Initiation* presents a tutorial on defining project success criteria.
2. Wiegers explains this idea more fully in his book *Creating a Software Engineering Culture* (Dorset House, 1996).
3. See Chapter 5 of *Practical Project Initiation* for more about defining product release criteria.
4. See Chapter 6 of *Practical Project Initiation* for an overview of software risk management.
5. Chapter 11 of *Practical Project Initiation* presents a tutorial on the Wideband Delphi estimation method.
6. Chapter 10 of *Practical Project Initiation* is all about contingency buffers.
7. Chapter 15 of *Practical Project Initiation* describes the project retrospective process and provides a worksheet to help you plan your next retrospective.

ABOUT KARL WIEGERS

Karl has provided training and consulting services worldwide on many aspects of software development, management and process improvement. He has authored 5 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. http://www.processimpact.com.

ABOUT JAMA SOFTWARE

From concept to launch, the Jama product delivery platform helps companies bring complex products to market. By involving every person invested in the organization's success, the Jama platform provides a structured collaboration environment, empowering everyone with instant and comprehensive insight into what they are building and why. Visionary organizations worldwide, including SpaceX, The Department of Defense, VW, Time Warner, GE, United Healthcare and Amazon.com use Jama to accelerate their R&D returns, out-innovate their competition and deliver business value. Jama is one of the fastest-growing enterprise software companies in the United States, having exceeded 100% growth in each of the past four years, during which time both Inc. and Forbes have repeatedly recognized the company as a model of responsible growth and innovation. For more information please visit http://www.jamasoftware.com.