# The Project Postmortem: An Essential Tool for the Savvy Developer

**October 12, 2006**
**By Mike Gunderloy**

It's pretty rare for a software development project to go perfectly. In fact, although developers tend to have eternal faith that the *next* project will succeed completely according to plan, I can't think of a single project that I've been involved with over the past 25 years that went off without a hitch. Dealing with bugs, sudden requirements changes, hardware and software that break down mid-project, unexpected personnel turnover, baffling interactions between components, malevolent servers, and incompetent subcontractors seem to be the order of the day. The question is not so much whether you'll get hit with something unexpected, as how you'll deal with it - and what you can learn from it for the next project.

The difference between average programmers and excellent developers is not a matter of knowing the latest language or buzzword-laden technique. Rather, it can boil down to something as simple as not making the same mistakes over and over again. Fortunately, there's a powerful tool that any developer can use to help learn from the past: the project postmortem.

## Building an Institutional Memory

Though the name is well-established by now, "postmortem" has somewhat unfortunate connotations. The purpose of a good software postmortem isn't to carve up the corpse of a collapsed software project so as to assign blame for failure (though in dysfunctional organizations postmortems get used this way anyhow). Rather, the goal is to build up an institutional memory and develop a set of best practices that work for your own organization by meticulously recording what went right *and* wrong over the course of a project.

The difference between an organization with a culture of postmortems and one without can be dramatic. This is probably in part because of the good effects of postmortems, and in part because companies that lack the discipline to perform postmortems tend to be at a rather chaotic level of practice. When postmortems are institutionalized, you'll find people saying things like "we organize our source code tree this way, because we've found in the past that it works well" or "we stopped using that particular risk assessment practice because it just wasn't giving us any useful information." Without postmortems, developers are more likely to invent techniques as they go along, without much regard for what may or may not have worked in the past - and more likely to be surprised when something fails for the second (or third, or tenth) time.

Like most other organizational changes, effective use of software postmortems can't be mandated from above without employee understanding of the benefits, and it's very hard to sneak in from below without management support. But management can go a long way to create a welcoming environment for postmortems, and developers can educate management about the benefits. If you're not already using this practice in your own organization, you can start any time - as soon as your next project ends, in fact.

## Nine Suggestions for Effective Software Postmortems

While it's impossible to give a single set of rules that work for every development organization, there are some things that (in my experience, at least) make software postmortems more effective. If you're starting (or refining) a practice of postmortems, you may find some useful advice here.

**1. Plan for Postmortems** - Your project plan should explicitly set aside time and space for the postmortem. To get the most value out of this activity, you need to take it seriously; people need to have time to think without being thrown immediately into the next project. The postmortem should be a scheduled activity, with time for a meeting of the team to discuss the lessons learned and time for someone (or some group) to write the postmortem report. If you can finagle a nice off-site conference room for this, so much the better; a little physical (and psychic) distance from the day-to-day workspace seems to help some people see more clearly what they spend their regular workday doing.

**2. Keep it Close in Time** - Don't let memories fade by scheduling the postmortem too long after the end of the project. Ship the software, have the celebration, and then roll right into the postmortem, rather than waiting for a convenient break in the action (which never comes, anyhow) a month or two later. On longer projects (anything over a couple of months), you

need to encourage people to keep notes towards an eventual postmortem, or hold a series of mini-postmortems at milestone dates, lest things get forgotten.

**3. Record the Project Details** - Part of the postmortem report needs to be a recital of the details of the project: how big it was, how long it took, what software was used, what the objectives were, and so on. This is not padding, but a way to help people looking for applicable experience in the future. If you build up a library of dozens of postmortems, a team about to embark on a 5-person, 6-month effort can use the project details to look for similar projects that the organization has tackled in the past.

**4. Involve Everyone** - There are two facets to this. First, different people will have different insights about the project, and you need to collect them all to really understand what worked and didn't. (In the worst of all possible worlds, the postmortem gets written exclusively by two or three top managers and is worthless except as a pat on the back for management). Second, getting everyone involved - by requiring their attendance at a meeting, if necessary - helps prevent the postmortem from degenerating into scapegoating. It's a lot harder to grouse that "Bob always screwed up the build" if Bob is sitting right across the table - and a lot easier to look for the more important lesson (in this case, likely something to do with the fragility of the build process).

**5. Get it in Writing** - The postmortem effort will usually kick off with a team meeting (unless the team is very small; fewer than 5 can probably conduct a postmortem entirely in e-mail), but it can't stay there. If it doesn't, you've got a bull session that will soon be forgotten, not a postmortem that will inform people in the future. The project manager needs to own the process of reducing the postmortem lessons to a written report, delegating this if necessary.

**6. Record Successes as Well as Failures** - Unless your project was an unmitigated failure you need to spend time recording what went right as well as what went wrong. It's easy for a postmortem to degenerate into a blame session, especially if the project went over budget or the team didn't manage to deliver all the promised features. But people need to hear positive messages as well as negative ones, and they need to hear what things are worth repeating as well as which things are worth avoiding in the future.

**7. It's Not for Punishment** - If people think you're using the postmortem to plan salary cuts or firings, or otherwise hang the blame for a failed or suboptimal project, they'll lie about what happened. It's as simple as that. If you want honest postmortems, management has to develop a reputation for listening openly to input and not punishing people for being honest. And the way to get that reputation is by not punishing people. (Of course, no one ever objects if the postmortem identifies some superior performers who get a bonus. Nobody ever said life was fair for managers either.) If possible, try to juggle the schedule so that postmortems and annual reviews don't come too close to each other.

**8. Create an Action Plan** - Step 1, meeting; step 2, written postmortem. Don't let it stop there! There should be a step 3: action plan. The written postmortem should make recommendations of how to continue things that worked, and how to fix things that didn't work. Remember, the idea is to *learn* from your successes and failures, not just to document them.

**9. Make it Available** - A software postmortem locked in a filing cabinet in the sub-basement does no one any good. Good organizations store the supply of postmortems somewhere that they're easily found. If you're the one responsible for producing one, you should consider sending out an e-mail at the end of the process saying something like, "The XYZ project postmortem is finished and available at \servershare. We recommend that future teams do a, b, and c and avoid d, e, and f. For more details, feel free to read our whole postmortem."

## A Competitive Advantage

Software postmortems, performed consistently, are a key part of bringing a development organization from chaos to smooth, repeatable functioning. In fact, if your development efforts are completely disorganized, postmortems can be a great way to start turning things around, because they will help you identify and keep the good parts while finding and throwing out the bad parts. If you're not already using this essential tool, it's never too late to start.

**About the Author**

Mike Gunderloy is the Senior Technology Partner for Adaptive Strategy, a Washington State consulting firm. You can read more of Mike's work at his Larkware Web site, or contact him at MikeG1@larkfarm.com.