

BETTER SOFTWARE

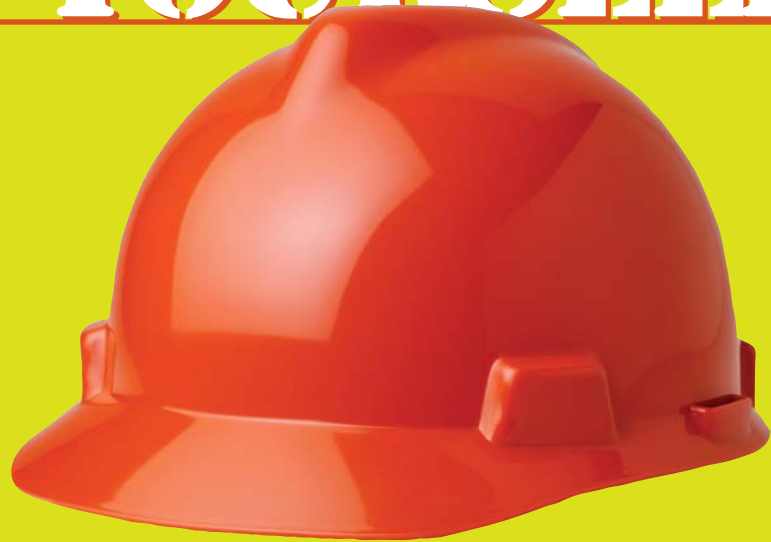
OUT WITH THE OLD
Put the squeeze on
legacy code
PAGE 30

IN WITH THE NEW
A lesson in scripting
languages
PAGE 24

The Print Companion to  **StickyMinds.com**

DO-IT-YOURSELF

A How-to Guide



for Fixing a

Failing Project

PAGE 18

DO-IT-YOURSELF

A How-to Guide



for Fixing a Failing Project

by Mike Cohn

When our company acquired a competitor's company, we were given less than a week to decide what to do with its largest project. Two colleagues and I had spent all day with the new team, and, after a late dinner with the team leads and managers, the three of us went back to our hotel. We found a quiet corner in the lobby and compared notes on what we'd learned through the long day. We'd learned that for the past year the project had consistently remained twelve months from shipping based on estimates at the time. We'd learned that the project had an incredible rate of unnecessary requirements change and that the product vision had changed repeatedly. The company also had selected a new architecture well beyond anything they had done before, and they'd chosen a new language along with it. We learned that the project had some very good people but that many team leads had been selected more for their closeness to the VP than for their skills.

Fast forward a few years and a few thousand miles to a different team and project. For unchangeable business reasons this team could only deliver significant new functionality in a single annual release that occurred in the same month each year. Time for the next release was rapidly approaching. The developers didn't want to start writing

any code until the business people had written the requirements; the business people didn't want to create a requirements document because they knew that anything that was inadvertently left out would be extremely difficult to get later. The project continued in this form of stalemate for months.

Each of these projects was in the process of failing. Projects fail all the time. They fail in a variety of ways and for a variety of reasons. I cannot possibly hope to present a solution for every possible cause of project failure. Instead, this article describes some specific steps you can take that will help in many cases. If it is your job to prevent projects from failing—or to fix projects that are on their way to failure, you will find this advice helpful.

Identify Failures: What Needs to Be Fixed?

The first thing I do when I encounter a failing project is to meet with as many people as I can and create a list of all the ways in which the project is failing or is considered to be failing—and these are not always synonymous. To create this list, you should talk to as many people as you can. Don't just talk to team leaders and key stakeholders, talk to people

involved at all levels and across all groups involved in the project. Some people are more comfortable speaking as part of a group; others will prefer speaking to you one-on-one. If you know individual preferences, try to accommodate them. If not, take a guess and offer individuals more than one opportunity to speak with you.

In general and within reason you should treat as confidential any specific information you receive. Occasionally you'll be told something that needs to be shared with others. Always get permission before doing so. To get that permission, you may need to agree to leave out some details. But if you value the trust of those with whom you spoke (or will speak with on the next project), only pass along that information for which you have permission.

I try to keep these discussions brief. One-on-one meetings usually last no more than thirty minutes. If I need to talk with a lot of people, I may schedule the meetings at fifteen-minute intervals. Group discussions need to be longer. Naturally, these meetings need to happen in a conference room where others cannot see or hear what is being discussed.

At one dysfunctional company, a software architect actually had run a microphone from the ceiling of the company's executive conference room to his office. His excuse was that he couldn't attend every meeting, yet he often needed to listen in on the meetings. Naturally he was fired for listening to meetings to which he hadn't been invited. Another company had windows seven feet up the wall between the conference room and a main corridor. Some individuals discovered that if they stood far enough down the hall, they could see through those windows and see most of what was being projected in the conference room. You may want to avoid rooms like these. Even better, you may want to avoid working with people like this.

Treat everyone you talk to with respect. You're not on a witch-hunt, looking for someone to blame for the failure of the project. You are looking for people who not only can help you understand why the project is failing but also help you correct the situation. You are going to need allies; treat each person you speak with as potentially your greatest

ally. As you interview people, ask open-ended questions rather than questions that can be answered with a simple yes or no. Your goal is to get people talking about the project. It's important to avoid discussing all the emotional baggage and politics and to focus more on the concrete problems. Some of the questions I ask are shown in the sidebar.

Nailing Down the Problem

- In what ways do you consider the project successful?
- In what ways do you consider it unsuccessful?
- (For each identified way in which the project is unsuccessful): What do you think caused that?
- What could you have done to make the project more successful?
- What could your group have done to make the project more successful?
- Tell me about a time when you were frustrated on this project.
- Tell me what you would have done differently on this project.
- Tell me about the biggest barrier to success.
- Tell me about the biggest misunderstanding between developers and managers (or programmers and testers, etc.).
- If you could fix one problem, what would you fix?
- What are the greatest strengths and weaknesses of the current team?
- What else should I know?
- Can you think of anyone with whom I should be sure to speak?

Be aware that some of those you interview will tell you what they think you want to hear. They will be more concerned with making a good impression on you than with giving you an honest assessment of the state of the project. On one project I assessed, the product manager knew I was very strongly focused on the testing and quality efforts that had gone into developing the nearly finished product. He kept stressing that "quality is baked

in." I let the odd phrase go the first time or two he said it. When I later pressed him for what it meant, it turned out that "quality is baked in" was his euphemism for "we have no testing at all, so programmers are expected to write high-quality code." Nice goal, but they weren't achieving it.

Identify Causes: Why Does It Need to Be Fixed?

Once you've assembled a preliminary list of ways in which people view the project as having failed, you need to look for potential causes. I've tried doing fancy, involved analyses of the failures as a way of finding common root causes. I've tried doing affinity groupings of the causes. None of this is really necessary, though. The causes are never that hard to find, and the fancy analysis ends up pointing to what was probably apparent midway through the interviews. If you've been managing software projects for awhile, you've probably seen most things that can go wrong on a project, so identifying the causes of failure won't be difficult.

One lightweight technique I have found useful is called "Five Whys." This helps our thinking get to the root cause—beyond the apparent cause of failure. For example, suppose you have discovered that the project is struggling because the architecture is needlessly complex and built entirely on new technologies.

1. Why? Because they were chosen by an architect who wanted to learn something new.
2. Why? Because she'd like to add those skills to her résumé.
3. Why? Because she may want to quit and leave the company.
4. Why? Because she feels she's grown stagnant in her job.
5. Why? Because each application she architects is similar to the last one.

There's no magic to the number five, but you want to ask why enough times that the answers are pointing closer to root causes.

Look for Solutions: How Can It Be Fixed?

Having identified some of the reasons that the project is failing, you need to identify solutions and work to correct the problems. Keep in mind that in most cases, projects fail because of the people involved—not the technology. We are rarely so pushing the envelope of technology that the project simply cannot be done. On the other hand, we often assemble dysfunctional teams of individuals who don't want to work together, who don't know how to work together, or who have an agenda other than project success. When you encounter a failing project, look first to the people involved. When you figure out who has a hidden agenda, who doesn't like working with whom, who isn't sharing information, and who is trying to make a political statement through the project, you often will discover how to treat much of what ails the project.

Enlist Help: Who Will Join the Construction Crew?

You cannot right a failing project on your own. Since you will need help from the team, it's important to involve them in identifying the problems and solutions. You have to be careful with this and remember that the problems the project is facing are quite likely—at least partially—the fault of those you are asking for solutions. Ask, but listen carefully to their answers. They may be honest with you; they may not. There will undoubtedly be some team members who care passionately about the success of the project. Find them and engage them in solving the problems.

However, be careful that you do not pass the responsibility to these individuals. If you've signed up to right the failing project, own the task completely. Project members may be looking for someone who will step up and take on that responsibility. Engage others to the extent that they'll help. If the project is turned around, those individuals deserve all the credit you can give them but the ultimate responsibility for success or failure resides with you.

Go Back to the Beginning: Why Was This Project Started in the First Place?

In the movie *The Princess Bride*, three thieves are hired to kidnap a princess. The leader among them instructs the others that, if anything goes wrong, they are to go “back to the beginning,” where they will presumably regroup and try again. This advice was appropriate in ancient Florin, the home of the princess, and it is appropriate advice for struggling projects today.

If a project is failing, you need to revisit the beginning of the project and understand why the project was initially undertaken. If you are an outsider coming into the project for the first time, this is relatively easy. No one will look oddly at you when you ask questions with seemingly obvious answers, such as “What was the project intended to accomplish?” and “When was the project originally supposed to be completed?” If you've been involved in the project for awhile, you may be able to dig up notes about why the project was initiated. However, even if you think you have determined why the project was initiated, it is often useful to ask key stakeholders why *they* think it was started. Often their answers will differ from your notes or memory. This may be a source of some of the problems the project is experiencing.

Once you've identified the initial objectives of the project, your task is not to aim the project at those objectives. Your job is to find a project outcome that is satisfactory to all project stake-

holders. At first it is very likely that

holders. At first it is very likely that stakeholders will not be interested in negotiating this with you. After all, they've already gone through this once at the start of the project and now are being told they aren't going to get what they'd negotiated for at the beginning. To find and negotiate a project outcome that is acceptable to all stakeholders, you need to talk with the key stakeholders, the project sponsor, and others to determine their minimum set of goals for the project. Once you know each person's or group's minimum conditions of satisfaction, you can begin to find a way to meet them within the technical constraints of the project. For most projects, the *conditions of satisfaction* consist of the three dimensions of the infamous iron triangle: cost, schedule, and scope. In order to succeed you must find some point within that space that is acceptable to all project stakeholders. This defines a new set of goals for the project. These goals will not be as ambitious as those with which the project initially was undertaken but they will be achievable given the current situation.

Set Expectations: What Is the Crew Expected to Build?

Once you have negotiated with key project stakeholders and have settled upon a revised set of goals for the project, you need to make sure that everyone in the extended project community understands the new expectations. On past projects I've conveyed new expectations through company newsletters, posters hanging in the hallways, intranet sites, lunchtime demos open to everyone, and many one-on-one meetings with individuals in other departments. Do whatever you need to do, but make sure that everyone understands the newly agreed-upon goals of the project. This isn't about lowering expectations among stakeholders; rather, it

Keep in mind that in most cases, projects fail because of the people involved—not the technology.

holders. At first it is very likely that stakeholders will not be interested in negotiating this with you. After all, they've already gone through this once at the start of the project and now are being told they aren't going to get what they'd

is about making sure everyone knows what the new plan calls for and in what ways the project has been scaled back. People need that information so they know what they can ask for and what they can expect of the project and the team building it.

Act: Start Making Improvements Now

With new goals and expectations in place, immediately start moving toward the project's goals. Do not wait until you know everything or until you've assessed every possible cause for every possible failure. This project is failing, and it's probably getting worse day by day. Act now.

By using short iterations and asking the team to make incremental enhancements to the product, we get the dual benefits of rapid feedback and visibility into our progress. If things start going astray again, you'll know it in no more than two weeks.

Don't be afraid. You are in a wonderful position—almost anything you do will help.

Because so many problems are related to the people involved, you may very likely need to fire someone or at least transfer them off the project. If this is the case, don't hesitate. Cutting someone loose who has key knowledge rarely (if ever) hurts as badly or as long as you think. Do it just like you'd pull a bandage off a wound—fast. I've never let a key person go and not had someone else on the team step into the role faster than I could have expected.

Get Solid: Your Foundation Must Come First

Years ago, after managing the completion of a particularly large and risky version 1.0 development effort, I commented that I never wanted to do that again. A typical 1.0 release has many moving parts, a huge amount of uncertainty, and many concurrent efforts that all need to come together in time for release. On the other hand, an upgrade, such as a version 1.1, is smaller and is much easier because it is building on the solid foundation of the previous release. Yes, 1.0 projects are fun because the field of possibilities is wide open but that can be outweighed by all the concurrent complexity.

However, immediately after vowing that I was done with 1.0 releases, I realized what I really needed was to quickly and consistently get any new project to the point of having a solid foundation. If the foundation of a project is unstable and changing, it is extremely difficult to add new capabilities to the system at the same time you are addressing the stability problems. If a project is failing,

immediately stop adding new features. Put all of your focus on solidifying the core set of functionality that thus far has been shakily built. Fix all the bugs that would prevent you from shipping. Make sure the application can be easily deployed or installed. Only add new functionality once all the partially completed features have been thoroughly debugged and shaken out.

Work Iteratively and Incrementally: Add On in Small Steps

Once the application is stable, ask the team to figure out what new capabilities can be added in the next two weeks. The new capabilities will need to be tested, and the application will need to remain stable. Having achieved a solid foundation, we don't want to give it up. Don't look any further out than two weeks. The project was failing before and part of getting it back on track will be keeping the time horizons short. By using short iterations and asking the team to make incremental enhancements to the product, we get the dual benefits of rapid feedback and visibility into our progress. If things start going astray again, you'll know it in no more than two weeks.

A key advantage to iterations is that they allow the team to establish a track record of successes. Instead of feeling as

though the team is failing at a long project, team members feel that they are succeeding at many small projects. This will have a very positive effect on morale, which will have a positive effect on the project. Low morale is a common factor on failing projects. On one struggling project the VP of the development team commented that his team was not ready for a "home run project." So instead we

put together a plan that allowed the team to string together a series of "singles" that eventually added up to a successful overall project.

Learn

The only true failure is one from which we don't learn. If you are on a failing project, learn from it. The problems you encounter on a project today are ones you're likely to encounter on future projects. Learn from your current project so that you can correct or prevent similar problems in the future. **{end}**

Mike Cohn is the founder of Mountain Goat Software, a process and project management consultancy that specializes in helping companies adopt and improve their use of Agile processes and techniques. The author of Agile Estimating and Planning and User Stories Applied: For Agile Software Development, Mike is a founding member of the



He is a technical editor for Better Software magazine, a regular columnist for the magazine and StickyMinds.com, and a frequent presenter at the STAR conferences and the Better Software Conference and EXPO. He can be reached at mike@mountaingoatsoftware.com.