4

UNBRIDLED ENTHUSIASM

After reading the first three chapters of this book, you might have gotten the impression that I'm one of those leads who likes to keep my team members' noses to the grindstone. I've certainly put enough emphasis on staying focused to justify such a suspicion. But my goal is not to extract the maximum amount of work out of each team member. My goal is to put out a great product that the development team has an exhilarating time putting together.

Have you ever worked on a project that sizzled with enthusiasm? If you haven't, have you at least had single days on which you felt great as you left the office? Think back to such a day. Was it filled with meetings, reports, interviews, and e-mail exchanges, or did you spend the day working uninterrupted, creating great new designs and coding hot new

features? We both know the answer. I've never met a programmer who got excited about having written yet another report or having attended yet another meeting.

One of my driving goals as a lead is to create an atmosphere in which the development team can have a blast as they create a product they're proud of. I do that in part by working hard to ensure that programmers don't have to write unnecessary reports, or attend unnecessary meetings, or fuss with schedules, or do anything else that pulls them away from creating new and exciting features for the product. In this chapter, I'll tell you why I think such processes are harmful—as they're commonly practiced—and how you can replace overblown corporate processes that suck the life out of projects with simpler, more effective practices.

THE UNREAD REPORT

Right after I got back from a business trip one time, my lead called me into his office and quizzed me on all the details. When we'd finished talking, he asked me to write up a detailed trip report describing everything I'd just told him. That seemed like a waste of time to me, so I asked if the report was really necessary. He assured me that it was, so I spent the better part of an afternoon writing that report instead of working on features.

Later that month, my lead asked me a question I had fully answered in the trip report. I could understand his not remembering the details, but I was puzzled that he hadn't referred to the report for the answer, so I asked him if he'd read the report. He admitted that he hadn't—he had filed it as soon as I'd given it to him.

"Why did you have me write that report if you never intended to read it?" I was irked.

He gave me a surprised look and said, "*Everybody* has to write trip reports. It's policy. . ."

Any reason he could have offered for having me write the report would have been better than that poor rationale. If he'd had me write the report because studies show that writing something down cements the knowledge better in your head, I could have understood that. If he'd

told me he intended to pass the report on to his own lead or to other teams who could benefit from its contents, I could have understood that too. But having me write a report when he knew he was going to file it away unread was absurd. This was an excellent example of somebody's following a guideline as if it were an ironclad rule, and because of it we wound up doing something stupid. Any time corporate policy has somebody writing a report that nobody will read, corporate policy is wrong—unless the ultimate purpose *is* to cement the knowledge better in the writer's head. But if that's the case, do all trips need such cementing? My lead was following business-as-usual, not ruthlessly eliminating all obstacles to product improvement.

The Mystery of the End-Cut Pot Roast

One problem with any process you put in place is that over time people tend to forget the original reason you set up the process, and continue to observe it even though it may be outdated. Somewhere I read a story that succinctly illustrates this point:

> A young boy once asked his mother why she'd cut the ends off a pot roast before she put the roast into the oven. "Well," she said, "because that's what my mother taught me to do." But the question got her to wondering, so she asked her own mother for the reason behind lopping off the ends. "To tell you the truth, I don't know," answered her mother. "I've always done it because that's what I saw your grandmother do." A real mystery. So the boy's mother put the question to her grandmother. Grandma's reply: "Back then, I had a small roasting pan—roasts wouldn't fit into the pan unless I cut off the ends."

Like that boy's mother, the lead who had me write a report he didn't intend to read was following a procedure without understanding its original purpose. Since having me write a report that he knew nobody would read was so clearly a questionable practice, he should have (or I should have) tracked down the idea behind such reports. We might have discovered that not all business trips call for trip reports, and in fact I later found out that they don't.

Some trip reports are definitely worthwhile, particularly the reports people write as soon as they get back from trade shows such as COMDEX. Those reports are typically chock-full of observations and insights about the state of the industry, about what the competition is up to and how the crowds responded to the competition's booths and demonstrations, and about how those same crowds responded to their own company's booths and demonstrations. After a show like COMDEX, trip reports flood e-mail networks. Great stuff.

But not all trip reports provide that kind of value. Just because you flew to Kokomo, Indiana, to isolate a bug at a site there that you couldn't reproduce in your office doesn't mean it's worthwhile to write a trip report when you get back. Would you write a trip report if you had walked down the hall to isolate a problem that showed up only on a tester's machine? I hope not. Would you write a trip report if you had driven across town to isolate a problem that affected a local company? At Microsoft, you wouldn't. You wouldn't even write a trip report if the offsite location were on the opposite side of the state and it had taken you four hours to drive there. But if you took a 20-minute puddle-jumper flight, most managers would ask you to write a trip report. Why? I don't believe the reason has anything to do with whether the report is actually needed. The manager has to fill out special paperwork to authorize the expenditure for the plane ticket. The trip is therefore special, the manager reasons, and requires a report.

When I said that leads should ruthlessly eliminate unnecessary work, the superfluous trip report was the kind of thing I was talking about. Just as I don't call meetings unless the value they provide offsets the interruption they cause, I never ask for a report unless there is a compelling reason for one. I'd much rather have people working on the product and interacting with other team members than working on a report I don't really need. My view is, any time I'm about to interrupt a team member's work, I'd better have a darn good reason—to heck with business-as-usual.

I rarely ask for reports because I don't believe they're worth the disruption they cause. But when I feel I must have a report, my preference is to get an oral report because it takes much less time—5 minutes of interactive communication vs. 30 minutes—or more—of writing.

If you ask some people to write a report, their eyes glaze over. If you drop by three hours later, you're liable to find the person frozen in front of the word processor, having written only two paragraphs. For some people, writing a report ranks right up there with speaking to a full auditorium—it paralyzes them.

Another problem with written reports—if you don't explain exactly what you want—is that people go on at length about stuff you have no interest in. And many people get bogged down in bad prose because they think the text has to "sound like" a report. Instead of writing "the bug showed up only when the floppy drive was empty," such people think they need to say something like "the error occurred only in those instances in which the drive mechanism contained no media." It's harder and takes longer to write in that unnatural style. It's also harder and takes longer to read that kind of writing. Besides, reports written that way are about as exciting as the test pattern on your television set.

When I do ask people to write reports, I ask them to keep the reports as short as possible and to keep the writing informal—to avoid report-speak. I don't demand that they write reports in this style, but I do encourage it. For people who get paralyzed at the prospect of writing a report, these two requests help make it a less painful, less time-consuming affair.

"Keep it as short as possible?" they'll say. "No problem!"

If a team member wants to expand on a few ideas in a written report, that's all right with me too. Some people prefer writing reports over presenting them orally, particularly if they're trying to persuade the reader to act on the contents. A written report can enable both the writer and me to carefully consider exactly what's reported and the line of action the reporter thinks we should take. My goal overall, though, is to get the information I need with the least amount of pain and interruption to the writer.

Written reports, like meetings, interrupt the writer's work. Don't ask for them unless they provide real value—enough to offset the cost of the interruption they cause.

Be sure that every report you ask for is worth the time it takes for the writer to prepare it.

THE GOOD, THE BAD, AND THE SHELVED

One kind of report I have found to be invaluable when it's done well is the project postmortem report. I'm talking about the project analysis some teams write up shortly after a release. A postmortem report answers the question "What can we learn from the project we just finished?" What went right (let's keep doing that) and what went wrong (how can we prevent those problems from recurring in the upcoming project?)? Postmortem reports are crucial because they force the team members to actively consider how they can improve the development process.

I love reading project postmortems because they contain so much good information. But all too often, I'll read a postmortem report that contains important insights but is effectively worthless because the writer hasn't taken the next step: describing exactly what's going to be done about those insights. In one case, I read a pile of postmortems a product team had written over the years. Each report started with "We should have included more debugging code at the start of the project," followed shortly by "We also should have fixed bugs up front instead of allowing them to collect until the end of the project." Both excellent observations. Unfortunately, the same insights appeared in the postmortems for release after release of that product. Apparently, nobody was acting on the team's hard-won knowledge.

If you do postmortem reports—and I advise you to—be sure to include a detailed attack plan that describes how you plan to take care of each known problem so that it doesn't come up again in the next project cycle. I'm sure the team whose pile of postmortems I read never included more debug code at the start of each project or changed their bug-fixing habits. Once the postmortem report was written, they stored it away on some dusty shelf, never to be read again nor acted upon.

Some postmortem reports I've read have contained attack plans, but the plans were ineffective because they weren't specific enough—they had no teeth. Suppose a postmortem report contained this problem and attack plan:

```
Problem: external beta sites felt their bug reports fell
on deaf ears, mainly because the bugs they reported would
continue to appear in beta release after beta release of
our Mandelbrot package. These bugs were slipping through
the cracks because we had no systematic approach for
tracking external bug reports. In the future, we must
try to track external bug reports more carefully.
```

In many cases, that was all the postmortem report would say on the matter. From the occasional team who took the extra step and developed a more specific plan of attack, the plan would look like this:

```
Solution: the Plotting Division needs to implement a
better method for tracking bugs reported by external
beta sites.
```

From the rare team who developed a detailed attack plan, the plan would look like this:

```
Solution: to prevent our losing track of bugs reported
by external beta sites--a problem that affects not
only our Mandelbrot project, but also the Biorhythm and
Morph projects--Hubie Dobson has agreed to review three
well-respected bug-tracking systems (Bug Control, Pro-
grammer's Database, and FixIt!) and recommend one tool
for division-wide use. Hubie will make his recommendation
within the next two weeks (by June 12). We will use the
system Hubie recommends for our Mandelbrot project as
an initial test case, and we will maintain a list of any
tracking problems we encounter.
```

Which of the three reports do you suppose would be most likely to produce change in the development process? The one that states little more than the problem and an intention to do better, the one with a simple attack plan, or the one with the detailed attack plan? Is there any question that the final plan would be the most effective?

The final plan will be the most effective because it tells exactly what the solution will be, who will be responsible, when the deadline is, and where the plan will be applied. The plan also provides for evaluation. Who is accountable in the first or second example? It's easy enough to say, as the second report does, that the Plotting Division should implement a new bug-tracking system, but who is that? And by when

should the Plotting Division implement a new method for tracking external bug reports? On which project will they try it out? Will they report the results of the trial? Without such details, attack plans are toothless.

The postmortem report should also describe development practices you found to be worthwhile in the course of the project. The report might say that once the team began using program assertions and debug code, hidden bugs began popping out everywhere, even in code thought to be bug-free. The report might note that the practice of stepping through code in the debugger the moment it is written was at first a bit tedious, but that once programmers got used to the practice the number of bugs found by testers dropped considerably and without hurting the schedule. Or the report might observe that having detailed project goals really helped the team stay focused. These are excellent points. But the report shouldn't stop there.

For each such observation, the postmortem report should indicate how the observation will be exploited in the future. It's not enough that a team discover what works well; they must use that knowledge to its full advantage. If only some team members were habitually stepping through their code the moment they wrote it, for instance, the report might describe the steps that will ensure that all team members will begin to use that bug-finding technique.

Finally, the postmortem should describe as part of its attack plan some method for making the findings in the report available to other teams. This part of the attack plan could be as simple as saying "we will provide copies of this report to the following leads by such and such a date." That plan tells what, who, and when.

Researching and writing postmortem reports takes time and adds yet more process to development—which I oppose on principle—but the educational benefits of postmortem reports compensate for the time deficit, with one caveat: you must act on your findings. If postmortem reports end up on the shelf, never to be read again, they haven't been worth doing.

By the way, you don't need to wait for the end of a full release cycle to write a postmortem report. Every time you run into a problem or discover a better way of doing things, jot your findings down in an ongoing document and take immediate action to exploit your new knowledge. Why wait until you've shipped to gain the benefits?

Use postmortem reports to improve your development process. To make a report effective, describe exactly how your team plans to fix known problems and how it plans to exploit the effective development practices it has discovered.

MEETINGS TO MISS

In Chapter 1, I talked about why I think weekly status meetings are unnecessary if you're already collecting status reports of some kind. But status meetings are just one form of the recurrent meeting, a kind of meeting I routinely try to eradicate. By "recurrent meeting" I mean any regularly scheduled gathering. You arrive at work, week in and week out, and you think, "It's Tuesday. I'd better not forget that regular three o'clock meeting."

I rarely hold meetings because they can be so disruptive to the smooth flow of work, and I particularly dislike recurring meetings because the motivation for holding such meetings usually isn't clear. Are you meeting because you need to, or because it's three o'clock Tuesday? Some people would argue that weekly status meetings are indispensable. I've gone without them for years. It's not the meeting that's important; it's the information you would get by attending such a meeting. If you can get (or pass on) the status information more efficiently without a meeting, why not take that approach? As I said in Chapter 1, my teams' little "I've just finished. . ." e-mail notes have worked fine for me.

Does it ever make sense to hold meetings? Of course. There are times when meetings do more good than harm. In particular, meetings can be valuable when

- one individual must pass information on to a large number of other people and a meeting is the most efficient way to do that
- people must be able to actively respond to information—to ask questions or to interact with other attendees
- value will be realized from seeing or experiencing something—a product demonstration, for instance

- a matter too delicate for a memo or an e-mail—a reorganization or layoffs, for example—must be discussed

A meeting doesn't need to meet all of these criteria in order to be worthwhile—any one of them will do, provided there isn't a better alternative. In the days of stone tablets and parchment scrolls, it made sense to hold meetings to pass out information—that was the most efficient method. You gathered the masses and did the "Hear ye, hear ye" bit. But today, with photocopiers, electronic mail, and electronic bulletin boards, you can pass out information with much greater efficiency, and without interrupting people's work. Of course, you should use common sense. If you have something important to say, holding a meeting to say it underscores the importance and guarantees that everyone will hear the message. And if you're a dynamic speaker, you can rally attendees to action.

Before you call any meeting, take a minute to ask these questions:

Will the results of this meeting be important enough to warrant interrupting the work of the people who will have to attend it?

Is there a less disruptive way I can get the results I'd get from holding the meeting?

Team Spirit

I've heard some leads say that their weekly status meetings are important because the meetings get the entire team into one room where they can see each other face-to-face. The practice builds team spirit, they say. I've heard of leads who hold status meetings *primarily* so that the team members can get together. The objective is a good one, but in my experience, the status meeting is just about the worst venue for promoting team spirit. If your status meetings are like the ones I've described, in which the focus is on what everybody *didn't* get done that week, such status meetings won't really help build team spirit.

If your team members don't tend to meet often in spontaneous hall gatherings and brainstorming sessions, maybe you do need to create opportunities for mingling. If that's the goal, go out for group lunches, or schedule some other *positive* activity together. Forget using those punishing status meetings for that purpose.

When you ask these questions about a prospective design meeting, you can see that it probably does make sense to interrupt the team's work, or at least the work of part of the team. The work done at a design meeting improves the product. It directly influences how the product will be built. The team may be getting pulled from their individual tasks, but they're still focused on the product, not on housekeeping. Design meetings also encourage rapid-fire debate over the trade-offs among various designs. You can't easily or efficiently brainstorm that way over e-mail.

I would be suspicious, though, of any design meeting that was regularly held at three o'clock every Tuesday. Unless you have scheduled a series of specific design tasks—design the memory manager this Tuesday, the file I/O next Tuesday, the internal document structure the Tuesday after that, and so on—I doubt that a regular design meeting makes sense. I'd imagine that a recurring design meeting would always open with the question "Have any new design issues cropped up this week?" And I'd like to assume that if such issues had cropped up, they wouldn't have been kept quiet until the next Tuesday. Team members should bring up new design issues immediately, and if a gathering

Good Meeting Times

If you must have a meeting, at least schedule it so that it doesn't break up an otherwise large chunk of time. Don't schedule your one-hour meeting at 10:00 A.M. or 3:00 P.M. so that it chops the morning or the afternoon into two-hour pieces. Schedule the meeting at the beginning or the end of the day, or just before or right after lunch. In other words, schedule your meetings next to standard break times to maximize the size of uninterrupted time blocks.

Another approach is to schedule all your weekly meetings in one continuous block—say, on Monday morning or Friday afternoon. Monday morning and Friday afternoon are notoriously the least productive times of the work week anyway. Put all your meetings into one of those blocks of time, and keep them out of the better, more productive, parts of the week.

seems necessary to work out problems, you can call an ad hoc meeting. Reserving a time each week "just in case" there are problems seems to me to be more disruptive than helpful.

Beware of recurrent meetings. Make sure they're worth the disruption they cause.

Effective Meetings

As much as I dislike holding meetings, or attending other people's meetings, I recognize that meetings are sometimes necessary. And as for any unpleasant task I believe is necessary, I ask the benefits-drawbacks question: How can I get the benefits of this meeting without the drawbacks?

The benefits of meetings are the results you get out of those meetings, and the chief drawback is that so many meetings are a waste of time because there aren't any results—often because the purpose of a meeting was never clear to the participants. You can hold a far more effective meeting if you first decide exactly what you want to accomplish at the meeting and then come up with a plan to get those results by meeting's end. It's the old "set your goals and create an attack plan" scenario.

Once you've decided that a meeting is necessary, be sure to ask this question before you send out the invitations:

> *What do I expect to achieve at this meeting, and how can I be sure to achieve it?*

If you ask this question before each meeting, you have a much better chance of not wasting everybody's time with random presentation and discussion.

Remember that hypothetical house-moving lead I talked about in Chapter 3, the one who didn't drive ahead to check out the route before the house hit the road? The driver ran into overpasses, hills, and roadwork because the lead didn't take the steps beforehand that would have ensured they could get the results they wanted.

When you ask yourself what you expect to achieve at a meeting, you force yourself to look ahead for possible obstacles to what you hope to achieve and to take steps to avoid them. If you have a clear idea of what you want to achieve and of what is necessary to achieve it, you can see that all key decision makers attend and that they bring whatever you'll need for the results you want. How many meetings have been for naught simply because a key decision maker couldn't attend, or because somebody didn't know to bring a vital piece of information?

Still, despite your best efforts, there will be times when you won't have all the information you need to make a final decision. When that happens, the meeting coordinator will often say something like "George, find out if your two-week guesstimate for the Anagram feature is accurate, and we'll meet again to decide whether to include it in this release."

Leads who use that approach waste people's time. Everybody met, yet nothing was decided. If your goal is to get a decision, *make sure you get a decision,* even if it's a conditional one. It's far better to end a meeting with, "Assuming that George's guesstimate for the Anagram feature is accurate, does everybody agree that this feature is strategic enough to delay our WordSmasher ship date?"

With that question, you may find that nobody thinks the feature is strategic enough to jeopardize the ship date. Or maybe that they think the feature is so important it doesn't matter how it affects the ship date. But more often than not, you can get a conditional decision: "Let's do it, provided the Anagram feature won't delay the ship date by more than two weeks."

Such a decision may not be as concrete as a definite Yea or Nay, but it's infinitely preferable to postponing the resolution of the issue and calling yet another meeting. If your goal is to get a decision, *get one.* If your goal is to achieve something else, make sure you achieve that.

Before calling any meeting, be sure you know exactly what you want to achieve and what you need to achieve it. Then make sure you do *achieve it.*

A Metric for Meetings

I've hammered on the idea of getting decisions at your meetings because almost every worthwhile meeting ends with a decision of some kind. If you hold a meeting that doesn't end in a decision, that meeting has probably been a waste of time.

Think about a status meeting. Is it held to reach a decision? No, its purpose is to pass information around. What about a design meeting? Yes, you're deciding on a design for the product. The meeting may be a brainstorming session, but the goal is to leave the meeting with a design, or at least a design approach, that everybody agrees on.

What about an upper management project review meeting? Can it end with a decision? That depends. I've seen two types. In the first, the lead describes the course of the project over the last year, touching on major highlights, and finishes by reviewing the current schedule and expressing some level of confidence about the projected dates. In the second type of project review meeting, the lead doesn't dawdle over the past but instead describes in detail where she is taking the project, why she has chosen that direction, what her detailed attack plan is, what the alternative approaches that she rejected were and why she rejected them, how her plan fits into the long-term direction of the company, and finally how upper management can help—all she needs is their support.

The first type of review meeting is just dumping information on upper management, whereas the second type is a presentation to persuade upper management to back the lead's plan, to get them to *decide* to support her plan. Which type of presentation do you think is better for the company, getting upper management to focus on the past or getting them to commit to a course for the future?

Some gatherings, such as pep rallies and the annual company meeting, don't result in decisions, but those meetings have a different purpose, and more important, they aren't held each week, or even each month.

NO FOLLOW-UP WORK

Another drawback to meetings is that they tend to create follow-up tasks for the people who attend. Sometimes you can't do without a follow-up task—you need to have George figure out exactly how long it would take to implement that Anagram feature in WordSmasher—but a lot of follow-up work is busywork. Remember the lead who required team members to send follow-up e-mail repeating what they'd said at the meeting? Follow-up work is just that much more work that pulls the development team away from the tasks they were doing before the meeting started.

Whenever you're wrapping up a meeting, restating the decisions you've reached and recounting the action items for various attendees, be sure to consider whether each follow-up action item is essential. I know several leads who seem to feel that everybody must have picked up at least one action item by the end of a meeting. Such a lead will circle the table mentioning what each person is to do—until he hits upon somebody with no follow-up task. He'll stop for a minute, scratch his head, and manufacture a task: "George, why don't you. . ."

If you've fallen into this tendency, try a different approach. As you circle the table, reevaluate each action item to determine whether it's really worth spending time on. A typical dialogue might go like this:

"Next is George. You were going to get an estimate for the Anagram feature. Realistically, do you think there's any possibility of doing that feature without affecting the ship date by more than two weeks?"

"Actually, I've thought of some additional issues in the last 20 minutes," George says. "I now think the feature will take at least three weeks to implement."

"OK. The Anagram feature isn't important enough to jeopardize our ship date, so let's postpone the feature until the 3.1 release. Everybody agree? Good. George, you have no action items. Now, Rebecca, you were going to. . ."

I think you'll be surprised at how many follow-up tasks don't seem nearly as important by the end of the meeting as they did earlier, in the middle of an intense discussion.

Try to eliminate unnecessary follow-up work.

Wriggling Out Of Work

Doesn't circling the table looking for ways to eliminate work create a harmful negative feedback loop, one that encourages people to misinform you so that they'll get out of some work? Did George really find another week's worth of work when he looked again at doing the Anagram feature, or did he fabricate that week's worth of work as a means of getting the feature killed—and reducing his work load?

In any organization, you're going to find some individuals who have no qualms about lying to ease their burdens. That's life. But I believe the vast majority of people are sincere and don't play such games. You quickly find out who the other few are.

Besides, I doubt that a team would so easily kill a feature (or an action item) if they felt it was important. That Anagram feature would not have been dropped if the others at the meeting had felt it was strategic to the current release.

BREAK OUT THE JACKHAMMER

If you want to keep the excitement level in your team high, enable them to work on the product without constantly being pulled off their work to write reports, attend meetings, and deal with other processes that won't help to improve the product. Unfortunately, the corporate tendency is to call meetings for every little thing and to ask for reports as a knee-jerk reaction: "I'm busy right now; send me a report."

You might think that a little speed bump in the road would be just a small obstacle, but imagine how such a bump would affect a race car going at a high speed—it could break the car apart. The development team is like that race car, raring to go, and just as they start to pick up

speed, WHUMP, they hit a speed bump in the form of a meeting, a report, or some other corporate process. Sometimes it's worse. The lead who regularly asked for status reports, called status meetings, and required follow-up reports was a one-man speed-bump builder. WHUMP, WHUMP, WHUMP. . .

You may not have control over all the speed bumps that slow your team, but you certainly have control over many of them. Retire that truck full of blacktop and break out your jackhammer. Do some real damage to those bumps.

HIGHLIGHTS

- Try to limit the number of reports you ask other team members to write. Be sure that every report you ask for will provide more value to you or the company than would be lost by interrupting the writer's work.
- Postmortem reports are invaluable when you do them correctly. Unless your postmortem reports explain exactly how you intend to fix known problems or exploit known improvements, though, the reports probably aren't worth doing.
- Before you call a meeting, be sure the results you think you'll get from that meeting are worth the disruption to the work of the people who would have to attend. Be particularly wary of any regular meeting. Regular, standing meetings often aren't worth the time to walk to them, much less attend.
- If you must hold a meeting, minimize the amount of interruption it will cause. Schedule the meeting so that it won't break up an otherwise large block of time.
- Whenever you call a meeting, be sure you know ahead of time exactly what you're trying to accomplish, and then make sure you do accomplish it. Remember also that conditional decisions are better than no decisions.