,ch27.20505 Page 365 Thursday, February 12, 2009 3:10 PM

CHAPTER TWENTY-SEVEN

Speed Versus Quality

Why Do We Need to Choose?

Johanna Rothman

DURING A PROJECT WHEN I WAS THE PROJECT MANAGER, I HAD THIS CONVERSATION WITH A SENIOR manager at a company:

> *Big Cheese:* "Stop those code reviews. They're slowing down the project."
> *Johanna:* "But then we won't know where the bugs are. We need the code reviews."
> *Big Cheese:* "Stop them or I'll fire you."
> *Johanna:* "You'll fire me for doing the right thing?"
> *Big Cheese:* "In this case, the right thing is to finish the project as fast as possible. Stop those code reviews."

Pretty strange conversation, eh?

I wish I could say this was an isolated incident. But even though this particular conversation occurred almost 20 years ago, it still happens every day someplace. This is the story of a team that refused to buckle under management pressure to do it fast. The team knew that they if they did a great job, they *could* do it fast, by doing it right.

How Did We Get Here?

If we turn on the Way-Back machine to the beginning of the project, Big Cheese brought the project team together and said, “I have great news. I sold a new version of ProcessControlApp. But it has to do these three things faster, and have these five new features. And, we need it in six months.”

The project team of six developers got together and discussed the problem. They’d worked as a team for several years. The newest team member had 18 months of experience working on the product with the team, and the most senior developer had initiated the product four years before. They knew each other and how to work together.

ProcessControlApp had no GUI because it was integrated into a manufacturing line. All of the access into and out of the application was through the command line or through an API. ProcessControlApp did a visual inspection of the material on the line, made some decisions about the material, and commanded the line based on its decision. So although there was no GUI, ProcessControlApp was a highly complex application, where performance (how fast could the software make the decision?), reliability (how much uptime did the software have?), and accuracy (how many false positives or negative decisions did the software make?) all counted.

For the previous two releases of ProcessControlApp, they’d all worked together, along with two testers and a writer. They knew how to work together as a team. They told the VP of software engineering they needed that same writer and two testers.

I was the SQA manager, so it was my job to assign testers to different projects. When I met with the engineering VP, Nancy, I explained I had no testers available. “But you have to give them two testers. They can’t get the project done without them.”

“I have no testers. Look, here’s where everyone is allocated.” We discussed the other projects and their relative priorities. Nancy agreed that I had no testers to provide. I asked about hiring more people and Nancy told me I had no budget to hire more people.

I thought for a minute and suggested, “Hmm, I can’t do the testing and I can’t give them testers. But here’s what I can do for the team. I can be their project manager, help them see options, and remove obstacles so that they can get work done. Since this is a process control application with no GUI, this is possible. Difficult, but possible. Is that OK with you?”

“Talk to the team and see if it’s OK with them.”

About the Team

This team was a true team, not a group [1]. The team chose its own practices, which is part of what made it so successful. But a big part of what made this team successful was the trust the team members had for each other.

Although I was the project manager, the team trusted me differently than they did each other. They were happy for me to make suggestions about what they could do, but they never blindly took my suggestions. They always discussed each idea thoroughly. If they didn't reject the idea, they adapted the idea so that it would work for them.

Becoming Part of the Team

I set up a meeting with the team for the next day, and explained, "I can't give you testers; I have no one to give you. I can't do the testing myself—I still have my management work to do. But I can offer you my services as a project manager, especially if you need someone to run interference and remove the other obstacles that will arise."

This was my first meeting with the team. I'd known these guys for a while—some for more than 10 years socially. Here's how I remember that meeting.

Dan, one of the longtime team members and one of the original developers, said, "Well, Johanna's OK as a tester and better as a project manager, but we really need more testers, not another manager. Sorry, Johanna." Dan gave me that lazy smile and leaned back in his chair.

Fred agreed: "JR, it's nice of you to offer, but we really need testers." Fred was also one of the original team members. He drummed his fingers on the table and his knee-thumping kicked up a notch. Fred was a great developer, and had a few nervous habits.

"Well, I can't give you testers. I have no one available, and no money to hire more people. I offered to be the project manager not because I think you need management, but to help you with ideas about how to work differently than you're accustomed to. It's clear to me that the team had a great process that worked before. But that process depended on the testers. I can't give them to you."

Clyde asked, "Why? What's the problem?"

I pulled out the testing assignment sheet from my conversation with the VP and explained how the other projects needed the testers and that this project, while important, was lower priority than the others.

Dan frowned: "That's just stupid."

I leaned forward. "I already checked with the VP and Big Cheese. This project is number 6 on the list, and I have testers for only five projects. I am not trying to make your lives miserable. I know it sounds like 'I'm from the IRS and I'm here to help.' Except that I'm from management.

"But you guys know me. You know that I know a lot about projects and that I don't want to tell you how to do development. I can help you think of alternative ways to do this project. But only if you want me."

Dan leaned back in his chair and exhaled loudly. "OK. Why don't you go away for a while and we'll discuss it?"

"OK, I'll be back in my office. Call me there."

I received a call about fifteen minutes later and returned to the conference room.

Fred said, "JR, the team discussed it, and we'll take you. But you better have some good ideas. Not everyone is convinced we need you."

"That's fine. If you don't need me, fire me. Well, not really fire me, just tell me you don't want me, and I'll stop helping. OK?" I asked.

Everyone nodded or said OK.

We talked about how they wanted to work and how I could help. I had a few tasks, including setting up a standard meeting time, helping them get conference rooms for reviews, and seeing whether I could get a few cycles from the testers who'd worked on previous projects to see if they could fix the regression test scripts.

"How did the regression test scripts break?" I wanted to know what had happened, so I could understand how much time the testers needed to provide, to see whether this was even a possibility. One of the more junior members, Sam, blushed, and explained that he had improved a global class, and now several things didn't work. He had looked at the regression tests, but he didn't understand how a couple of them worked.

"So, maybe I should see if the testers could spend an hour explaining how the regressions are set up for the whole team. And then spend some time with you, unraveling this problem."

"Yeah, that would probably work."

Starting Off Right

This team had more going for it than many other teams. They knew how to work together as developers. They were talented people who all wanted to do a good job. And they were open to working in whatever way the project required them to work to do a good job and meet Big Cheese's commitments.

The team had a "gentlemanly" culture. They didn't swear when things went badly. They did let off steam with pranks. It was acceptable to call each other "brain-dead" when someone made a dumb mistake. But it wasn't acceptable to use profanity or be late for meetings.

Technically, they had other assets. Their automated regression test suite had worked for the most recent release. They had an automated build system. And they were familiar with code review, unit tests, and continuous integration.

Before the team started on the performance work or the new features, they decided to fix the regression test suite, so they would know that the current builds worked without having to do manual testing.

The developers and the previous testers convened a regression test meeting.

Jack, a tester, started explaining how the regression tests worked. "OK, everyone look at page 3, that's where the central loop for the tests is."

Sam interrupted, "No, that's not the central loop. Look at page 6—that's the central loop."

Jill, the other tester, piped up, "No, you just don't understand how it works. Look at the main part on page 2. See the sequencing?"

Sam glared at Jill, "Do you think I'm brain-dead? Any fool can see that the loop is on page 6."

Jill looked at page 6, and said, "Oh, now I know why you think that. Look at this call over here and look at that loop over here." She pointed to the different pages, explaining where she was each time.

"Oh, that makes a little more sense. But why didn't you write anything about that in the comments?" asked Sam.

"Because I ran out of time, and thought I would be working with you on the next project. It never occurred to me that the company wouldn't put testers on this project," replied Jill.

Everyone turned to look at me. "Hey, I'm sorry, but I can't put people on this project who don't exist. Jack and Jill, you know your project has more revenue at stake." They nodded. "Well, this team will figure out what to do to make this work, even though it's not the best situation."

The testers walked through how the regression tests worked, covering the tricky part that had stumped Sam. The team discussed how they wanted the regression tests to work, and divided them into several chunks, assigning different team members to the chunks. Sam took the tricky part "to get more experience with this test framework" and the testers agreed that he could ask them questions if any arose.

Within a few days, the developers had the full regression test working. In addition, they had chunked it into pieces so that they could use pieces of it to take performance measurements and verify that the new features wouldn't break anything else. Now it was time to start on the new work.

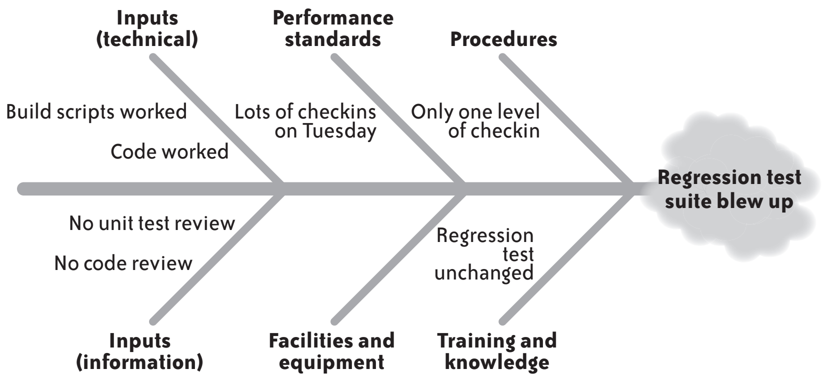
Everything seemed to go swimmingly until it was time for the first feature to be integrated. Every piece of the regression test suite blew up.

Solving Problems As a Team

It would have been easy to blame the developer, Fred, who had integrated his feature. But that wasn't the team's culture. There was a rousing chorus of "brain-dead code" at our meeting. I suggested we do a quick root-cause analysis at our team meeting, to see what had gone wrong. I was sure we had several issues at work, and it would be worth our while to see what was happening. The team agreed.

The team discovered several things had occurred (see the following figure):

- No one had looked at the code before it was checked in.
- The team was no longer using their multiple levels of check-in.
- "Too many" people had checked in on the same day.
- No one had added anything to the regression test suite to check for new features.



Root-cause analysis diagram

Dan took the lead. "OK, what do you guys want to do?"

Fred replied, "Look, we all know how I can be." Everyone except me nodded, grinning. "I really want someone to review my code each and every time. I'll write unit tests, especially since we don't have testers, do we, JR?"

I replied, "Nope, we don't. Unit tests are a great idea. They aren't the same as system tests, but they should prevent this problem."

Dan said, "But we're not done yet. Just code review and unit tests aren't going to be enough. We're going to have to go to multiple levels of check-in, like we did on the last project."

I replied, "You don't have to do that. You could all just check in all the time, and let the build and unit tests catch the problems."

"No, are you brain-dead?" asked Sam. "That will never work. We need multiple levels of check-ins."

"Well, let me explain how it works," I started.

"Look, you might have used that on a team with testers, but without testers it ain't gonna work," replied Sam.

"Well, if you feel that strongly about it," I replied.

"You bet I do."

"How about everyone else?" Everyone else nodded. "OK. Let's see how the multiple levels of check-in work before we consider an alternative, OK? I'm going to want to build a rolling wave schedule [2] with you so I know what's going on and can track it."

Sam was concerned. "This rolling wave schedule—what is it and how will it help me? Am I going to need to do anything more than my work?"

"Well, you'll all need to spend about 30 minutes getting the schedule ready the first time. Then, we'll update it with your status reports and my one-on-ones with you. If we run into trouble, we'll have to spend another 30 minutes all together, but that's it. Will that work for you?"

"Uh, OK."

"Sam, you know, JR isn't brain-dead, at least not yet." Fred started, and everyone chuckled. "You haven't worked with her before, have you?"

"Nope."

"She pretty much leaves us alone, but she runs interference. If Big Cheese comes to you with a request, you suggest—politely—that he go to JR, and she'll take care of it. Besides, if we don't like what she's doing, we can fire her. Right, JR?"

"You bet!"

Now the team had some safeguards (the multiple levels of check-in) for managing the issues of people working independently and wanting to check in at the same time. They decided to add code review back into their practices. In addition to code review, all developers were now required to write unit-level tests for all their code.

The team decided not to add anything to the system-level test regression suite. They could not imagine they would have enough time to manage the development and that level of testing. But they were quite happy to write unit tests for all their code.

What Code Review Looked Like

The team chose to use a modified Fagan-style code review. Each code review had a moderator and several reader-reviewers, and the author was the scribe. In a true Fagan code review, the reader-reviewers would each read the code individually in advance, and read the code out loud at the review. But this team chose not to read the code out loud during the review. The moderator would ask, "Does anyone have any issues on page 1?" Everyone would respond with "No," or "Yes," and explain his issue. Similar to a Fagan code review, the reviewers would hand their small issues to the author at the beginning of the review.

One of the reasons for not reading the code aloud was to save time. If the developers had chosen to read every line of code aloud, the team would have had to schedule many more code reviews, because each one would have taken much longer. With these code reviews, the team could review 50 pages of code in one sitting.

The team realized they had to trust each other to explain at the beginning of the review if someone hadn't finished reading the code. In those cases, the review stopped where everyone had finished reading and another review was scheduled for the next day.

If the author had concerns about how he was supposed to fix a problem in the code, the author and the moderator managed those concerns together.

Unit Tests

The developers knew they had to keep up with testing. But as one developer said to me, "I'm just not nasty enough to be a tester. I know how to do unit tests, but I'm not very good at system tests."

Nancy and I and the team agreed that we would incur the technical debt of not having enough system-level tests. But the team agreed that there was no way they could meet the deadline without having unit tests for all of the code they wrote.

The developers wrote their code and then wrote their unit tests. One of the developers started using McCabe's tools (there was a freeware Unix version) to check that he had written enough unit tests for each of his classes and modules. As the developer explained, "I'm really good at writing tests for everything that works. I always seem to miss the piece that doesn't work. With McCabe's, I'm less likely to miss that one thing."

Check-ins

I tried to convince the team to use continuous integration, but they didn't buy it—at least, not from me. But the team developed another approach to continuous integration, by staging the integration in a way that made sense to them. They chose three levels of integration:

Raw
: The code that had been compiled and unit-tested

Cooked
: The code that had been through code review and any issues fixed

Golden
: Code that had passed a build and all the existing regression tests

Each developer was responsible for compiling and unit-testing his own code. If a developer fell behind, he was supposed to tell me, and I would work out the scheduling issues with the team.

Builds

The team chose to kick off an automated build every night at midnight. Every night, all the code labeled "golden" was built, and all the unit tests and regression tests were run against it. The results were emailed to each team member.

As soon as someone checked in code, he labeled it "raw." Every developer built a raw build locally, to make sure he hadn't broken anything. Once the code had been through code review, it was labeled "cooked." Once a developer had "cooked" all his code, he promoted it to "golden."

Schedules

We had a meeting the following day, just to start develop the rolling wave schedule. I posted the months on stickies across the top of the wall, along with the known major milestones, such as internal demo, initial customer demo, and customer release, under their correct months. I listed the features down the side on a piece of flip-chart paper, in the order the developers told me they were going to implement them.

I asked, "What will it take us to get to these first two features?"

Dan replied, "No, JR, we're working in parallel. That's the wrong question. The real question is who will do what when?"

"OK, as long as we don't plan the whole darn project at once. The idea of doing this on stickies is to make sure we can figure out a Plan B if this plan doesn't work. Remember, when you write your tasks on stickies, make the tasks as small as possible."

Everyone wrote his tasks on stickies. Most of the tasks were one or two weeks long.

"OK, guys, I'm in trouble now."

"Why?"

"Because I am not going to know for too long if anyone is in trouble. Neither will you. You have to make smaller tasks. I bet each of you does something every day, right?"

Heads nodded.

"OK, then write all those things down, just for this first week. If you know enough about the second week, add that, too."

After they'd revised their first two weeks of stickies, I explained, "OK, I'm not going to ask you every day what you're doing. You're going to tell me if you don't make the progress you think you're going to make. If you know why, tell me that, too. Some tasks you'll finish on time, some might be early, and some might be late. My job is to figure out if you're late on a task how that matters to the rest of the team and the project. If it's a systemic problem, we'll address it in a group meeting. If not, the two of us will discuss it.

"I won't bug you for status. You'll email me status on Friday mornings, and we'll have one-on-ones on Tuesdays. That way I know if I need to do something a couple of times every week. OK?"

Sam frowned, looking concerned. "Johanna, this seems like a lot of work for us. At least for me. Do I really have to break down all my tasks into small pieces for the rest of the project?"

"I don't know how to effectively manage this project given the tight schedule and technical risks. Do you have a better idea?"

"No."

"Well, try it for two weeks and we'll see how intrusive it is. If it takes too much time or you think it's a waste of time, we'll address it at a weekly team meeting. But this has worked for me before. You need to plan only the next two to three weeks in detail. Not any more than that."

Status Reports

As the project progressed, the team members did email me their status on Friday mornings. I collated and sent out status by Friday afternoon, so everyone knew what everyone was doing. We used our team meetings to solve problems, such as the performance issues.

I prepared a monthly status report for Nancy and Big Cheese. The first three months were fine. But for the fourth month, we had encountered several problems with the performance enhancements, and were behind on them. I sent an explanation with my status report, and we continued.

Go Faster Now!

Nancy mentioned that the team was behind in our one-on-one. She said, "What are you going to do about it?"

"Huh?"

"Well, they're behind by a week or two. What are you going to do to make up the time?"

"Nothing."

"What do you mean, nothing??"

"Nancy, the team is working as fast as they can. They are working only on this project. They are doing what they can. Some of them are working overtime already, which is not good because we still have six weeks left on this project."

"Well, why don't you cut out the unit tests and reviews? That would free some time, wouldn't it?"

I was stunned. "Nancy, you are joking, right? Without the unit tests, we would have no testing at all. Without the unit tests, we have no way to know if what we're doing is good or not. You want to release this product in six weeks, right?"

Nancy nodded.

"Then let my team work the way they need to."

"But what about the code reviews? Maybe they can forget them. That might save some time."

"Nancy, it's been a long time since you wrote code, right?"

Nancy nodded.

"Well, it's harder now than it was for you. Did you ever write code that worked the first time?"

"Yeah, a couple of times."

"So you wrote code that worked the first time, maybe five times in your 10 years as a developer? Were you dealing with pointers? I thought you were writing in assembler."

"Yes, it was assembler. The pointers were not like the pointers in C."

"So why do you think that eliminating the code reviews is going to help us finish faster? That's just going to make it slower at the end, when we won't know what's wrong.

"Why are you asking me about this anyway? What's going on? Are you under pressure?"

"Yes. Big Cheese is telling me my job is on the line if we don't deliver in six weeks, not eight."

"Well, doesn't he tell you that about every project?"

Nancy grinned, "Yes, he does."

"So why are you believing him now?"

"I really want to make sure you are doing everything you can."

"Look, if Jack and Jill can stop their other project, and come back here and do system testing, this might speed up the project. But the developers can't work any faster than they are.

"I'll tell you what. In our next team meeting, I'll ask the developers if they can think of something that will help them work faster. Whatever it is, I'll let you know, OK?"

"OK, but see if you can do something."

Looking for More Speed

At the next team meeting, I asked the developers whether they had any ideas about going faster. Dan answered first. "Well, if you got us those testers…"

"Sorry, not going to happen. I checked with Nancy and I can't take them off their current project. Does anyone else have another idea?"

Fred cleared his throat. "We could start working overtime every day and a day on the weekend."

,ch27.20505 Page 376 Thursday, February 12, 2009 3:10 PM

“Do you really want to?” I asked.

“No.”

“Then let’s not bring that up as an option. Overtime this far in advance is not a good idea. Sure, if it was the last week, maybe we could put that extra effort in. But now, we’ll just make ourselves miserable. Any other ideas?”

Tim said, “I’m almost done with my last feature. Fred, do you want me to help you with the performance? Maybe we can work together on it.”

Fred agreed.

“Is anyone else just about done with his work? Maybe more doubling up will help?” No one else was almost done. “OK, keep at those little tasks. Let me know if you run into trouble. I’ll look for your status on Friday.”

Losing a Week at a Time

By Friday, it was clear that Fred was struggling with the performance enhancement. He hadn’t made any progress yet. Tim had just finished his feature work, and was ready to work with Fred. I sent my status report to the team and the higher-level status report to Nancy and Big Cheese. I was reviewing everyone’s tasks to see whether we could rearrange anything to help Fred. I was pretty sure he needed Dan’s help, too, to see other options.

Big Cheese arrived at my office, and said, “I need to talk to you now, Johanna.” Big Cheese strode in and stood over my shoulder.

I glanced up at him, and said, “OK, let me just save what I’m doing.”

“No. Stop doing that and listen to me.”

I turned around, and asked, “What’s the problem?” I motioned to my visitor chair. He shook his head.

“I have no problem. You, however, have a big problem. Stop those code reviews. They’re slowing down the project.” Big Cheese shook his finger in my face.

“But then we won’t know where the bugs are. We need the code reviews.”

Big Cheese stood yet closer to me. He shouted, “Stop them or I’ll fire you.” By now his face was red.

I stood up. “You’ll fire me for doing the right thing?” Big Cheese wasn’t a tall man, but I’m not tall either. I came up to his shoulder. But that was better than sitting down, at gut-height. Big Cheese took one step back.

“In this case, the right thing is to finish the project as fast as possible. Stop those code reviews. I expect to see more progress next week.”

Big Cheese left my office. I took a deep breath and walked over to Nancy's office, but Big Cheese was already there, screaming at her. I decided to wait to talk to her.

What to Do Next

I was in a bit of a quandary about what to do next. I wasn't going to tell the team to stop unit-testing and doing code reviews—I felt as if those two activities were going to save us. But I did want to be able to keep my job. And I didn't want to lose more time on the project. I decided to talk to Dan now. Dan was the laid-back type. He thought before he spoke, and not much ever seemed to faze him. Aside from being an original developer on the product, he would help me think through the problem. He wouldn't freak out.

I emailed Dan asking for about thirty minutes of his time for help with this problem. He should let me know and I would go over to his office. Dan has that lazy look that masks a brilliant brain. He's the kind of person who ambles rather than strides, who looks as if he's half-asleep and then says something that stuns you with its brilliance. If anyone had ideas about what to do, it would be Dan.

About ten minutes later, Dan sauntered over. "Hey, thanks for coming over. Want to go to your office?"

"Nope, I bet you have all the task information here."

"You're right. Let's take a look. I don't want to lose another week. I also don't want to stop the code reviews—or the unit tests—but Big Cheese is having a bird. I need to show some progress next week."

"Did he threaten to fire you?"

"Yup."

Dan paused for a minute and frowned. "Uh-oh. He usually only threatens to fire Nancy. The last time he threatened to fire a manager, he did."

"When was that?"

"Two weeks ago."

Oops. I was in trouble.

"Dan, I am not going to change what the group is doing. The code reviews and unit tests are working. But I would like to keep my job—at least for a few more weeks. Do you have any ideas about what to do?"

"Well, we could have a whole-team review of Fred's code to see where the problem is. Has Fred instrumented the code, so he knows where it's slow?"

"He told me he had, but I didn't review the instrumentation. Maybe he's missing something?"

"Yeah, maybe it's time to review the instrumentation. Maybe we need more data, too, about where and why Fred thinks it's too slow."

"Dan, let's talk about where you are. I think Fred needs more help, and you have as much experience with this code base as he does. If you help him, what do we need to do about your work?"

Dan and I discussed options for his work, and decided not to change his work yet. Dan would ask Fred about performance tests, and suggest a test review and a code review. Dan might suggest an instrumentation review. But that's all Dan would do until I met with Fred on Tuesday—unless Fred asked for help.

I was still concerned. It's not every day a senior manager threatens to fire you. But it was clear to me that the team had accepted me because I was helping them get their work done. I wasn't going to prevent them from finishing the work. But I did need a plan.

Retaining Integrity

I was caught in a bind I hear many managers and project managers discuss. If they do the Right Thing, they're in danger of losing their jobs. But if they cave in to what a senior manager wants—no matter how misguided—they lose the respect of the team. In this case, we would have also endangered the schedule.

I chose to keep my integrity and the integrity of the team. To be honest, it was an easy decision. If I'm working with a team that's working well, why would I do something to disrupt their flow?

The Rubber Meets the Road

We decided to use our next team meeting to review the instrumentation and performance issues in the code. I played the role of scribe, since I didn't know enough about the code to usefully comment. In preparation for the meeting, I suggested the team consider reading the code out loud and that we plan to spend two hours on the review. I added a note at the bottom of the email: "Remember, don't tell anyone we're doing this long of a code review—Big Cheese will have my head." At the time I thought I was joking.

We slowed that code review down to a crawl. After two hours, we'd gone through 15 of the 30 pages. Fred had a number of ideas about how to re-instrument the code, but no ideas about how to work faster. We'd been at this for two hours and had no more mental energy.

"Let's stop this for today," I suggested.

"No, we can finish," Sam said.

Tim, a.k.a. "Silent Man," piped up. "Sam, no."

"No?"

"No way. We're tired. We've been focusing on this code and the instrumentation, and it's gotten us somewhere. But just spending more time right now isn't going to help. We need a little distance and a break. I need a bike ride."

"Look, Tim, I know you get all these great ideas on the bike, but…"

"No, I'm learning. So are you. JR, can you book us a room for 10 to 12 tomorrow? I think we can finish then. And it not, we'll figure out what to do next. Sam, listen to me. Go play with your dog or work out or take a shower or something. But do not look at this code anymore. Got it?"

"Uh, sure." Sam looked shell-shocked.

As Tim left, Dan chuckled. "Sam, you didn't realize that Tim is really the power behind this group? Time you learned."

I booked a conference room and emailed the ideas we had so far to the group, as well as the conference room location for the next morning. The only problem was, it was on Exec Row, near the senior management offices.

At home that night, I explained to my husband what I was doing. "Honey, remember that conversation about code review I told you about a couple of weeks ago?"

"Yup."

"Well, we're doing a two-hour review tomorrow right near Big Cheese's office. I might be home early, looking for a job."

"Well, it won't be the first time you've been fired, will it?"

"I prefer the term *layoff*. That's what it was."

"Oh, fine. Just start on your resume tonight. Oh, and call me at lunch to tell if you're still employed."

The next day, we all arrived at the conference room and started on page 16. We made some progress. At 10:40, Big Cheese barged in.

"Johanna, what are you doing? This looks like a code review."

"Does it? We're trying to figure out how to make up time on the project. We don't have a plan yet, but we're getting there."

"Glad to see you've come to your senses. You knew I was right."

After he left, everyone cracked up. Tim was the first to speak. "JR, you can't play poker, but you can lie without blushing?"

"I told the truth—at least, partially. We are working on a plan. It's just that we're using a tool he doesn't like." Everyone laughed again. "OK, let's get back to this."

By noon, we had two more suggestions for instrumentation, and one suggestion for changing the code. Fred and Tim were developing their inch-pebbles, coordinating who would do what, and promised to let me know by the end of the day what their plan was.

I spent some time reviewing the schedule and floated the idea of a phased release to the team. We could release the new feature set on time, and release the performance enhancements two weeks later. I asked the team to consider that as an option we could discuss at our next team meeting.

Success at Last

The team agreed to the phased release. They kept up with the code reviews, unit tests, and multiple check-ins. I maintained the projectwide inch-pebbles. As other team members finished their features, they worked with Fred and Tim on the performance issues. We got lucky. At the desired date, we actually had everything done: the features and the performance changes.

The next day, we had a little celebration with ice cream sundaes and we discussed what we'd learned.

Sam started. "Johanna, I had no idea that breaking tasks into small chunks would really help that much. But I'm even more impressed with the code reviews and the unit tests. I could never have done all this without them."

Fred added, "I wish I'd asked for review earlier. We wouldn't have needed that overtime."

Tim said, "Maybe now you'll believe me. Now all I have to do is to get you to start eating organic food." Everyone laughed.

I added, "I'm really impressed with what each of you accomplished, separately and together. Thanks for the opportunity to work with you."

Epilogue

I stayed with that organization for another few months and then was fired. (It was the best career move of my life.) That team stayed together for another two years. Big Cheese separated them when it became clear they worked well as a unit and were influencing other teams.

I've maintained ties with some of them. Several still work together. A couple have moved on to management positions, where they're working on re-creating that great team experience. I'm thrilled I had a chance to work with a beautiful team.

And the product? It won a Japanese award for quality. Big Cheese accepted the award, and said, "The team was inspiring in their dedication to quality."

References

1. Rothman, Johanna, and Esther Derby. 2005. *Behind Closed Doors: Secrets of Great Management*. Raleigh, NC: Pragmatic Bookshelf.
2. Rothman, Johanna. 2007. *Manage It! Your Guide to Modern, Pragmatic Project Management*. Raleigh, NC: Pragmatic Bookshelf.