Of Crazy Numbers and Release Criteria

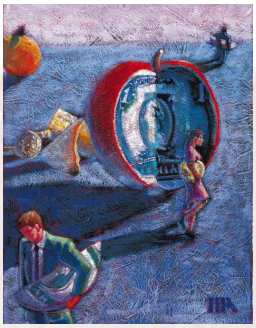
Johanna Rothman, Rothman Consulting Group

James Bach and I have had an ongoing, multiyear discussion about my definition of release criteria. He thinks I sometimes come up with crazy measurements based on no apparent foundation. He's right. I do. I use release criteria to decide when a project has a "shippable" product. I do this so that the decision to ship is a thoughtful team decision based on data and not dominated by the gut feeling of the most powerful person.

Sometimes a ship decision is made when people are very tired and stressed. Sometimes the ship decision is based on a corporate need for revenue. I prefer to reduce the stress of making the ship decision by addressing—before the last minute—some of the issues that cause stress. Quantitative release criteria are one way to do that, and I do use them. It's just that these release criteria can look arbitrary, even crazy, to anyone not familiar with the details of a particular project.

THE VALUE OF A MAGIC NUMBER

I recently worked as a project manager for a small middleware applications group. The group consisted of six developers, three testers, and a part-time writer. When I arrived, the group was in the midst of preparing for a beta shipment. After a short assessment, I realized the software would not be stable enough for beta at the time the group wanted to ship. The lack of stability was due to too many defects in the current release and too much time fixing defects from the previous release.



You very well might cling to the belief that each defect is an exception to the rule that your product is ready to go.

But the team was not tracking defects. So I developed defect-oriented beta and release criteria. The beta criteria required that all known bugs be entered into the bug tracking system and that there be fewer than 36 open high-priority bugs. The release criteria, then, were that there be no high-priority bugs open. There were other criteria, but these were the defect-oriented ones.

When I described the situation to James, he rolled his eyes and asked me, "Why 36? What is the significance of 36?"

I explained that when I realized we could not have a successful beta with the current software stability, I wanted the team to realize just how many defects we were dealing with. A real number of defects would do this because this team didn't think in terms of other defects, just the one they were working on.

The team had an interesting view of defects. At any given time, the developers operated on the assumption that the problem they were working on was the last one in the product. In hindsight, this seems a bit silly. But imagine you're in this situation. For the past five years, you've been stuck between two states: thinking that you're close to shipping and realizing that what you've just completed is not successful. Your senior management wants a release last quarter. Your customers want you to get to beta. Not all the developers understand the entire product and you're all exhausted. The testers don't completely understand how to test the product and they're tired, too.

What would you do? You very well might cling to the belief that each defect is an exception to the rule that your product is ready to go. You would keep your head down, work like crazy, and hope that this defect is the last one.

I wanted to burst this misconception. Rather than hope we were on the last problem, I wanted to create a buffer of problems and give the project permission to ship with some known and acknowledged problems. If we could agree on nonzero defect criteria, the team could get out of the rut they were in. They could stop thinking that they had created perfection and stop being disappointed when they did not achieve it. We would then be able to move the project to beta and ultimately to release.

ARRIVING AT THE MAGIC NUMBER

In truth, my original suggestion was that all known bugs be entered into the bug tracking system and that there be fewer than 40 open high-priority bugs. When I made this suggestion, I was called into the division manager's office. We had a fascinating conversation:

Editor: James Bach, Reliable Software Technologies, 21515 Ridgetop Circle, Suite 250, Sterling, VA, 20166; j.bach@computer.org

DM: "Johanna, you've offended the entire project team. What kind of a project manager are you? Why do you think there are so many high-priority bugs in the system? I don't think we have even 80 bugs total and I bet only a handful are high priority."

JR: "Well, based on our testing, I think we have over 100 bugs and most of them are high priority. I think when we fix a bunch of high-priority bugs we'll find more low-priority bugs."

DM: "No way. I bet we have fewer than 70 bugs total. Maybe only six high priority."

JR: "Well, we have a 40 percent fault-feedback ratio and the testers are finding more than five bugs a day now. We can't fix them faster than we can find them. I think 100 is an extremely optimistic estimate." A 40 percent fault-feedback ratio means that for every ten bugs fixed, four new bugs are created. (See G.M. Weinberg's *Quality Software Management, Vol. 2*, Dorset House, 1993.)

DM: "No. You're wrong. We can't have more than 60 bugs. If we do, I owe you a coffee."

You can see that if I'd let this conversation continue this way, the division manager might have denied that they had any bugs at all. This was a smart senior manager who just didn't understand software projects. He didn't understand that defects cluster and that sometimes if you fix something, 10 new ones pop up because you've now gotten to that code path. I decided to redirect the conversation.

JR: "OK. Let's come to an agreement here. For the sake of argument, let's say you think there might be 70 open bugs. Let's say I'm hopelessly pessimistic and half of them are high priority. Let's take 35 as the high-priority bug count, add one to give us some leeway, and call the beta criterion 'No more than 36 high-priority bugs.' OK?"

BEARING WITH DEFECTS

The division manager agreed as long as I paid up in coffees for every high priority bug fewer than 36. We started using the bug-tracking system. At the end of the first week, we had more than 100 bugs open in the tracking system, more than 70 of which were high priority. It was a struggle to reduce that amount to 36 for beta, but we did. And we were honest with ourselves.

It wasn't just that I wanted the project team to be aware of defects, I wanted them to be comfortable with acknowledging defects. They had to realize that they had produced defects and that they would not necessarily be able to fix all of them for a given release. That, in turn, makes awareness of defects less overwhelming. It makes the defects bearable.

This project's customers were ecstatic about the beta. They'd never had a release this good from this organization. When we shipped them the final version, they called to congratulate the project team.

I'm not claiming that fewer than 36 defects is good quality and 37 defects would be unacceptable quality. I used this defect metric as a way to help the team make a first step in becoming aware of quality. If someone on the team said, "I don't think 36 is the right number," then we would have a conversation about what quality really means for this product.

In this case, the beta and release criteria helped the organization look at what they were doing. In addition, each criterion gave the organization common goals and a common language—and it gave them permission to be good without having to be perfect.

In this case, the defect counts were an aggregate team measure. The team could only succeed if everyone worked together as a team. I pushed the team to a specific goal that then allowed me to help them improve their product development process (use a bug tracking system, plan their work, use a configuration management system, and so on).

By the end of the project, the team realized 36 was just a nonzero number, a number that did not demand perfection. The number was valuable because it got people thinking about the data—where it comes from and what it's used for. The goal gave the team an incentive to use a process that would help them succeed. The team figured out how to get the work done.

I don't care within the context of the project if the release criteria look crazy, especially to people outside the project. As a consulting project manager, I need to make sure the people understand the reasoning behind the numbers by the time the project is over. In this case, the numbers helped people realize they needed to track and quantify some project data. Once they could track the data, they could choose what to do about it.

I don't care if the numbers have no relationship to what normal people with normal guts might think. I only care that release criteria help the project team ship the right product on time with defects the customers won't scream about. And that they do it without ulcers and exhaustion, and without driving each other crazy. ❖

Johanna Rothman is president of Rothman Consulting Group, a company that provides management consulting and training for product development organizations. Contact her at jr@jrothman.com.