QA vs. QC

As freshly-fried potato slices jostle through the drying vents at Frito-Lay, an electronic eye detects minute discolorations on the golden surfaces. These discolorations are due to normal color variation in potatos, but field tests have shown that Ruffles' consumers would rather consume a perfectly consistent yellow chip than one that belies its vegetable pedigree.

In milliseconds the eye identifies unsightly interlopers and just as quickly a jet of air blows the imperfect crisp off the conveyor belt. The unacceptable chips probably get incinerated, but I prefer to imagine them dumped into an Olympic-sized swimming pool where employees are encouranged to jump in and loll about in greasy crunchiness.

This is known as Quality Control (QC): The product is finished, ready for shipment, but we give it one last review before we let it go out the door.

As far as I can tell, the entire software industry has misnamed their QC department by calling it Quality Assurance (QA), and in so doing we've done ourselves a disservice.

QC does not ensure quality, it only exposes lack of quality. QC can flick bad chips off the line but it cannot tell you why you're producing so many bad chips. And that of course is the important question. (Answer: Potatoes with too many eyes.)

If we're serious about "quality assurance," we have to admit that producing quality software must enter into every step of development from use cases to requirements to design to writing code. All QC can do is notice that quality was not, in fact, "assured."

This is not to say QC is useless; on the contrary it is necessary. But it's a finder of problems, not a solver of them. **If software quality is a problem, "beefing up QC" is not the answer.**

Only the Germans.

QC could help us find the answer though. We're bombarded by in-vogue quality techniques — unit tests, code reviews, scrum sprints, pair programming, agile requirements, TDD, BDD, ADD — but which of these would have the most impact for you specifically? Your programmers, your culture, your product? If QC is measuring the nature of the defects that escaped development, you should be able to establish a baseline, try one of these techniques, and measure the effect.

For example, I just attended a conference where Jeff Sutherland showed Scrum delivering fewer defects by an *order of magnitude*. Who knows whether you would see similar results, but if that's the claim, QC should be able to measure it!

In the end, all this would be clearer if we called the QC department "QC." Then "QA" is something *else*, something everyone should be thinking about and working on.

Either that, or keep calling that department "QA," but let them do their jobs. That means QA isn't stuck at the end of the assembly line, powerless to do anything but complain about the product shooting out. Instead they're involved at every stage, helping to define and develop a testable product.

Just another example of how important it is to name things properly.

Do you have better ideas for what to call these roles? Have you tried sausage-flavored Pringles?