The Power of Collaboration

By Matthew Heusser Matt@xndev.com

Imagine this scenario:

The company has programmers, analysts, and testers. The programmers complain the analysts do a poor job; they have to figure out what the software should really do. The testers come along and find problems with that, unconsidered scenarios, and we have to do everything over again. When DevOps comes along, the company creates an entirely new team, the *DevOps Team*, that creates new clusters running Kubernetes with Docker. All of this is done without really talking to any customers, so the solutions sort-of work - they 90% work. They make a nice demo. Actually, using the cool DevOps things for real work is entirely different. Nobody really understands what the Security team does - it is mostly system hardening and auditing. When the company needs to do real performance testing, or penetration testing, or accessibility, it brings in experts who work for a week (or a few) and produce reports. Those needs are generally driven by a legal requirement, not any customer. DevOps does not displace or really have much to do with Operations, who continue to purchase the hardware and keep the data center running. A few people are doing things in the public cloud, mostly driven by credit cards. Management is uneasy because those cloud costs are hard to predict. They would rather pay too much 90% of the time than deal with unpredictable costs.

Consider the story I told above as a 10 on the conflict and rework scale. It is sure to result in products developed and redeveloped two or three times that, in the end, are a sort of awkward compromise that aren't really a great fit for any customers. When companies like Borland and Compuware developed a reputation for doing that kind of work, they dropped from innovative brands you looked forward to at conferences to a subdivision of a MegaCorp.

If the opposite side of the scale, a zero on the scale, is unicorns, rainbows, love beads, flip-flops and Kumb-bay-yah -- and, more important, mythical-level performance -- where do you stand?

I'm guessing you're not a zero. And if the comparison is to a literal land of fairy tales, you probably don't want to be a zero. Everyone knows fairy tales are not real. Today I'd like to take some of the mysticism out of collaboration, then talk about how to get there.

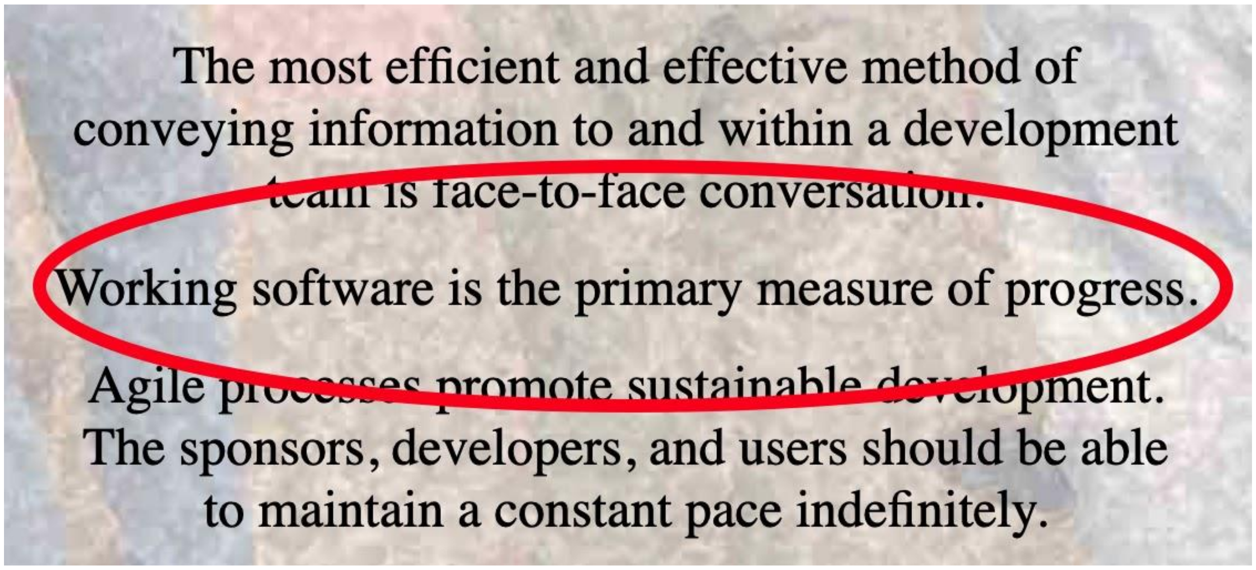
Let's start with the idea of "my work."

Beyond Unicorns and Rainbows

Most people in software understand this idea: the analyst needs to create the requirements document (or story if you prefer) and give that to the programmer. The programmer writes code, the tester takes the story and tests it, moving it to the right of the Kanban board.

Rubbish.

The job of the team is to produce value for customers. The primary measure of value for customers is a working software. Analysis, code, and "passing tests" are not in and of themselves “value.” They could be considered "work in progress inventory," as could stories that are worked on. In lean terms, excess work in progress is waste.



The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Excerpted from the Twelve Principles of the Agile Manifesto.

In constraint theory, we talk about the bottleneck. That is, the thing that slows down delivery, that constrains delivery. In the short-term, if one role is a bit slower, work will stack up in front of that role. That is work in progress inventory -- waste. The delivery speed of the entire team is slowed down by that step. Most of software development responds by asking "Why is QA always the bottleneck?" thereby shaming and judging, forcing the testers to either skip steps, let the software go out with known issues, or continue to slow things down and get bad annual reviews.

Those are not good choices.

The short-term fix is to throw resources at the constraint until it is not the constraint anymore. That might mean having a Database Administrator or a Developer do testing. Longer term, you either balance out the staffing, or make people more generalized, able to do more than one thing at once. That sort of being able to do more than one thing, of working together to strengthen what is weak instead of insulting it, can be the start of collaboration.

Toward Collaboration

The next step would be to work together to understand what we are building - to have all the roles involved in the process. Several friends, people I respect, get scared with this language. They think it is "hippy dippy stuff" where anyone can make up whatever they want; they prefer “The Product Owner” role in Scrum. With a Product Owner, someone decides what to build, and that person has authority and responsibility.

While I don't think a product owner is always required, I'm not opposed to them either. What I will say is that programmers and testers often have ideas they can put forward toward building a better product. Testers, in particular, can help define the nitty gritty details of how a product might act under odd conditions. Programmers can envision what the product is capable of when product owners might not, and often understand the conceptual limits, or how to combine features to make something amazing. To this day I still remember one programmer, working on a system to calculate a sorted list of doctors by distance. This was some time ago, and Ben wanted to integrate the feature with a new product called "Google Maps," so customers would see the locations of the doctors’ offices, and could click for details. Told the feature was out of scope, Ben built it anyway. The senior management of that project won an external award for it, and there was talk of filing a patent application.

That's a pretty good deal for a product they told him not to build.

I'm not trying to be critical here. Instead, I am saying people can contribute from places you might not expect, and they can pull the team forward in ways that might surprise you.

So how do we make the workplace more collaborative?

Pushing the scale

Before Agile became the ruling paradigm for software, I once got a poster of the Manifesto and put it on my office space. One leader walked by and told me she was offended by it. Three years later, when they needed someone to rescue a failing project, I was put in charge, and told that I could "do your thing on this one," but "don't tell anyone how you did it."

The number one thing to do to change your organization is to provide results. Consistent results speak for themselves. Before you preach, before you lecture, before you put a change request in the ballot box, use your discretionary time to do things in a better way, and get consistent, predictable results that beat your peers.

This is, by the way, how the cowboy coders who create amazing things in a weekend stay in business -- they create little innovations no one else would have thought of that save their peers time and effort. For production work, they do 80% of the work in 20% of the time. Of course, that is a buggy mess that needs another 95% of the time to test, fix, and retest. Notice the total is 115%. Still, the cowboys *innovate*.

If you can, partner with the cowboys. Work to create a vision for the product they implement, so they can actually do a 99% job in 30% of the time. Once you've proven yourself, influence your peers to work in your way of working.

In my experience, the challenge is when you want to go outside your role - when testers want to influence developers and when you want to bring other kinds of testing into the fold. That's when speaking a similar language and using similar tools can help. In a subsequent post, I'll talk about collaboration from a system's perspective, and the role of technology in supporting collaboration. In the meantime, two things: first, work within the system to have excellent performance as much as it is possible to do so; second, work to change the system.

We can talk about how soon. For today: What have you done, and what are your ideas? I'd love to hear about them in the comments.