How is Knowledge Work Different? (Why Johnny, Inc. Can't Code.)

The fundamental constraint of knowledge work – such as software development – is our ability to discover and create knowledge.

Imagine that you had spent Monday through Thursday working on an essay, a song, or even a computer program. You tore up seven ways that you realized wouldn't work. You took a walk outside. You slept on it. You did research and found an eighth way that looked like it would work, until you showed it to a friend who pointed out a serious flaw with one part of your idea. By Thursday afternoon you had nearly finished a ninth way that combined the best of your previous approaches and looked to be nearly done. Then Thursday night your cat walked across the keyboard and somehow deleted everything you typed that week.

Would it take you another four days to get back to where you were? Of course not! By 10AM Friday you'd be ahead of where you were Thursday night. Therefore what was the actual constraint on that knowledge work? It wasn't the typing. It was the learning and creation of new knowledge.

Why Johnny, Inc. Can't Write Software

Have you noticed that companies with lots of money and people can't seem to develop software that works very well? I know interns who can make better websites than certain airlines can. Those shops aren't built for learning. They're organized as if software development is just about typing code. When that fails, they try to fix it with harmful measures: adding incentives and "accountability", creating new roles and departments, hiring even more people to type code.

Hint: Johnny, Inc. Is Not "Understaffed"

I recently visited a big bureaucratic agency that had trouble doing things that are easy for a small co-located team using modern development practices. I almost fell over when I heard one of the dozen project managers say "We're understaffed." No … that's not the problem.

Implications

What would be different about an organization that recognizes that better software is really about our ability to discover and create knowledge?

Traditional Organization	Continuous Learning Organization
people work in separate cubicles, offices, or cities	each small team works at the same table
more people	fewer people
more roles	fewer roles
more departments	fewer departments
single-skilled workers	multi-skilled workers
fewer people learn from customers and end users (https://www.youtube.com/watch?v=RAY27NU1Jog)	many people learn from customers and end users
private code (my code/your code) practices or policies	internal open-source practices
people spend time reading/writing “tickets”	people spend time sharing screens and whiteboards
focus on execution, quantity, and velocity	focus on impact
learning happens only during brief training periods or on employee’s own time	learning happens every day on company time
mistakes are not survivable	experimentation is encouraged
retrospectives affect teams only	retrospectives cause improvements to company policies and structure (https://less.works/less/framework/overall-retrospective.html)
managers coordinate, reallocate, motivate, and mediate	managers are capability builders (https://less.works/less/management/role-of-manager.html)
extended overtime	adequate sleep

Becoming a *learning organization* isn’t something that gets solved once, by a training program, a reorg, or a one shot “Agile transformation.” It doesn’t happen just by talking about organizational culture. It is a deep change that cannot ignore the policies and structure of the organization.

Japanese version: 知識創造を行う仕事は、どうしてこうも異なるのか？（なぜあの会社はコーディングができないのか） (https://scrummaster.jp/how-is-knowledge-work-different-jp/)