Culture as Code


Culture as Code
Werner Müller posted on Aug 31, 2019
Welcome to culture as code! Now: obviously, this is a lie - culture is about social behaviour, about norms found among humans. Culture is more than what happens at your workplace. It is found in music, art, religion. You can't possibly put that into code. And you're right, you can't.

But hear me out.

Let me tell a story showing that code can indeed be the source of influence on human behaviour - and help to improve on your team's culture. And by adding that human to the mix, make this work.

Suspect a required movement

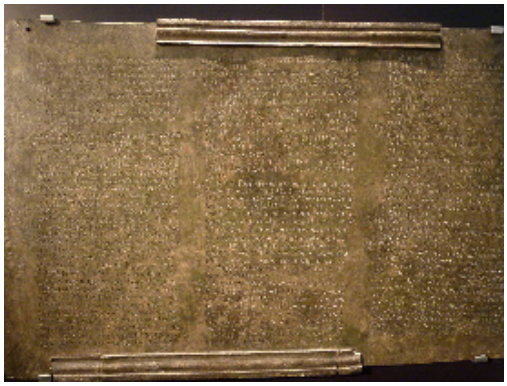
The situation in software development if often diverse. It all started with good intentions, enthusiasm, even huge ambitions. Then it all drifted into some arrangement of progress and staying alive. A drift that companies often try to come by with organizing work into departments, increasing delivery cycles. And people get used to those cycles. Rely on them, maybe even demand them.

A culture driven by processes and rules emerges. Not too bad, not great either.

So the cycle continues: That application needs to be done in a couple of months?! We will do testing when the software is more complete. Does the CI server show a broken build? We can fix that once we need it. It is more important right now to be able to work locally. Do the integration tests do whatever they want? We can take care of those once all services are integrated.

This mental model of postponing important things in favour of urgent work will only change when the culture changes! We need to change our culture!! This is what developers are born for. This is what developers are given time for!!! Culture changes!!! Psychological work, social competence. Well, maybe not every developer is up for the task. Still, this situation needs to improve.

Starting a movement



But what are developers good at? We declare our manifestos to follow, search for patterns to implement and re-use, define principles and write guidelines. Come up with practices to repeat the next time, since they worked. With rules to follow to not stumble over the same thing again.

Rules - we can implement those - can't we?

Investigating the situation

Some rules are surely harder to implement than others.

Like: Be humble to each other. Nice rule. Sounds good. Sounds important. The implementation may require some Commander Data brain. No one got time for that.

New rules, more concrete, more useful!

So here we go: The CI build is broken? Drop all your pencils and go fix it!

The code analyzer brought up a crime to light? Drop all your pencils and go fix it! The security scanner found a new vulnerability? Drop the pencils and update that library some insane person added! The deployment failed? Drop all your pencils and fix the deployment. The service crashed miserably? Drop all your pencils and investigate!

These are all important activities, and the quality in which we perform them affects our efficiency and how much time we can spend on other things.

We don't do all this because we would need to check the CI server, the code analyzer, the security scanner, the deployment tools, the service logs and monitoring tools - just to be able to know what is going on, if there is something to do at all and if the problem that came up is actually ours or caused by some other poor fellow.

So back to our plan: what are developers good at? Well, I do hope at writing code! And all of the above can be expressed in code and be shown to your team in an instant. Is that a culture change? It sure is not. But we ain't there just yet. Let's keep trucking.

This overview will help us to identify what task is in the wild and should be done - or - honestly - should have always been done - but we didn't - because it was too fiddly to check everything. So we can go from dropping the pen directly to addressing the issue, taking a huge jump over all the forensics.

Once we have a collection of this information and can show a summary, we make transparent what is hidden somewhere. No more constant searching for the same information. For us and others. The situation of our software is transparent. And obvious state in software? Isn't that beautiful.

Writing a build monitor

Well there is one already, isn't there? Sure there is. But none that shows our tools, our state, our information the way we want it. And - to anticipate a little bit here - we want to do more than just rules on some state. We target culture.

So once we wrote a small application that collects all the information we need (from the CI server, the code analyzer, the security scanner, the deployment information, and the application health state itself) we figure: we have quite some information at hand!

For example: is the production version newer than the version on pre-production? Is the application un-healthy because of some other service that is unavailable - because that may just mean to let them know about it? Does the meta-information of our application suggest any actions? Did we configure the application correctly so all monitoring tools can work? Was there a SNAPSHOT version deployed? Is the version that was created by the CI server the one that was deployed to the test environment? Or is there a version gap? Is the test-coverage anywhere near where it should be?

All kinds of things can now be checked, beyond just a state in a single tool. The state of a delivery pipeline can be observed and checked against our rules. The rules we currently need.

Some may call this IT governance. But there is one big difference. These are our rules, they represent our current focus and priorities. Ideally these match - but let us not jump into this snakepit today.

Observe the change

Now as every team has a reddish screen, some will take on the challenge to go greenish. And they will notice that once something comes up, it is usually caused by other teams. These teams will now either: be lost and frustrated (because there is no one to lead them out of the mess), or they take on the challenge themselves. Because this is our tool and our rules. And no one wants to come in last. Some claim it - but I don't believe you. So application by application, pipeline by pipeline, team by team the situation improves. Because people care. Because they see the state, they see the effect of their work, they notice the improvements over time.

And when people start taking care where they previously didn't: there is your culture change.

So a simple build screen? A tool that allows implementing some custom rules? Does that work?

It does. But it does not on its own. Because tools are only tools. If a team does not find a way how to help themselves or to whom to turn to for embarrassing questions, they will ignore this. Delete the transparency and keep living in the mess. You need the humans willing to go through some essentials - application by application, team by team. You need some of them who care and carry this care to others. By making the things they care for transparent - for everyone.

This worked exceptionally well for me. Because you can always find other knights willing to ride along to battle. Because they are sick of living in the mud. And this way - step by step, floor by floor you will reach the roof under the stars. I will not argue it is a fast process. But some processes are more healthy if slow. And even with 40 applications, if you can only heal 2 a month, after only two years you will have healed them all. Looking at some mess today: don't you wish you would have started 24 months ago? Because today: it would be the roof and the stars!

So if it is not today: let it be tomorrow.

When we think about some mobile apps and how they changed how we meet people, how we connect and get in touch: then why would code not be able to influence and change a culture? Code certainly already did this on other occasions.

Yes! Code can change a culture, together with the people that hold on to it. I've seen it.

The tool that came out of all these steps was called "Mobitor" - because it needed a name at some point. You can give it a run yourself: https://mobiliar.bitbucket.io/mobitor/

The next steps are on www.cultureascodemanifesto.org to collect some guidelines to help you orient yourself on the journey.

The picture of the wall with laws is from wikipedia. The Roman bronze plate picture is from Claude. The Codex of Pirates of the Caribbean is from the fandom wiki (remember: it's just guidelines).

- cacman
- cultureascodemanifesto
- cultureascode
- development