

THE
CHEERBY
MODEL

Anneke Keller &
Tim Meeuwissen


*four seasons of innovating
IT organisations*





winter

In the winter season, the cherry tree has to put all its effort into surviving. In autumn it has shed its leaves so it can save energy now. If the tree isn't too young, it is hardy and more likely to stand the freezing cold. However, if temperatures drop too low for too long, or if the cherry tree's roots or trunk are damaged, it may not make it to spring without help. The only thing our cherry tree can do is enter full survival-mode and hang on.



*I*t was the first week of my new job at Jumbo. I was introduced to my team and colleagues, and my calendar quickly filled up with getting-to-know-you-meetings. As the new head of an IT-team I was hired to make significant changes in the department. They asked me to found the “Jumbo Tech Campus”, a new department that was supposed to become the tech innovation driver of the company. All the people I met were aware of this, so they were all curious and sometimes a little suspicious. I spent the first month or so mainly gathering information. I asked everyone about their role at the company, what they thought worked well, and what they found could be adjusted or improved. My days were long and strenuous. Gathering and processing all this new information took a lot of my energy, but it was worth it. My approach to effectively assess my new department and team was by listening and asking a lot of questions. Based on the feedback I got, I mentally visualised my new environment - how processes ran, how people worked together, how we collaborated with our partners, and so on.

In an ideal world I would have had enough time to talk to everyone and only after a few months make a plan on how to approach the Big Task at hand. It’s my experience, however, that you hardly ever get time for this. You were hired for a reason. There are urgent issues that need attention ASAP. This case was no different.

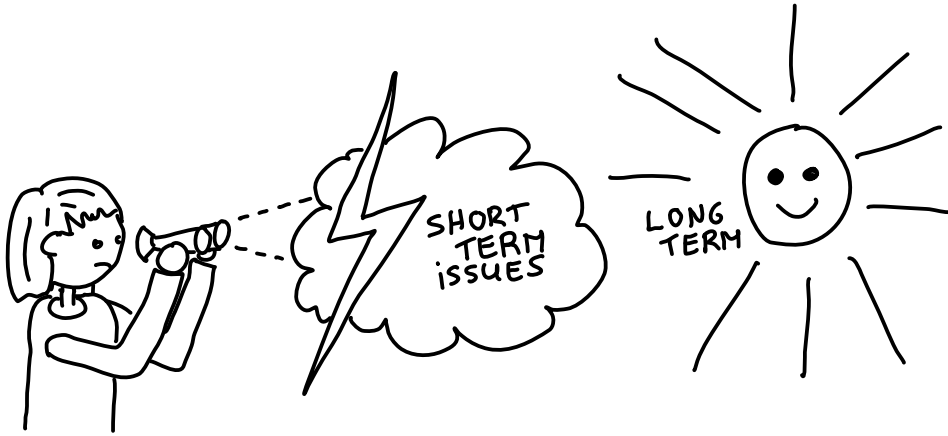
On my third day at Jumbo, something went terribly wrong. I noticed people on my team were making nervous phone calls, gathering around the desk of one of the developers and intently looking at his screen. It was 9 o’clock in the morning. I asked them what’s going on. ‘Well, we did a new monthly release last night and now customer service says users can’t order on our website anymore. To be honest, this is not the first time.’ My plan to take my time to make a proper long-term oriented plan had just flown out the window. This needed my immediate attention.

Stormy weather

The organisation is in wintertime. This means that we want to put all our effort in stabilising, making sure the team is ready for spring, and is fully equipped to start growing. For some reason, when I start in a new role leading an IT team, my new team is often in what I call “stormy weather”. Meaning that the amount of issues requiring direct attention is so enormous that there is hardly any time for anything else. This is often the case for at least part of the team. I don’t know why this happens so often, maybe I am attracted to this type of organisation or it is just a very common state to be in, as an IT team. The upside is that over time you get some practice in getting out of these situations.

When people ask me how to solve a situation like this, I always advise them to prioritise getting out of stormy weather over everything else. “Thunderstorm

mode” is a very discouraging state for a team to operate in. If you don’t have this problem and your team is currently in a pretty stable condition, I would



still recommend reading the following chapter. It will help you evaluate if you have given enough thought and attention to all the issues involved in “stormy weather”. Like the weather turning, situations can change quickly. If your processes and leadership aren’t prepared to immediately respond effectively, things can easily run out of control. Winter is for stabilising, even when the weather is calm.

The deeper you delve into the details IT entails, the more predictable it becomes. It’s all 0s and 1s after all. However, when it comes to systems, data, and platforms, it’s a whole different story. Due to its complexity, IT can be unpredictable, especially if you’re working with partially deprecated or legacy systems. This may cause teams to unintentionally create issues in the software solutions, or in the hardware configuration it is running on.

Thunderstorms are very counterproductive for a team, especially when they last for a long time. Much of the team members’ focus goes to solving short-term issues. Since these issues often cause major problems for the running operation of the software as well, the pressure to solve them is high. On top of that, the longer you remain in stormy weather, the harder

it becomes to distinguish what’s urgent from what’s important, but not urgent.

Managers are likely to assign these issues to the senior team members and are often involved with them themselves as well. In the past they may have learned that being on top of the problem solving process is part of their job and the more they focus on that, the better the outcome will be. This means that the team members who are not directly involved in solving the issues, will have to do their job without seniors and a manager to guide them. Chances are this part of the team will get unproductive, unmotivated, and may in the long term even cause bigger problems than the ones at hand.

When you are leading a team that is currently in this state, or when you are part of one, what is the best thing you can do? Just go with the flow, perform your tasks and wait for the storm to blow over? This might work, but it is a risky approach. This would mean that the head of the team, the managers and the seniors all lose sight of the long-term goals and the team may not get out of the storm at all. The storm may rage on and cause team members to get frustrated, fall ill, and even quit.

I normally take a different approach. First off, I'll focus all my attention on getting the team out of this stormy weather. This means that as a manager or leader of the team, I have to make

1. Fixing the processes,
2. trusting the team.

Let me explain.

Fix the processes

Working with a team that is dealing with a storm requires a number of things that may feel counterintuitive:

1. Burden the team with mandatory evaluation sessions,
2. Reduce testing time before code goes to production, even though the team wants to do more testing,
3. Step out of your comfort zone and start soul finding sessions with peers in other departments,
4. Tell everyone about what went wrong in the department.

I'll admit, I'm not really selling these tasks here. That being said, getting out of a storm isn't easy and requires

sure that not all my time is consumed by solving or communicating about the problems. Limit the time and energy spent on this to create focus on two other very important things:

effort and leadership from a leader, their team and their colleagues. Let's go through the steps one by one.

1. Close the feedback loop

Failing isn't bad for an organisation, it may even have important upsides. Failing means the organisation is experiencing new things and trying to make progress. Failing without learning on the other hand, is very bad for an organisation. It is costly and leads to customer dissatisfaction. So, I need to make sure my team learns from mistakes. In other words: close the feedback loop.

When issues occur on your production system, something is failing. It might

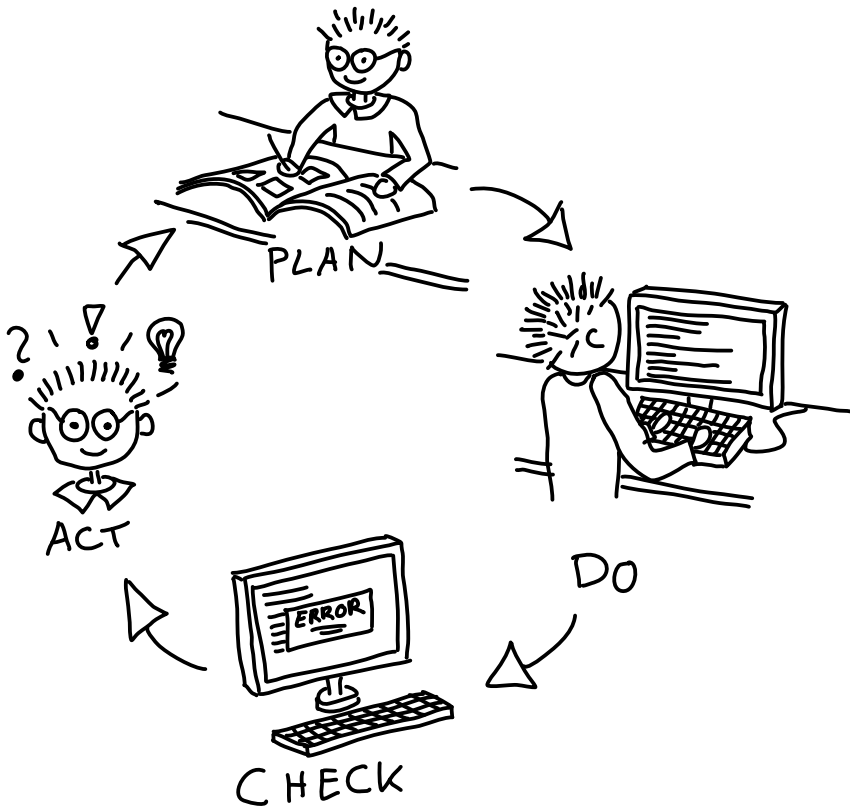
be a bug in the software, a hardware configuration, or both, or even something completely different. There are countless chances to fail. Key is to know exactly what failed and how the team solved it. A root cause analysis (RCA) has to be made for everything that went wrong, or a post-mortem as some call it. (Personally, I hate this eery term, but my teams won't stop using it.) What's important is that the team evaluates the process of dealing with an issue.

Discuss post-mortems with the team, partly to pinpoint which issues are one-offs and which ones need structural fixing. Structural fixes need to go to the team's backlogs, with the right priority. The team needs to stop "firefighting", therefore structural fixes deserve a place high on the backlogs.

I learned about the importance of quality control in the years I worked in the automotive industry. For these companies the quality of manufacturing is vital. A car is built up of about 30,000 parts. If one of these parts isn't

manufactured well and breaks, this can have dire consequences, ranging from costly recalls to fatal car crashes.

Every single part of a car has to be of very high quality. At the company I worked at, the quality system was based on a simple principle called PDCA: Plan > Do > Check > Act. This is a common guideline in the automotive industry and is visualised as a circle. The idea behind it is that quality can only be achieved by continuous learning. The circle is an effective way of structuring that.



This is how it works:

Plan: Develop a solution you want to make with the whole team.

Do: Execute the solution according to plan and record all outcomes.

Check: Analyse the outcomes and determine success.

Act: Implement improvements and iterate back to “plan” if needed.

When you think about it, PDCA is nothing special, perhaps it is so simple you can't even really call it a quality system. Nevertheless, I was impressed by this process. Not because of its complexity, but because of the devotion with which it was followed by my colleagues. I worked at a manufacturer of navigation systems at the time, and we were collaborating with a large European car manufacturer to develop navigation software for their next range of cars. Their team used the PDCA process and held us to it as well. In the beginning, we weren't charmed by it, because it slowed down our scrum based development process. However the automotive team stubbornly kept

testing our software, and over time we started experiencing the benefits. Every time we thought we had delivered some pretty solid work, they managed to find flaws in it and our first priority would be to fix them. Over time, this resulted in us building a very robust product.

I believe this PDCA system is a powerful tool, especially when your team is in “firefighting mode”. It gives structure to your actions and it is helpful for you and your team to experience that if you relentlessly keep fixing faults and learning from them - painful as it may be - eventually the quality of the platform will improve and you will get out of the stormy weather.

Through the fire and flames

The period Anneke is describing, is one all too familiar to me. With my history in development I've seen many outages and this one was not different from the many others I've witnessed and solved. An outage can hurt the business in many ways. It's key however to make sure everybody that needs to feel the pain, feels it where it should be felt, and prevent the feeling from becoming diffuse causing a blaming culture. IT is intrinsically complex, and not everybody is inclined to, or even dares, to understand it. The more you know, the more you realise you don't know, which can lead to a blinding downfall on the spiral of self-worth and resistance or reluctance.¹

My hands were itching to help out.
I saw so many opportunities to

solve the issue at hand, as well as
improve the overall way of working

1. Kruger, Justin & Dunning, David, Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments (Journal of Personality and Social Psychology, 1999)