Euphoria turns into disillusionment

Why you will fail with Scrum

by Marc Bleß

At the beginning of your agile journey, management will eventually be ready to say enthusiastically: “Yes, we absolutely have to do this!” The euphoria is great, and the expectations for agility are at least as great. However, if we fast-forward a few months, the probability of not feeling much of the initial euphoria is relatively high. Instead, disillusionment exists, and the promises of agile methods have been shattered. You will then hear statements such as:

- "Agile has only made things worse."
- "We’ve already tried Agile; it didn’t work."
- "Scrum confronted us with problems we didn’t have before."

The word "agile" has been burned in your company for the time being, and agile methods are no longer considered as a possible, better way.

The reasons for the failure of agile methods can be manifold. This article highlights typical symptoms of why agile methods fail in practice and gives recommendations for countermeasures you can take.

The cycle of failure

The reason for the failure of agile methods as well as many change processes lies in behavior that prevents the elimination of the actual causes of the problem. Organizations are usually very good at identifying existing problems because they are easily observed and can be identified by behavioral patterns and conditions.

But many companies, however, do not take the next step in identifying the causes of the problem. The typical barriers lie in fear of too much transparency and are often expressed in statements such as:

- "Yes, that's correct, but we can't communicate that here."
- "We've already got enough problems; we don't want to get worked up over this too."

Such barriers can quickly initiate a cycle in which the same problems or symptoms can repeatedly be observed, but their permanent elimination is ultimately unsuccessful (see Fig. 1).

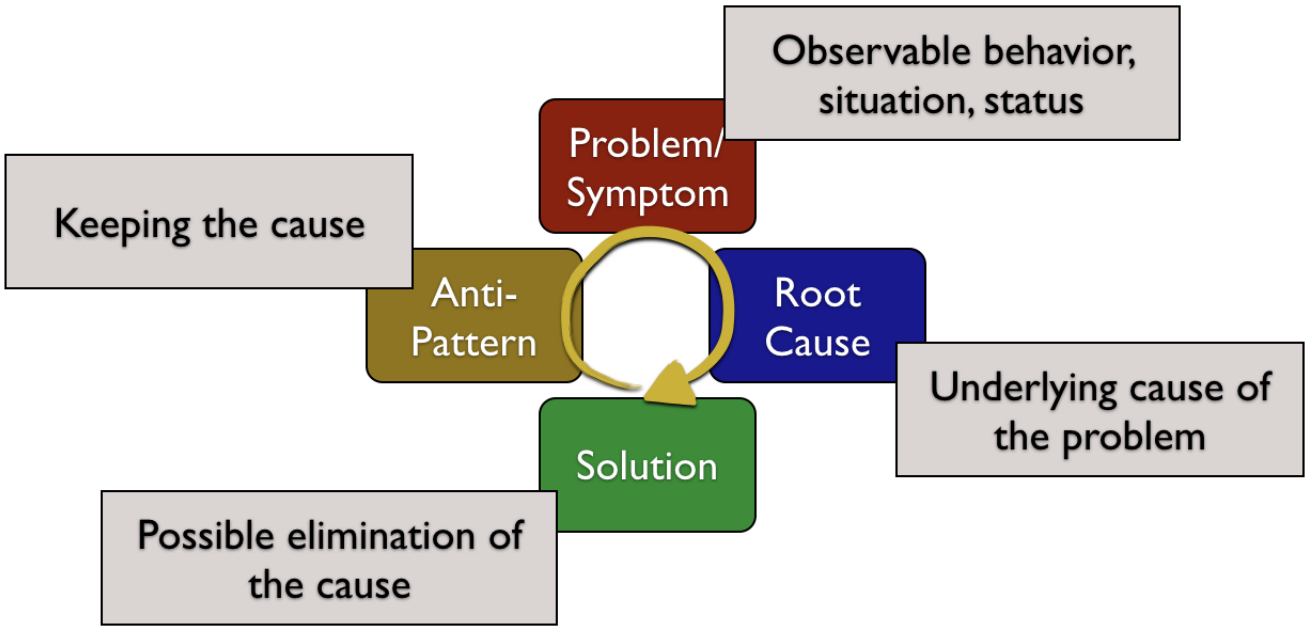


Figure 1: The cycle of failure.

Fortunately, you are still at the beginning of your agile introduction, and the euphoria is still high. This means, for example, that you not only have the right tools at your fingertips to get to the bottom of problems through retrospectives, but you can also determine the proper measures to eliminate these causes.

However, if you are unlucky, you are one of the organizations in which these measures are briefly made, heard and perhaps even understood transparently – but unfortunately, never have a chance of implementing them in the long term.

Four typical symptoms

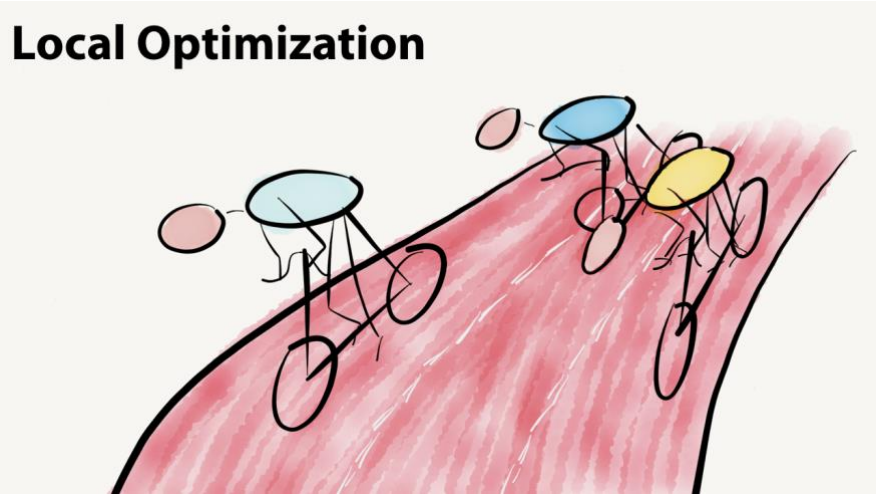
Let's first take a look at the most typical symptoms and their causes that lead to the failure of agile methods, and in this context also shed light on the resulting anti-patterns (i. e. the "bad solution approached" or arguments). We then focus on countermeasures. The symptoms in detail are:

- Local instead of global optimization
- Ineffective retrospectives
- Deficient error culture
- No management support

Local optimization

The problem of local optimization can be identified as follows: Every single team and every department tries to achieve the optimal result. In doing so, they often completely ignore which overarching goals are partially

threatened by this. In the end, no one knows about each other and everyone is wondering why the project deadline has to be postponed again and again and why results never get ready in time.



Cause: Individual goals

The reason for this behavior is that teams and departments are measured with independent team and department goals. Each department and each division is responsible only for the result it is responsible for. Developers don't care about testing, testers have nothing to do with requirements assessment, and product management will never talk to the developers again after unloading requirements.

The typical solutions a team can come up with in retrospectives are usually as follows:

- Establish a holistic view of project and product development
- Draw up value stream analyses with all parties involved
- Set up workflow visualization á la Kanban with all parties involved
- Cancel local goals without substitution and establish a comprehensive backlog

Anti-patterns

These solutions often lead to the following anti-patterns in the organization:

- **Lack of trust**: "It is not only the company that needs holistic goals, but they also have to be broken down for our department in the same holistic perspective. If we don't have them, everyone will do what they want, and nothing will be achieved in the end."
- **Search for someone to blame**: "If the responsibility for theme X is spread across many departments, no one is responsible in the end."
- **Efficiency instead of effectiveness**: "It is not efficient at all for each area to deal with other issues that do not fall within its area of expertise."

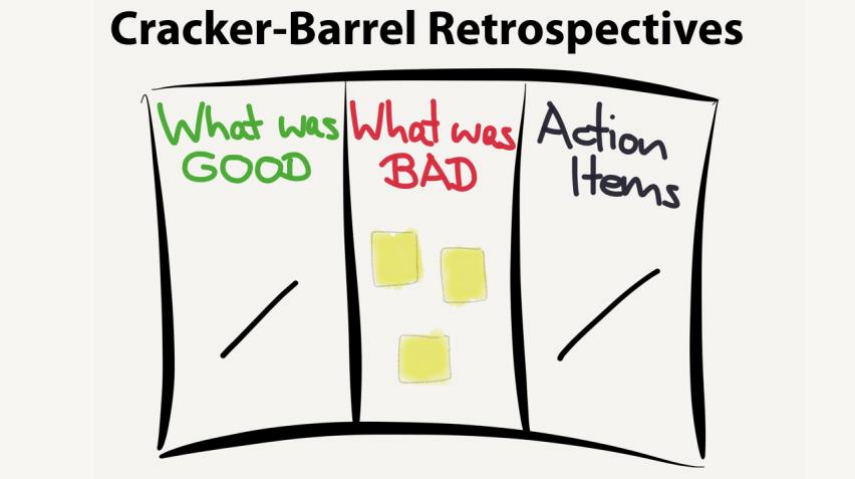
Ineffective retrospective

The symptom of ineffective retrospectives, also known as "cracker-barrel retrospectives," is as follows: In the retrospectives, the participants repeatedly talk about the same problems without there being any fundamental change. Specific actions to implement improvements do not follow the retrospective.

Typical examples and indications of this are:

- the remark "somebody should" is made again and again
- the team repeatedly talks about the unfavorable team/room situation or the lack of an air conditioning system

The team always agrees that only "the ones above" can change it and "we as a team can't do anything about it." However, the goal in the retrospective must always be to decide on measures that can be carried out by the team itself or that the Scrum Master takes into the organization.



A team that moderates poorly and/or carries out unsuccessful retrospectives will feel the positive effects of Scrum only marginally, as it is not able to reflect on its own behavior and adapt or improve it accordingly. Over time, such teams resign from the agile methodology.

Cause: Insufficient training of the Scrum Master

The cause of this ineffectiveness is often the person responsible for carrying out the retrospective. Usually, the Scrum Master has not received proper training: He lacks the basic skills to carry out retrospectives effectively and to sensitize team members to the importance of improvements. In the aftermath of the retrospective, the agreed action items often remain untouched. This means that continuous improvement never takes place to a noticeable extent.

The elimination of this cause is as simple as it is plausible. Investments in the respective employees must be made through training or further education. An external moderator or facilitator can also be hired to provide temporary support.

Anti-patterns

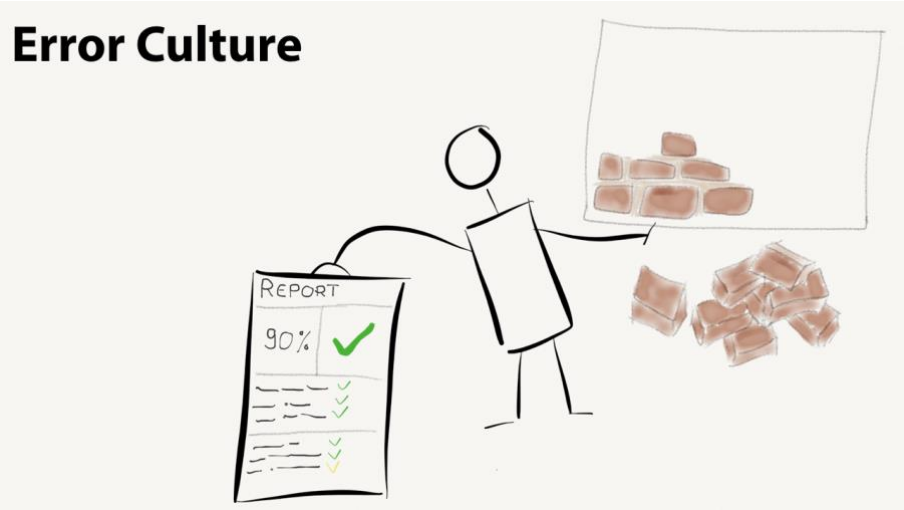
There is exactly one anti-pattern that works against these solutions: **Saving on the wrong things,** reducing them to **efficiency instead of effectiveness**. This anti-pattern is reflected in the following statements:

- "The Scrum Master certification has cost us 2.000 Euros already, so he has to be able to do it now."
- "I'm not paying my employees to improve their resume so they can find a new job afterward."
- "The daily rates from outside experts are way too high, so I'd rather do it myself."

> A little joke in between regarding the second anti-pattern:
>
> A CFO and CTO meet. The CFO says, "Imagine we would train our employees and then they'd quit." The CTO says, "Imagine we wouldn't train our employees and they stay!"

Error culture

The symptom of a bad error culture is that problems are kept a secret, and the true state of the problem is sugar-coated. The big bang comes at the very end - when it's too late.



Cause: bad leadership behavior

The reason for this behavior further down in the organization lies in the leadership behavior described above. There is no room for mistakes, and the messenger of bad news is not doing very well. Consequently, such organizations never make any official mistakes! Every subordinate has an underlying fear of being pilloried and publicly held accountable for difficulties, problems, and mistakes by his or her superiors.

To eliminate these causes, the company must initiate a change in the management culture and encourage managers to take a positive view of the constructive handling of difficulties and turn it into a culture. They have to admit their own mistakes and thus become role models.

Anti-patterns

The existing, anxiety-driven behaviors are usually so deeply rooted that the following anti-patterns occur:

- **Concealment**: In the following example from the real project life, the classic traffic light colors were explicitly omitted and replaced with a check mark in case things go badly. "We will report the true state of the project from now on. Let's use one check mark for red, two check marks for yellow and three for green."
- **Search for someone to blame**: "It's not my fault, it's the fault of other colleagues, teams or circumstances."

No management support

A typical symptom of the lack of management support becomes evident in organizational problems, which are regularly addressed in agile retrospectives, but still remain and are not eliminated.

Cause: No record of the new culture

The cause of this dysfunction is often the introduction of Scrum "from beneath," i.e. by individual development teams or a development department. Such implementations fail sooner or later if management support is not available. No matter how large the local optimization in development may be - if management does not allow the entire organization to change, such an approach will quickly come up against the organization's glass ceilings.

This has nothing to do with whether the introduction of agile methods was driven by the teams or by management. If the higher-ups expect agile methods to be used below, it will only work if the higher-ups also use these methods.

This indicates the solution: The culture of an organization must be open to the agile value system - or else the use of Scrum is doomed to failure! Top management shapes the culture. For this reason, the agile cultural change must take place at all levels, even more so at the top than at the bottom.

Anti-patterns

These anti-patterns may become visible instead of solving the actual problem:

- **Lack of trust**: "We've already introduced Scrum, we can't make another change in the organization right now; the employees wouldn't be able to cope with that."
- **Efficiency instead of effectiveness**: "Scrum has to be successful in the development teams first, then we can think ahead."
- **Search for someone to blame**: "Management has no problem, nor does the rest of the organization. The software development has to prove that it can actually deliver with Scrum."

Anti-patterns and countermeasures

In summary, three central anti-patterns remain:

- **Efficiency instead of effectiveness**
- **Lack of trust**
- **Search for someone to blame**

We can sometimes look at the right countermeasures in different ways for managers and teams.

It is essential to know that the three anti-patterns are attached to each other. Which means, it is basically possible to develop a separate solution for each anti-pattern. But as we will soon see, the anti-patterns are logically linked with each other. The "Responsibility Process" model provides us with an approach to the "search for someone to blame," and also has a positive effect on the other two anti-patterns.

Efficiency instead of effectiveness

The most critical indicator in efficiency-driven organizations is always to get performance as quickly and cheaply as possible. The supplier with the cheapest offer is preferred, even if the own employees say that the cooperation with this supplier is very difficult and the quality deficiencies are serious. Also, the project plan with the unbuffered, optimistic deadline is preferred, even if the own employees say that it is entirely unrealistic and essential risks were not considered in this plan. Fast and cheaply, the magic project management triangle inevitably means that the **quality will be poor**.

Another symptom in an efficiency-driven organization is that it is an impossibility if **resources are inactive**. This means that individual employees should not be temporarily excluded from projects or have nothing to do. Everyone in such organizations must always be busy. Otherwise "people do what they want" and would be paid for doing nothing.

The symptoms mentioned above can all be reduced to a single denominator: **lack of trust** in the own teams. As long as this trust is not given, efficiency will always tend to come before effectiveness.

Lack of trust

According to the "5 Dysfunctions of a Team" model by Patrick Lencioni, trust is the necessary foundation of all cooperation. Only through a mutual trust at all levels can we succeed in building and maintaining a successful organization.

According to Lencioni, the only way to build trust is to overcome our **need for invulnerability**. This need ultimately arises in affected organizations from the culture of having to find a person responsible for every problem. Since no one wants to be held accountable for problems, he/she acts in a way that prevents them from being attacked and thus remains invulnerable.

As a result, management is unable to express confidence in its teams, as the problems in the existing system end up in the lower or middle management, which is then made accountable from above. So, to protect oneself, the problem can only be blamed on the bottom of the teams.

In turn, the teams themselves cannot put their trust in the management, as the blame for wrong decisions is always dumped on them by the management. The result is a vicious circle of "It's not my fault, so I'm not to blame" and "The others are responsible for it, so I'm not to blame."

As long as we are "searching for someone to blame," we remain vulnerable and are unable to build trust.

Search for someone to blame

In Christopher Avery's "Responsibility Process" model, accusing is one of the lowest levels in which responsibility is replaced by naming others who have caused a situation. The model consists of the following steps:

1. Denial – The existence of a problem is ignored
2. Lay Blame – Blaming others for causing a problem
3. Justification – Finding excuses or blaming the surrounding system for causing a problem
4. Shame – Blaming yourself / feeling guilty
5. Obligation - Doing something because it must be done and not because you want to do it
6. Quit – Fleeing / inner dismissal / giving up to avoid shame and obligation
7. Responsibility – Having the ability and power to create and choose for yourself

Further information on the "Responsibility Process" can be found at www.christopheravery.com/responsibility-process as well as in a video by Christopher Avery on YouTube.

The assumption of responsibility as the highest asset

It is only at the highest level of responsibility that we can shape our own lives and environments. The lower down our behaviors and thought patterns are in this model, the less responsible we feel for the things that surround us.

As a team, it is often the fault of other teams, departments or management (lay blame), or the existing technical systems do not allow proper work (justification). As managers, we are bound by corporate compliance and cannot do what we actually want (justification), or we have to implement things in the way that the works council, the supervisory board, or the shareholder's demand (lay blame).

The measure for almost every team, every manager, and every organization is to gain knowledge of this responsibility model and perceive the behaviors of the individual levels themselves. This ability to perceive gradually leads to a **reflective thought process**, which must be promoted throughout the entire organization. Destructive behavior can be addressed openly and constructively resolved in the resulting discussion. With this, the first step of a common, new culture has been taken, however, it must be promoted continuously. Since the overarching organizational culture is always shaped from above, top management is obliged to define this path and set an excellent example in their role as managers.

No manager can rationally deny the goal of creating an organizational culture in which every employee carries an overall responsibility.

Example

Together with a team, I worked out the "Responsibility Process" model in a two-hour workshop with sticky-notes on a whiteboard and discussed the individual responsibility levels of the model (Fig. 2). This exercise was all about raising the team's awareness of the topic.

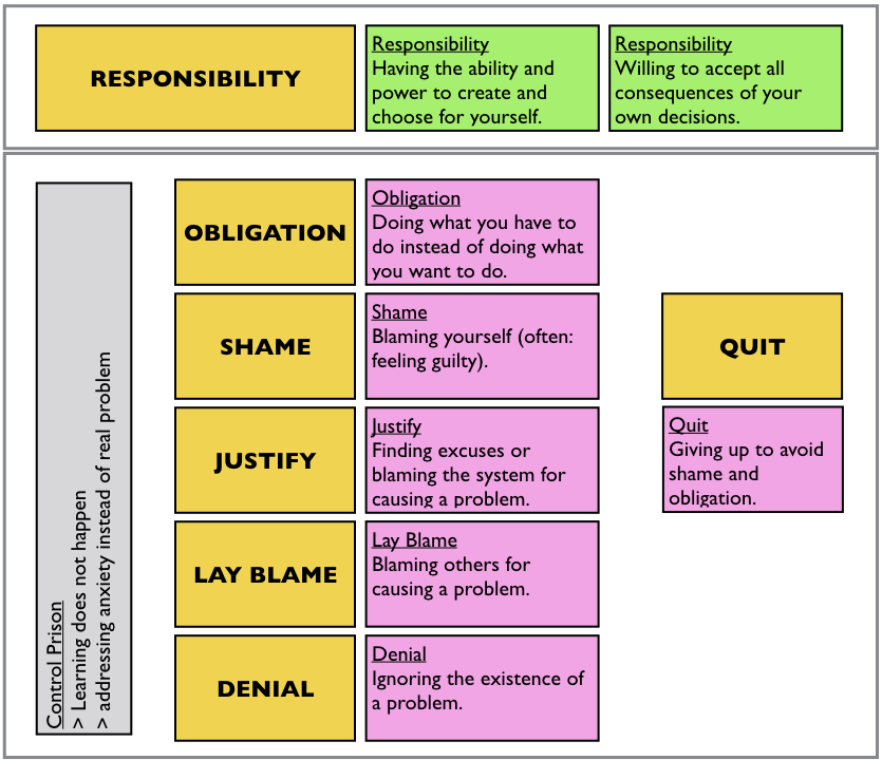


Figure 2: Responsibility levels of the "Responsibility Process" model.

I always start with the visualization of the model from bottom to top. After a short introduction to the workshop topic ("Today I would like to discuss with you the question of what responsibility is and what forms of responsibility there are"), I start setting up the whiteboard in this order:

- Denial – At this stage, we are at early kindergarten age and simply claim that an obvious problem does not exist at all. What does not exist cannot be my problem, and that is why I have no responsibility. We rarely find such behavior in the work environment, but it does happen occasionally.
- Lay Blame – We search for the cause of a problem at a higher level with other persons or a group of people (team, other department, external service provider, etc.). I will name a guilty party, and thus I am not at fault. If I can't find someone to blame, I'll probably find someone on the next level.
- Justify – Perhaps the given system or current methodology is responsible for a problem. Or "things are just the way they are, there's nothing we can do."

Now I let the team discuss the last two levels. The team members should find out what specific observations they could make in their environment at these levels. The team quickly discovers that these two levels often form a cycle and the communication between employees or departments goes around in circles.

This is also how I lead the team through the next steps to the actual responsibility. The discussion that develops during this process deepens the model in people's minds so that the team is sensitized to reflect on their own behavior quickly.

Only a few days later, the team began to critically question itself in the Scrum meetings. Suddenly, difficulties were no longer argued, but one of the team members quickly asked: "Where are we with this discussion at the moment? Are we still laying blame or are we already justifying?" This has immediately put the focus on the team's responsibility and led to the search for constructive solutions.

Conclusion: Agile doesn't work without responsibility

There are many reasons why agile methods fail. However, a closer examination reveals that the fundamental cause of all described anti-patterns is the lack of responsibility. This applies equally to employees, managers, teams and the entire organization.

Being aware of the higher responsibility in the agile context is an important milestone for the successful introduction of agile methods. If the anti-patterns mentioned above can be observed in practice, the "Responsibility Process" model can be used at every level of the project to sensitize an individual as well as a team to the critical issue of responsibility.