You Won't Change Your Organization Without An Optimization Goal

A distinguishing feature of Craig Larman's work (e.g. *Scaling Lean & Agile Development*) is the explicit focus on a system optimization goal for a change initiative. Here's an example system optimization goal that we consider consistent with Agility:

> Increase an organization's ability to respond to change.

Craig Larman clarifies:

> *tis true that the org design from LeSS is consistent with broad adaptiveness, but… suggest that we coach that adaptiveness is not for its own sake. rather, adaptiveness is in the service of something else: learning towards the discovery and delivery of high value or high impact, in the context of a world in which it's usually difficult to a priori know for sure what is high value/impact, and a world in which competition does stuff, new technologies emerge, new trends emerge, etc.*

So, less succinctly:

> Increase an organization's ability to discover and deliver the highest value in a world where we don't know everything, and everything's changing.

I'm not seeing any consistent optimization goal in SAFe, Scrum@Scale, the "Spotify model" and the way some people explain basic Scrum.

Why not just "Make Everything Better"?

Better in some ways is *worse* in others. For example, the goal of *increased customer satisfaction* could be inconsistent with *increased stock price this quarter*. Or another example, I heard a project manager turned Scrum trainer say "In my experience, Scrum of Scrums works great!" And I can see how that could be true, if we're optimizing for the sort of problems project managers are usually asked to solve. But if we're doing Scrum to increase agility, we'll want to consider some better alternatives.

What happens without an optimization goal?

I spent a little some time with a company which I initially thought was a perfect candidate for an Agile adoption. They had less than 100 people in the company, all co-located on the same floor of their hip, modern office. Their management initially seemed quite gung ho. But as the discussions progressed, it became more clear that this management did not *want* to untangle the byzantine organizational structure and the overspecialization that had built up over the past 20 years, didn't think people could learn new skills, and really felt it was best to micromanage employees.

When the company attempted an Overall Retrospective (https://less.works/less/framework/overall-retrospective.html), their actions were to *increase* the organizational complexity that was at the root of their problems! Local optimization bias is so powerful that doing retrospectives blindly can actually make things worse if we are not clear about our optimization goal.

At another similarly-sized company I worked with, the CEO himself came to our mob programming training sessions to see the company's code for himself, and suggest ways of adding automated tests. (This is similar to Ahmad Fahmy's Gemba Sprint (https://www.infoq.com/articles/guide-gemba-sprint/)) This sent everyone a clear message that it's often appropriate to *stop and fix*, rather than continuing to add bugs by churning out crap code.

If management cannot express a clear optimization goal and act consistently with it, perhaps we're dealing with too low a level of management.

Does the change initiative have consistent goals?

Here's a list of other optimization goals[1] that may exist in your organization. They may be explicit or implicit. Some may be consistent with each other in your situation, and others may not:

- increased release frequency
- fewer defects in each release
- predictability
- clarity about who does what
- resource utilization (keeping people busy)
- ideation (creating ideas)
- typing code faster
- secrecy[2]
- comfort

- employee satisfaction/retention (break this down into different categories of employees: coders, line managers, department heads, etc.)
- customer satisfaction
- appearance that the change initiative has succeeded
- increase top-line revenue
- maintaining lead or monopoly
- reduce cost per developer
- product versatility
- broaden/diversify customer base
- preserve status quo
- compliance with rules/regulations
- lead the market by disruption
- increase return on investment
- survival
- conformance to a plan
- rate of developing features immediately
- rate of developing features within this quarter
- rate of developing features next year

1. Adapted from a list Viktor Grgic assembled. ↩

2. Yes, I have seen this as an *implicit* goal that was surfaced during training. ↩