
Felipe Castro
STOP
WATERFALL
GOALS
Using OKR to Focus on Value
Instead of Features

Table of contents

The Feature Factory

Delivery Agile

Waterfall Goals

Full-Stack Agile

Creating Value-Driven Teams

From Opinions to Data

Enabling Autonomy

The Feature Factory

Agile was created to deliver software. It was conceived as an alternative to waterfall development for managing software projects. As such, it focuses on managing deliverables (user stories, features, and epics) and not actual value (business or customer outcomes).

> *In fact, there is not a single ceremony in Agile for tracking results.*

The Agile Manifesto itself is misleading as it explicitly tells people to measure themselves based on deliverables. "*Working software is the primary measure of progress*," the seventh principle states.

Implicit in the Manifesto is the assumption that all working software is valuable – which is obviously wrong. Some projects will deliver value and others won't. Some features will be adopted by users while others will fail.

Most organizations are stuck in the "feature factory" model, where the teams have no focus on delivering value. Developers are "just sitting in the factory, cranking out features, and sending them down the line," as John Cutler described.

Marty Cagan highlighted the dire consequences of feature factories:

> The teams are just there to flesh out the details, code and test, with little understanding of the bigger context, and even less belief that these are in fact the right solutions.
>
> Teams today are all too often feature factories, with little regard for whether or not the features actually solve the underlying business problems. Progress is measured by output and not outcome.

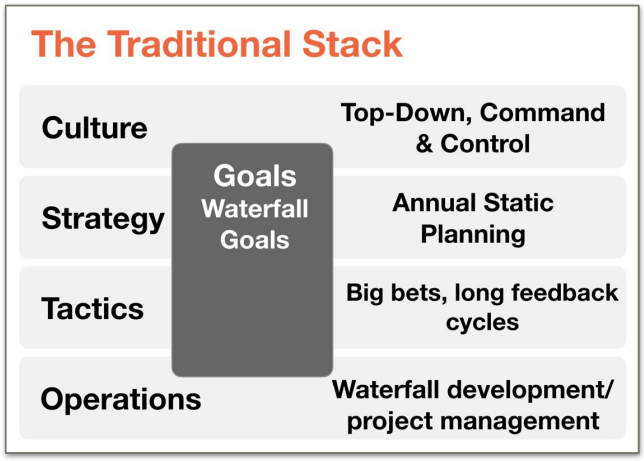
More than 15 years after the manifesto, most companies still use Agile for delivery only. Scaling frameworks usually do not help, as they take the path of least resistance and concentrate on improving software development. As such, very few organizations ever achieve business agility.

Delivery Agile

The best way to understand this issue is to think of organizations as stacks composed of different layers: Culture, Strategy, Tactics, Operations, and Goals.

Goals permeate all other layers, as they reflect how the company works and behaves.

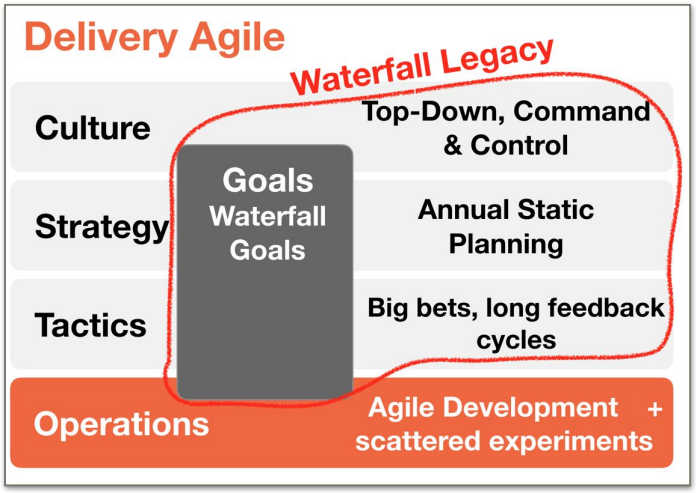
The organizational stack for traditional companies is depicted below:



In the traditional stack the layers are:

1. Culture is top-down and command-and-control.
2. Strategy uses annual static plans.
3. Goals follow a waterfall approach.
4. Tactics work with big bets and long feedback cycles.
5. Operations use waterfall development and project management.

In most cases, when companies adopt Agile, they follow the Delivery Agile approach, which merely replaces the bottom layer of the traditional organizational stack:



Delivery Agile utilizes Lean and Agile at the operational layer only. Agile displaces waterfall development while the teams run scattered experiments. The experimentation culture is not present and, although a few A/B tests occur here and there, many high-risk assumptions remain untested.

Since the other layers remain unchanged in Delivery Agile, its benefits are limited by a waterfall legacy that is in direct conflict with organizational agility.

Waterfall goals

When it comes to setting goals, the waterfall command-and-control mindset is still the norm: organizations use an annual, top-down process to create a set of static goals that is in direct conflict with being agile.

Waterfall Goals follow a static planning model. Usually, it starts with a corporate retreat where the "strategic thinking" happens and the company goals for the year are defined. Then over the next weeks or months, goals *cascade* through the organization, creating a fixed detailed plan for the year for the company.

> *Can you think of a more top-down waterfall analogy than a cascade?*

The static model carries several assumptions:

1. All steps of the plan can be defined in detail in advance;
2. The vast majority of the plan will be correct;
3. Market conditions will remain mostly the same;
4. Changes will be small and may be dealt with in a review in the middle of the year - where an updated detailed static plan will be created.

Waterfall Goals are project-based

To make things worse, instead of being focused on delivering value, waterfall goals revolve around a set of projects approved by management.

In the Taylorist approach to Agile, teams exist to deliver projects and features planned by the executives, in true feature factory fashion. This waterfall planning model slows companies down, makes it harder for them to adapt, and increases risk and waste.

Most, if not all, scaling frameworks work within Delivery Agile. They are sophisticated approaches focused on using Agile to deliver waterfall plans.

From Static to Dynamic Planning

The followers of the static model behave like the Kremlin's central planners that created top-down 5-year plans believing they could predict the future.

In contrast, dynamic planning embraces change and works in smaller planning cycles in an iterative model. Dynamic planning assumes that market conditions and the plan itself will change. More than that, our understanding of the problem will evolve as we learn, and our plan has to reflect that.

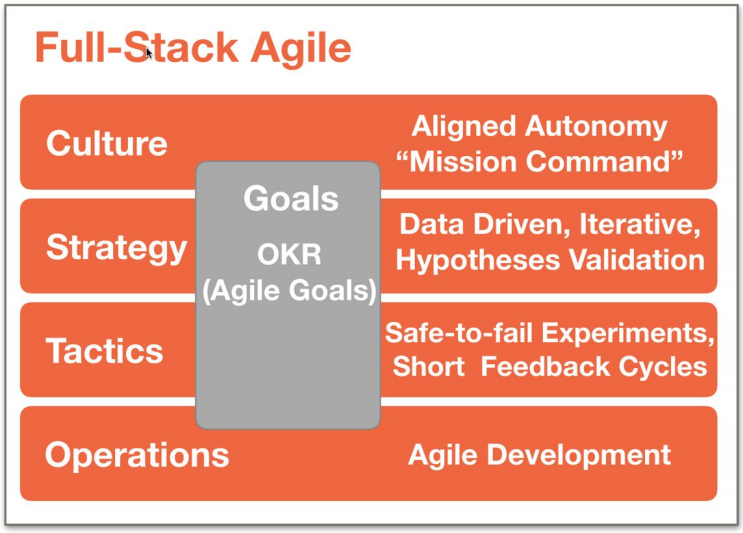
As Kent Beck wrote, "The only way it's all going to go according to plan is if you don't learn anything."

If you want teams to work in short iterations and test hypotheses, how can you use a static set of goals defined in an annual waterfall process that could have been devised by the Kremlin?

You can't. So although we have been using Agile for delivery, we are still using waterfall mindset and processes for everything else. This needs to change.

Full-Stack Agile

To reach business agility, companies have to be Full-Stack Agile, a model that replaces all the layers of the traditional organizational stack:



In Full-Stack Agile, the layers change:

1. Culture is based in creating aligned autonomy with the teams. Instead of controlling detailed plans, it follows the principle of mission, where leaders "specify the end state, its purpose and the minimum possible constraints."
2. Strategy is data driven, iterative and focuses on validating hypotheses.
3. Goals follow an Agile model, using OKR (Objectives and Key Results).
4. Tactics run safe-to-fail experiments with short feedback cycles.
5. Operations use Agile development.

The waterfall legacy that permeates Delivery Agile creates a second type of debt: *organizational debt.* And just like its technical sibling, it makes companies harder to change and is paid with interest.

To become Full-Stack Agile, companies have to fix the organizational debt by refactoring all the layers of the stack. But that is easier said than done, as many have tried and failed. What would be the best approach?

From “Mindset” to “Plumbing”

A lot of people in the Agile community believe that the only solution is to focus on a mindset shift. It seems as if we could just change the mindset of the organization, all the problems would go away. In fact, several luminaries wore a t-shirt that read “Agile is Mindset” during a major Agile conference.

Focusing on mindset change alone can be harmful, as it is not actionable. “'Mindset' seems to be replacing 'values' and 'mission' as the latest action avoiding platitude,” Wrote Dave Snowden, creator of the Cynefin framework.

The alternative is to focus on practical actions that can change how organizations behave. As Stanford Professor James March reminds us, “Leadership involves plumbing as well as poetry.” While the poetry aspect is important, most organizations forget to renovate the plumbing: their underlying systems and processes. Changing the plumbing is often messy and takes time, but it pays for itself.

There is one actionable tool for business agility that can change the “organizational plumbing.” The tool is OKR (Objectives and Key Results), the goal setting framework used by firms like Intel, Google, and Spotify.

The big difference from traditional planning methods? OKRs are set and evaluated frequently - typically quarterly. Furthermore, rather than being cascaded down the organization, OKR is bidirectional: teams create most of their OKRs in alignment with the company goals and then hire them with the managers in a bubble-up approach.

This approach provides a much more engaging environment for teams, who now feel responsible and accountable for the goals they help set, which they track on a fast weekly cycle.

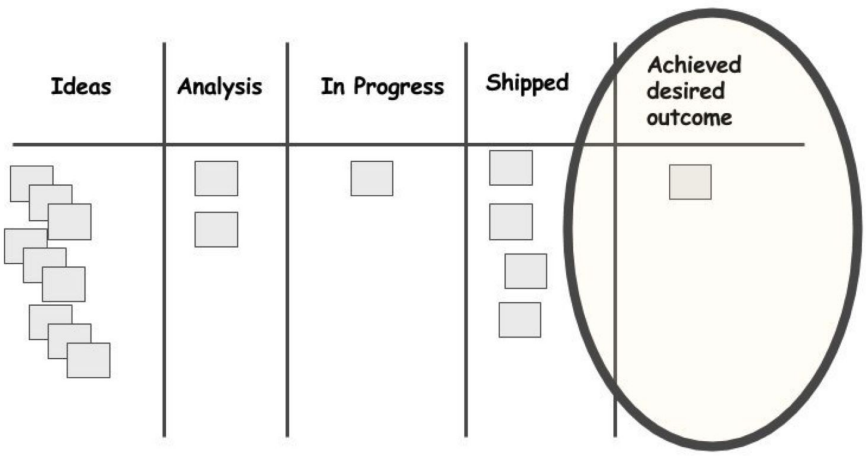
If used correctly, OKR can enable organizations to become Full-Stack Agile.

Creating Value-Driven Teams

The key to becoming Full Stack Agile is to focus on value. The challenge is that the whole system has been optimized since the Industrial Revolution to deliver tasks planned by the executives. Unfortunately, Agile was also optimized for delivery, giving birth to the feature factory model.

This obsession with delivery runs deep. It starts with the use of working software as a measure of progress and continues today. Scrum is, after all, "the art of doing twice the work in half the time," as the title of Jeff Sutherland's book states.

The problem is there is a column missing in Kanban/Scrum boards: "Did it work?", as this provocative illustration from John Cutler shows:



"Done" only matters if it adds value.

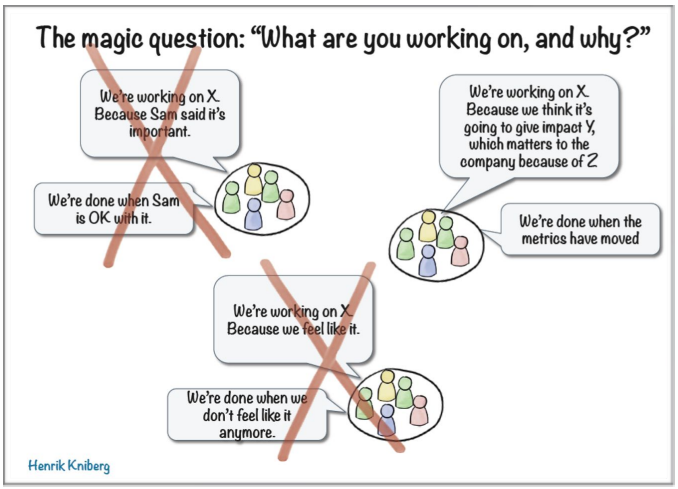
In fact, delivering features that do not positively affect the selected metrics (called Key Results in OKR parlance) may generate negative returns. The new code may have bugs, will have to be maintained and the product itself will become more complicated.

Although the wording of the Manifesto is misleading, some of its authors have written about the need to focus on outcomes:

> The key to [defeating] waterfall is to realize that agilists value Outcomes over Features. The feature list is a valuable tool, but it's a means not an end. What really matters is the overall outcome, which I think of as value to the customers.
> - Martin Fowler

Why are you working on that?

Henrik Kniberg has a great slide about what drives each team:



The first option represents the feature factory. Its underlying assumption is that the team is incapable of deciding what to build, so they work on things because somebody told them to ("Sam"). This approach is based on the taylorist principle of separation between planning and doing, which is both demotivating and incapable of driving innovation.

The second approach is the other extreme, where teams work on things for no other reason than "they feel like it."

The third option is the *Value-Driven Team*. A team that is focused on delivering value and understands how they can make an impact. They have a clear purpose and a line of sight between their work and the company strategy.

The Proper Way to Use OKR

Like any other tool, OKR can be misused and become a to do list. However, if you want to focus on value, your goals have to reflect that. You have to use Value-based Key Results:

Value-based Key Results

Measure the delivery of value to the customer or the organization.

Value is like a joke: you don't get to tell the other party if it's good or not.

Value-based OKRs are not about simply measuring outcomes. You have to understand what is valuable to your customers and your organization.

The examples below show the difference between the two types of Key Results:

Activity-based Key Results	Value-based Key Results
Develop 3 new landing pages	• Generate 100 Marketing Qualified Leads. • Increase lead conversion from 5% to 8%. • Reduce Customer Acquisition Cost from $25 to $5.
Launch new product	• Reach 500.000 Daily Active Users of the free version. • Achieve 5% conversion rate from free to paid users. • Achieve a Net Promoter Score of 35%.

Using Value-based Key Results can be transformative, as it can be the missing link between Agile and Lean, bridging the gap between product and engineering. On the other hand, when you use Agile with Activity-based Key Results, it creates friction since agile teams already have roadmaps. So why do they need OKRs?

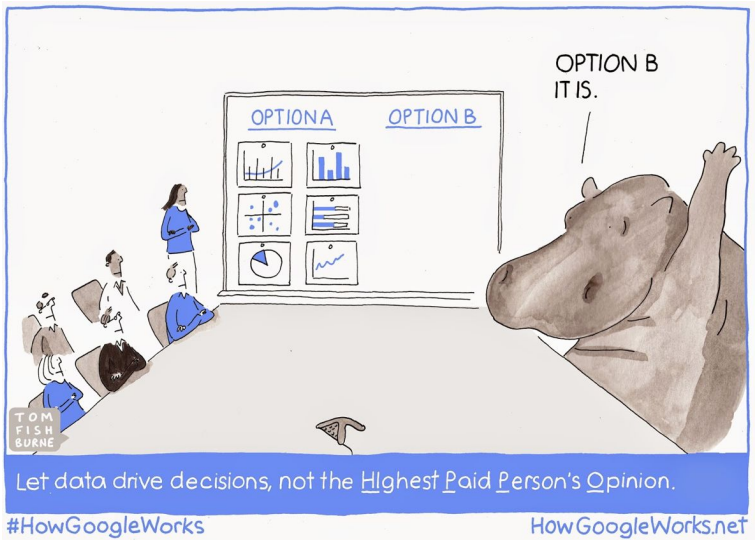
Every time I meet someone that does not understand how to connect OKR and Agile, it's because they are focusing on activities instead of value.

Changing the Language

Even the nomenclature that Agile uses focus on delivery. We need to abandon task-based concepts such as the definition of done, burn-down charts, and velocity, to focus on value. Instead of acceptance criteria, we need success criteria - with OKR.

From Opinions to Data

Instead of being data-driven, standalone Agile is driven by the HiPPO: the *Highest Paid Person's Opinion*, as brilliantly illustrated in the book How Google Works:



There is a flawed assumption behind Agile. The whole model is based on the stakeholders telling the teams what needs to be done and then reviewing the work afterward.

Ron Jeffries described the approach through a hypothetical conversation with a stakeholder (emphasis mine):

"Every week **you get to tell us** what's most important to you, and we'll tell you what we think we can accomplish. A week later, you have it in your hands. [...] **You could ship it out if you want to.**"

The stakeholders decide what to do and if work should be shipped, following the taylorist model. The underlying assumption is that the stakeholders know what is valuable and their opinion is a measure of actual value.

But data shows otherwise. For example, a paper published by Ron Kohavi, Microsoft's General Manager for Analysis & Experimentation shows that only 1/3 of ideas create a statistically significant positive change in the desired metrics.

Instead of collecting data and measuring what works, Agile is based on asking the HiPPOs what to build. And they have an error margin of 66% or more.

Many companies are still using the "voice of the customer" model, where someone represents the end customer. This model made sense in the past, since collecting data was hard, but nowadays it is just another waterfall residue.

Replacing HiPPOs with Experiments

The fact is that teams don't need someone to be the voice of the customer. They can interview customers and measure behaviors. OKR can replace the HIPPO with measurable experiments that allow the team to learn and iterate.

It enables teams to adopt practices such as Hypothesis-Driven Development, as described by Barry O'Reilly:

> We believe **<this capability>**
> Will result in **<this outcome>**
> We will have confidence to proceed when **<we see a measurable signal>**

In this model, instead of showing deliverables during a sprint review, the team discusses the metrics and lists the main hypotheses to improve them moving forward.

Enabling Autonomy

Command and Control is still here.

The command and control mentality still permeates Delivery Agile, where the stakeholders decide what to build. The teams are working on something "*because Sam said*, "and will stop "*when Sam is OK with it*."

When your team has no voice regarding what to build and does not understand the underlying business problems, you are not getting their full contribution. As Marty Cagan wrote, "If you're just using your engineers to code, you're only getting about half their value."

To enable autonomous self-organizing teams, you need to give them the freedom to decide how to achieve the desired valuable outcomes. The purpose of the team has to change:

The Feature Factory's Purpose	The Value-Driven Team's Purpose
Delivering the features the stakeholders want	Achieving the agreed Value-based OKRs.

As Mary Poppendieck wrote: "Perhaps the biggest shortcoming of agile development practices is the way in which teams decide what to do. [...] for the longest time, answering these questions have not been considered the responsibility of the development team or the DevOps team."

Instead of implementing a waterfall plan conceived by the stakeholders, teams can discover what the customers want using tools such as dual track Agile and design sprints.