

Product Compass



An Adventurous Journey to the Temple of Done



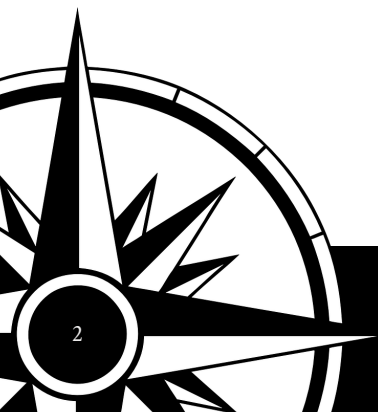
by Mike Veerman




Think About Whether or Not...

Think about whether or not you agree with the following 12 statements.

1. Most companies struggle with building software.
2. The success theatre in magazines and social media is a lie.
3. The large majority of projects go over budget and deliver subpar results.
4. Most people associate software development with frustration.
5. Vendors that tout their 100% success strategy are lying.
6. Even the flashiest agency or the biggest-name consultancy doesn't have a guarantee.
7. Companies underestimate the effect their own organisation has on the end result.
8. The way you approach a software project makes all the difference.
9. Gathering requirements, defining the scope and then implementation is not an ideal approach.
10. You'll meet so many obstacles on the way, making contingency planning futile.
11. You'll get so many new insights on your journey that sticking to the original plan is wasteful.
12. There is no way of knowing upfront if your project will be a success.





“Software is not a rigid step-by-step plan. It’s an adventure.”

Software development is hard.

Sure, writing code and linking together API’s can be frustrating and time-consuming. But that’s the easy part—the real difficulty is deciding what to build.

Successful companies understand that building software is a discovery. That it’s a set of controlled experiments to see what gets you to the desired outcome. Software is not a rigid step-by-step plan. It’s an adventure.

Welcome to your adventure guide!

The Adventure Begins

Building custom software is an expedition into the jungle of an unknown mountain. Such an adventure has risks and often a high cost, but the treasures that await will be worth it.

Showing up at the foot of the mountain and starting an unprepared ascend is foolish. Your chances of getting there are small.

Planning the entire trip is not a smart move either. Nobody has ever been there, so your map is wrong by definition. There might or might not be crocodiles on your track. There is no way to know.

Smart adventurers limit their risks. They gather expertise and make sure their crew is agile and flexible. Some will hire a local guide who knows the mountains a bit. Others might bring a translator who can talk to the jungle tribes. And most of all: they bring a compass.

Most expeditions fail. Not because the crew gets eaten by a mountain lion, but because they get lost. Moving up the mountain sounds easy, but those winding jungle tracks are treacherous. You might run out of food after backtracking yet again. Your crew might despair and quit the expedition. The longer you wander, the worse it will get.

This not-so-subtle metaphor translates directly to software projects. Adventurers show up on “sprint 0” and start climbing the mountain. They’ve hired a crew to help them, but they risk getting lost. There is no way of telling whether they’re hiking in the right direction.

They lack a compass.





Building custom software is always pioneer work. Whatever you build, it's created for the first time. Sure, other companies also have reporting systems. But they have never been tailor-made for your business. You are the first to build this one with this team.

When we select teams or vendors to build our dream, we look for people who have experience in similar projects. That is definitely invaluable. But we should not make the mistake of believing they have the experience to make our exact product. No two jungles are alike.

Yet this is what happens. A team will come in and will estimate the project. It will take 300 man-days. We have Epics and User Stories to describe all the features. There's a Project Manager that keeps an eye on the scope and a bunch of developers that churn out quality code. Let's get that Burn Down Chart going!

It never goes well. The project is late. There's scope creep. Quality is low and users are frustrated.

Why is that? Why doesn't it go as planned?

Expeditions & Camps

We tend to think of software development as a big chunk of work. The “entire scope” of the project. “Phase 3” of the roadmap.

In reality building software consists of many small iterations. That’s how we explore the unknown. We don’t take a direct flight to the Temple of Done. Remember, nobody’s ever been there.

If we can accept that we cannot predict the outcome of building custom software, we are much closer to success.

It’s the basics of Lean and Agile Software Development. Yes, we need a plan, but it’s even more important that we can adapt to new situations and insights.

We’ll launch small expeditions to explore the jungle. If we made a mistake, we’ll backtrack. If not, we’ll set up camp.

The journey is an adventure. Unpredictable things will happen. Each expedition can run into unforeseen lessons. That’s why we want to keep the expeditions small and focused.

Focused means we’re going up the mountain for a specific reason. To do one thing only.

Expeditions are high-energy and at times stressful. We’ll make sure that our team has plenty of rest in between.





**What about small tweaks? And bug fixes?
When should we do these?**

There is downtime between two expeditions. It's the perfect moment to repair some tents, wash clothes and finally fix that one annoying CSS-bug. There's a lot of things team members can do on their own. So, let's give them the time to do that while they are in the camp.

Once the expedition starts, full focus is required. You don't want people cleaning their shoes while exploring that mysterious cave.



Product Compass

What is a compass?

It's a device that helps us navigate. It's a sixth sense that prevents us from getting lost. It's a simple tracker that works even when there is no GPS or internet connection.

A compass consists of three parts.

First, there is the magnetic north. It's huge. It's cosmic and complicated. It's the Big Hairy Goal. The result is there even if we don't fully understand how it works yet. But it works.

Then there is the needle. It's the simplest component but it makes the compass the compass. North might not be our current destination. Sometimes we need to travel West for a while. The needle is the beacon that guides the next leg of our journey.

And then there is the habit of checking the compass. The most delicately crafted compass is worthless if we don't stop to take our bearings from time to time.

Our Product Compass will consist of the same three components. A Big Hairy Goal that we don't fully understand, a simple device that points us in the right direction for now and the habit of checking up from time-to-time.





The North

Before we put on our hiking shoes and pack provisions, let's agree on what we want to find up there. What outcome are we hoping for? Where is our True North? This is the answer to the notoriously difficult question: what does success look like?

Think about one of the projects you're currently working on and say in one sentence: "This will be a success if ...".

For most software, this mission statement will be lacking. Others will have a very vague one.

If you're running a project, your mission statement should be laser-sharp. The sharper the better. Quantify it. Time-box it. Slap deadlines on it and make sure there is no room for interpretation. No "And", "Then" or "Also".

This sounds simple, but it will get brutally hard.

Let's give it a shot with the scenario on the next page.

Here's the scenario: Customers are not happy with our service. A survey has given us some negative feedback and we want to fix that.

VAGUE

“This project will be a success if we increase customer satisfaction.”

That's a very vague one. What customers are we talking about? How can we measure their satisfaction? The survey shows that a lot of customers complain about long delivery times. We want to fix that.



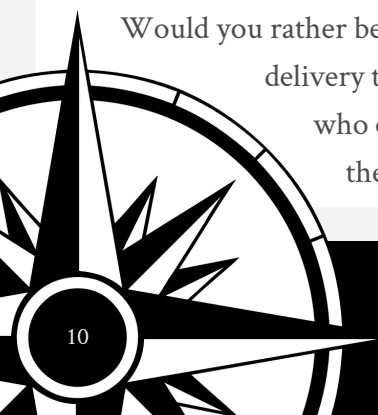
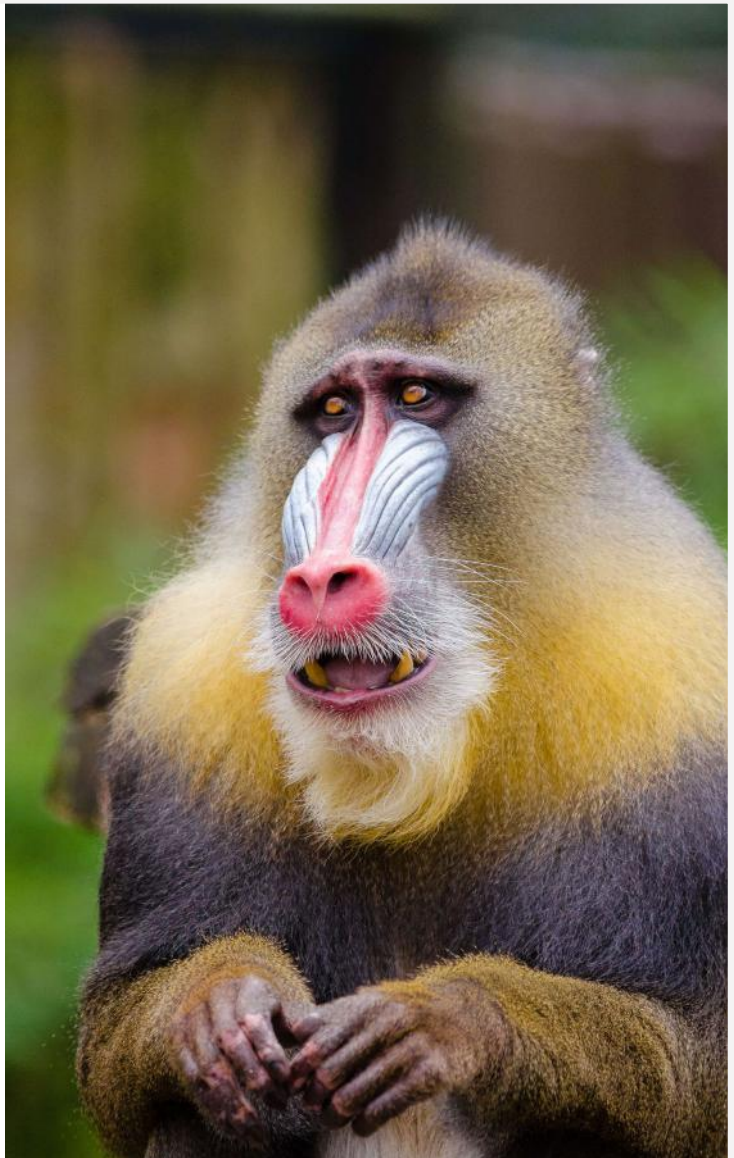
BETTER

“This project will be a success if we lower delivery times.”

That's better. But it's still vague. Let's make it quantifiable. How much? I would advise to keep it conservative. If your software projects are not going the way they should, set yourself up for success by not being overly ambitious.

That doesn't sit well with a big part of business and management people. I get that. Their go-big-or-go-home mentality is part of their charm, but being pragmatic is a superpower here. You can always adjust the metrics upward if you're overshooting your targets. That's a luxury problem.

Would you rather be the one who improved delivery times by 10% or the one who **claimed** they could cut them in half?





GOOD

“This project will be a success if we lower delivery times with 15%.”

That’s much better. We have a clear measurable goal to work towards. Let’s slap a date on it and we’re done.



BEST

“This project will be a success if we lower delivery times with 15% by June 1st”

Perfect!

What about products?

Projects are simple. They have a start date, an end date and a clear goal. They’re time-boxed one-trick-ponies. Products are way more complex. They have a plethora of features. They have no end-date. They have roadmaps. A product is never done. They often involve a collaboration of many teams with their own agendas.

It’s a bit trickier, but we can give clear goals to iterations and treat them as mini-projects.

“This iteration will be a success if we can improve report load times with 50% by November 15th.”

If you’re doing Scrum, you could dedicate sprints to it.

Our expedition has a goal now.

Great! We know now what we want to achieve by being up the mountain.

The next part is clear communication with our team. We want everyone to know what the goal is and to focus on that.

“This project will be a success if we lower delivery times with 15% by June 1st. Everything that doesn’t contribute to this goal is out-of-scope.”

Here, the same people will start to feel queasy. Isn’t that low-balling it a bit? An entire team that does only one thing? Surely, they can squeeze in a few bug fixes and minor tweaks?

That’s what focus means. You can’t have a good dinner conversation and read a book at the same time. If you’re reading, you’re reading. Yet most software development companies are in this multitasking mode. They are doing busy work on many things and produce half-assed results.

People are bad at multitasking. Teams are even worse. A team is not a bunch of people each working on their own tasks. It’s the collaboration part that makes teams great. So if they are not collaborating on the same thing, they can’t get things done.

This part is so crucial and so misunderstood. A group is only a team if they are working towards the same mission statement. The analysts are not preparing for the next expedition while the devs code. They work together.

Focus is doing one thing and ignoring the rest.

I promised you it was going to be brutal, right?





The Needle

We've touched on the two different phases a team goes through: the expedition and the down-time. That's an important mechanism.

There are times when we double down on a singular problem and times when we clean up smaller unrelated items. Most teams don't seem to do that. They seem to be trying to do important stuff while also doing small fixes and tweaks. Most teams never focus.

Now that we know where North is, we need something that keeps us on track. It has to be precise, but it's even more important that it's simple. When we are out in the jungle, we don't want to spend time triangulating our position. We want something quick and error-proof. We prefer ease-of-use over completeness and accuracy.

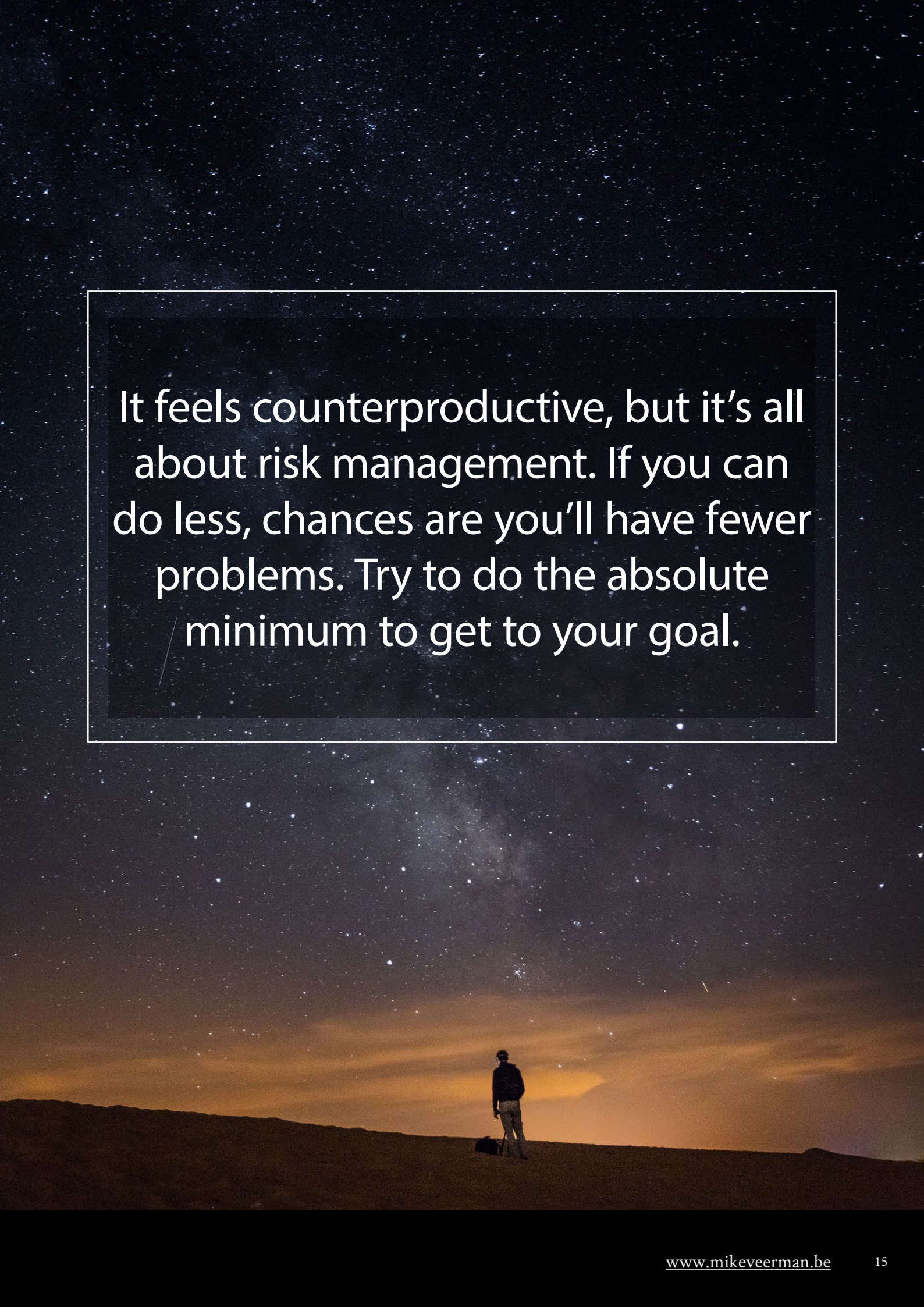
What would a needle look like for our software development compass?

Back to our True North:

“This project will be a success if we lower delivery times with 15% by June 1st. Everything that doesn’t contribute to this goal is out-of-scope.”

Our team gets together and comes up with four ways of lowering the delivery time. Are we going to build all of them? Of course not. We look at our goal and pick one or two low-hanging fruits. If they get us to 15%: mission complete.



A person is standing on a dark, silhouetted dune in a desert landscape at night. The sky is filled with stars, and the Milky Way galaxy is visible as a bright, hazy band of light stretching across the upper half of the frame. The person is wearing a dark jacket and light-colored pants, and is looking towards the horizon. The overall scene is dark and atmospheric, with a focus on the vastness of the night sky.

It feels counterproductive, but it's all about risk management. If you can do less, chances are you'll have fewer problems. Try to do the absolute minimum to get to your goal.

Let's say our orders currently end up in a mailbox.

It can take up to 10 hours before someone even reads it. This seems to be an easy fix. If we can speed up the order registration, we speed up the delivery.

We're going to start out with the first leg of our journey: replacing the current mailbox with a real-time ordering system.

What are we going to build? It's easy to come up with a configurable workflow engine. We need a web app with user management and different roles. An admin can assign tasks to order-takers, who can send the right delivery data to the warehouse and report back to their manager. We need a system that helps us manage the stream from customer contact to dispatch.

Do we?

Scope creep is the business equivalent of overengineering. We're ready to draw wireframes and use cases. We love coming up with fancy solutions. We always over-complicate.

Libraries are written about the Minimal Viable Product (MVP). It's the smallest thing you can build to validate your idea. It's tiny and it invites short-term feedback. We have an idea of an approach that might take us closer to our goal. That idea and verifying that idea only should be the scope of our next expedition.

It's the answer to a simple question: What is the smallest step we can take towards our goal? What is the smallest thing we can build that takes us closer to that 15%?

Adding requirements adds risk to the expedition. Resist the urge.



So now that we know the theme of our next expedition, we're going to spell it out again.

We don't want any discussion while we're in the jungle. It has to be specific and clear.

"In this iteration, we'll build an automated mailbox that processes orders and tries to match them to products. If the system can't match automatically, the mail will be forwarded to the original mailbox. We expect to cover at least 30% of orders this way. We plan to be ready by the 30th of October. This is top priority: the team will refuse any unrelated work except for P1 incidents."

This description has focus. It's short enough to print and hang on a wall but can guide our team.

It has 4 components:



THE SCOPE

The smallest low-risk next step we're taking in the direction of our goal.

THE HYPOTHESIS

At least one hypothesis. It has to make a prediction and be falsifiable. It's how we are going to verify the result of our expedition.

THE DEADLINE

That's the amount of time we are willing to invest in this.

THE SACRIFICE

We spell out the things we are NOT going to do.

This is our needle. It's small and simple. It's not too elaborate and it doesn't leave room for interpretation.



The Habit

Most companies that build software have some kind of governance meeting. This is the ideal moment to check the compass. Whether it's a Sprint Retrospective or a monthly Steering Committee, it's a great tool.

When returning to the base camp, it's a great idea to get the team together and reflect on how the past expedition went.

1. Did we build the scope and only the scope? Did we stick to our sacrifice or did we weasel out of our commitment? If so, why?
2. Is the thing we built testable? Can we verify the hypothesis in a month from now? If not, why not?
3. Did we need more or less time? Was the deadline met?
4. Now that we've built this, are we closer to our goal?

By gathering this iterative feedback after every expedition, we are forced to keep on track.

Question 1 makes sure we call out when the team builds unrelated stuff.

Questions 2 and 3 give us the tools to track our progress towards the end goal.

Question 4 gives us either momentum or a warning sign.





Stick to the Compass

What we want to avoid is panic up the mountain. We want each member of the expedition to know where they're going and make sure they stick to the mission at hand. We don't want people to improvise. That's risky.

Building things takes longer than coming up with ideas. But building the idea is what crystallizes it into a mature feature. There is a real temptation for idea people to run ahead of the team. They'll come up with the plans for the next expedition while the team is still in the jungle.

Don't.

It's the main reason most projects fail.

Running ahead is the enemy. It slows us down and makes everything we do riskier.

It's not smart. We don't take into consideration the things we learned in the jungle. We don't even look at the treasure our team brought back. We keep risking more and more.

It's bad for morale. Our team starts to feel like their input is neglected. They are adventurers, not labourers.

We have hired a team of problem solvers, not code monkeys.



It breaks our focus. Idea people don't just make up stuff. They need to verify their concepts with experts. That gives them a real incentive to shoot quick questions. Have a small estimation workshop. The problem is: our team is in the jungle! While they are on the radio answering your concepts for the next expedition, they are risking the current one! The only right time to plan the next expedition is when the team is back in the base camp.

It slows us down. Short concise steps in the right direction work. Big guesses don't.



Your Own Band of Adventurers

Most companies refuse to accept the uncertainty software projects bring.

They prefer the illusion of predictability and they pretend that's how reality works. They use the word "Agile" but plan to just hike up the mountain in a straight line. When they encounter a lake, they'll swim across. When they meet a tiger, they'll fight it. They waste energy and resources because they religiously hold on to their original plan.

Smart companies know that they can't know upfront. They arrive prepared but flexible. They keep their eye on the prize and are willing to adapt the plan at every turn. They'll walk around the lake. They'll distract the tiger. They'll form a team of adventurers that can apply their skills and creativity to reach the destination.



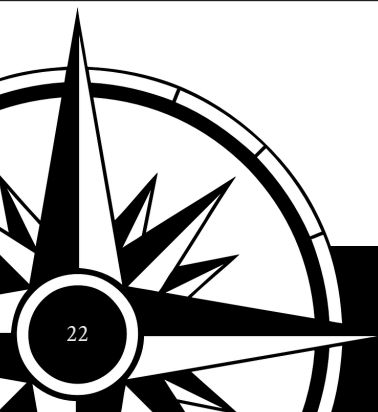


Take a look at your own team.

Are they sticking to the straight-line plan? Or do they have a compass?
Do they continue pushing through obstacles or do they come up with smart alternatives?

Building software is a daunting task. But most of the problems are not technical. It's knowing what to build and when to build it. When software projects go wrong, it's always a matter of organisation.

So be smart. Create your band of adventurers and set them up for success. Let them explore the jungle and they will come back loaded with treasure.





About the Author

Most companies struggle with building software. It's easy to find vendors, but it's much harder to find a guide.

Mike Veerman is a Software Strategist who helps companies get control over their software development. He shapes teams and structures so you can focus and get your products out.

With over a decade of experience in software development, leading teams and managing projects, he can help you ship faster and better.

www.mikeveerman.be

