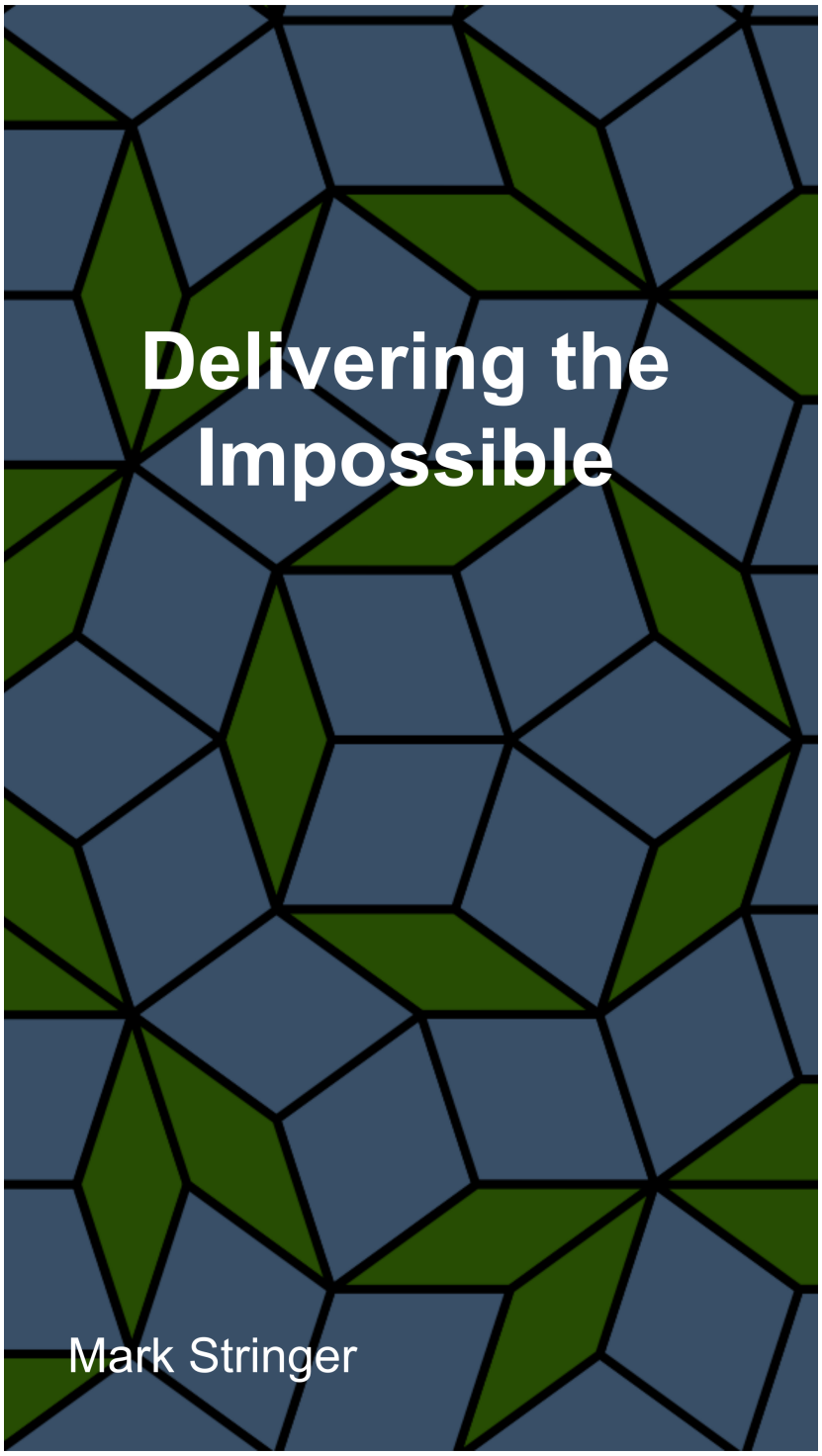
Delivering the Impossible
Mark Stringer

Delivering the Impossible

Mark Stringer

Introduction

This book is called "Delivering the impossible." But of course. The only things that can be delivered are possible things. So I might ask you, and you might ask yourself, Why on earth did you pick up this book? It literally has a contradiction, an impossibility right on the front cover.

But if I did that, I would be being more difficult than you want the author of a book that promises to help with impossible things. Here are some reasons why I think you might pick up this book.

How about this? You are working on a project, maybe you're a member of the project team, maybe you're the manager, and you feel stuck. It seems like the project is impossible. What's being promised can't be delivered, nobody really knows what's being promised, or what's being promised can't be delivered on time.

Every direction that you look in there seem to be problems that can't be solved. You somehow have found this book. You think it might be worth a read.

You have a bunch of people working on a project for you, and you've got this uncomfortable feeling that the project really isn't going well and, for the sake of the company or for the sake of your career, you really need this project to succeed. You'd wonder if there any gems of insight in this book that you could pass on to the team or maybe anything that *you* could do differently.

Even though we think the project is impossible, you haven't just given up. And it's worth asking yourself why? If the project seems so impossible, why hasn't it be stopped, abandoned, killed? Put out of it's misery. Why would anybody bothering carrying on with it?

Well, part of the reason is your inherent optimism. You know that a lot of things that seem impossible, are actually possible, give the right skills and expertise. You know from other parts of your life, that if you can only learn to look at things in the right way and do the right things at the right time, things that seem impossible can magically be made to work out.

So what you would really like and what you might expect to find in a book with a title like "Delivering the Impossible" is some kind of guide, some kind of handbook for dealing with seemingly impossible projects.

In such a book, you might expect to find methods for spotting potential issues that might make a project "impossible" as early as possible. You might then expect the book to go on to talk about what to do in these seemingly impossible situations to make a project reasonable and deliverable and give it a chance of success.

Finally, you might not be surprised if such a book also talked frankly about the things you might look for to satisfy yourself that a project is geninely impossible and gave advice on what to do in such circumstances, to keep ourselves and the

people around us financially, physically and psychologically safe, so that you can walk away from a lost cause and move on to succeed on other projects, ride into the sunset to fight another day.

If you're getting really demanding as a reader, you might also expect the book to talk about how deliver, not just on impossible projects, but also on impossible programmes, even in impossible organisations. Yes, if you're really demanding, you might want such a book to talk about how to deliver the impossible at scale.

Well, this is that book. And it tries to meet all of these expectations. This is especially that book if your project, programme or organisation is in some part dependent on the development of software.

If you read this book, you will definely learn, how to quickly identify seemingly impossible situations, how to manage and manipulate those situations so that's there's a chance, not just of delivery, but also of succees. You'll also get lots of advice about how to look after yourself and those around you while you're doing it.

Agreed activity

One of the great things about living in London is that you can sign up to all sorts of classes. Over the past ten years, I've repeatedly signed up for improvisation classes. Why? Because my experience is that every time I go to an improvisation class, I learn something new, more specifically, I have new experience, and these often somehow or other, turn out to be useful in real life.

One of the key ideas in improvisation is blocking. For example. If an improvised scene starts with someone knocking on a door. The scene might go something like this.

Person 1: [Knocks]

Person 2: [Opens imaginary door]

Person 1: Hello! I've brought you a cabbage.

OK, now we've reached a key stage in the drama. Because Person 2 has lots of options. Person 2 could be really happy that Person 1 had brought them a cabbage.

Person 1: Oh you remembered that this is the week that I make all my Kimchi! Oh darling! You're so thoughtful.

Or they can be really angry that Person 2 has brought a cabbage.

Person 1: And that's dinner is it? You know cabbage doesn't agree with me! Remember what happened last time. You're such an ass-hole Kevin.

The thing about both of these responses is that they move the action forward. They take any initial idea, no matter humble and move it forward.

In improvisation, this is known as “Yes and.” Accepting whatever your partner gives you and amplifying it. The opposite of “Yes and” is called blocking.

So for example, if we go back to what will forever now be known as the “Cabbage scene.” and get Person 1 to knock again on the imaginary door. Person 2 could respond to the “offer” (as it’s called) of the cabbage with some kind of bizarre argument.

Person 1: Hello! I’ve brought you a cabbage.

Person 2: No you haven’t that’s a Pomeranian poodle.

or alternatively:

Person 1: Hello! I’ve brought you a cabbage.

Person 2: Humph. Fine.

Generally in improvisation these kinds of responses are not recommended? Why because they “block” the scene moving forward. They don’t build on it and move it forward. The poodle response is completely trying to negate the “offer” of the cabbage, utterly ruining any chance unfolding of a couple who learn to fly on the wings of their flatulence or another possible story of the dangers of living with Kimchi that becomes sentient.

The second answer “Hmph. Fine.” Is perhaps even more dangerous to a good story because it just goes nowhere, what’s really happening in such a situation is that the person who is speaking is scared. They feel out of control, so they’re doing that absolute minimum possible, even though the result of this is very boring to watch, not very nice to their fellow performers and kind of defeats the whole object of improv.

Here’s another flavour of “Humph, fine.”

Imagine that instead of just two performers on stage for an improvised scene, there are a group, let’s say seven, give or take a few. As is often the way with improvised performances, the actors get a suggestion for a setting for the scene from the audience and get “The deck of a tall sailing ship.”

Here’s what might happen.

Person 1: [Putting mimed fake telescope to their eye] Look over there! On the horizon? There’s a ship, and a flag! Is that a skull and cross bones?

As with the first scene that we talked about, the other performers have a decision to make. One obvious decision here is to totally “yes and” the Pirate ship.

Person 2: Oh my God the Pirates are coming. Haul up the sails, let’s try to outrun him.

Person 3: Oh my God the Pirates are coming. The cannons! Load the cannons!

Person 4: Oh my God the Pirate are coming. Quick hide in the lifeboats.

And all of these are viable selections (providing trying to hide in the life boats is ineffectual).

But what can often happen in such a situation is that instead of saying one of the things above, that might move the story forward, someone says something like this:

Person 5: Hey! Let's scrub the decks!

Persons 6, 7, 8, 9 and 10: Yes! Let's!

For some of the performers on the stage, this can seem like an attractive option. But of course, for the audience, it's a very bad idea. Once the Pirate ship has been mentioned, they want to see the pirate ship arrive. They want to see what a chase between this ship and a pirate ship looks like? They want to see if this crew really can load a cannon, point it in the right direction and fire it. They want to see the pirate king and see what he'll do to the crew members who are hiding in the lifeboats.

They don't want to be treated to the sight of everyone on the stage miming mopping the floor.

But for inexperienced improvisers, the temptation to agree to swab the decks rather than wrestle with the implications of the approaching pirate ship is extremely tempting.

Why? In his books "Impro" and "Impro for storytellers" Keith Johnstone describes this kind behaviour as "agreed activity." Putting people on a stage in front of an audience makes them scared. And scared people try, instinctively to make themselves safe, even if what they're doing individually is making the collective endeavour less likely to succeed. Part of what constitutes the skill of being a good improviser is knowing how to avoid this primitive instinct for safety and instead, move the story forward. Accept that there's a pirate ship on the horizon and deal with the consequences.

What's particularly fascinating about this kind of behaviour is that everybody, I mean all the improvisers on stage, seem to instinctively know that it's the right thing to do. There isn't a long discussion. It happens in a second. And that is in spite the fact that it's exactly the wrong thing to do. What this really shows is how good we are as individuals and in groups at shying away from things that we think might be dangerous, or difficult, or really just anything that will make us having to change our behaviour and our thinking.

One way that Johnstone suggests to help the improvisers avoid "agreed activity" is to have a director who is watching the improvisers and can intervene during live shows. In this kind of set up, it's the job of the job of the director to spot which actions and suggestions by the improvisers will move the story forward - and even in desperate situations - suggest them herself. For example if a director were watching a scene where there'd been a suggestion of scrubbing the decks, she might have allowed the crew 10 seconds of deck scrubbing before shouting

"The deck's clean! The pirate ship is getting nearer and nearer! Deal with the Pirate ship!"

Why am I telling you this? Because of course, this idea of concrete practice has usefulness in the context of project management. As with improvisers on stage, there's a temptation amongst the people who work on a project to find some agreed activity as a way of avoiding having to think, or be changed by the realities of the problem that they're solving.

Of course, if we pick up on what we talked about in the introduction. If we want to deliver the impossible, the first thing that we have to do is stop ignoring the bits that look like they might be impossible.

In iterative, Agile, approaches to project management there are lots of opportunities for the team and the team manager to spot the pirate ship. Typically, every day there's a "stand-up" meeting where the team talk about what they did the day before, what they're going to do today, and, crucially if there's anything blocking their progress.

Of course, it's in the nature of agreed activity that if there's one big thing that's blocking progress, that might be the very things that no-one mentions at all in the stand up meeting. So the signs, that there is a pirate ship that needs tackling might very well not be the sight of a Jolly Roger and the sound of "Arrrrr!" Rather it might well be reports of agreed activity. Very often, the same report with mild variations for days and days.

Another sign of agreed activity and pirate ship avoidance is when a task or "Story" as they're often called in Agile, gets planned and then just doesn't get done. Nobody on the team decides to pick it up. It may well be that no discussion has been had within the team about why they don't want to pick up this story and tackle it. As with the crew on the improvisational stage, they just agreed, possibly tacitly, not to tackle it. Of course, in these situations, it's the job of the project manager to hold up with story and ask what it is about this story that means everyone is avoiding it.

So, one place to spot the pirate ship is stand-up. Another is the retrospective. That's a meeting that happens, typically every two weeks where the team talks about how the previous sprint went. What went well, what didn't go so well, what could be done better.

One of the weird things about project management is that very often everybody who is working on a project knows what's wrong with the project. If they're not saying, it's because they've tried to say or have said and have been either ignored, scolded, threatened or even disciplined for pointing it out. This in improv the equivalent of someone on an improv stage shouting "Look! A pirate ship!" and a director sitting next to the stage shouting "Shut up about the pirate ship! Clean the decks!"

A leader doesn't need to tell the team to shut up about the pirate ships and clean the decks many times before he gets one of the weirdest recurring problems that

I've seen - a highly skilled, highly paid team of professionals who've completely given up trying to think for themselves.

But no matter how many times they've been beaten down. If you run retrospectives and encourage people to speak, listen to what they're telling you and do your best to raise their problems with the people who either need to know about them, or can fix them, sooner or later, they'll mention the pirate ship. And when they do, it might be difficult to keep your jaw off the floor. For example during the retrospective for one project that I worked on, it emerged that all of the requirements for the project were coming from a business analyst. They made no sense. One of the main struggles that the development team had on any day, was making sense of what the business analyst had put in the stories. What struck me straight away was that the business analyst wasn't in the retrospective. In an agile project, the business analyst is part of the team - they're not supposed to be outside throwing in requirements. Especially if those requirements turn out to be problematic.

So my response to this problem being raised by the team was to suggest that they business analyst should be invited to stand-up meetings, and planning meetings. And retrospectives and show and tells. This was where the developers looked at each other sideways - this is always a sign that you're finally getting to the real issue.

"We're not allowed to talk to the business analyst."

"What? That's crazy! I'm sure that can't be right."

Yes it was crazy. Yes, it was right. I tried to talk to the business analyst. I emailed him asking for a meeting. I got a phone call from his boss saying I wasn't allowed to talk to him.

And of course, there was no way that the project was going to succeed until the team were at least allowed to talk to the person who was giving them the details of what they were supposed to do.

Side note: This project was crazier than it even sounds. It turns out there were two groups of business analysts. Technical business analysts (the business analyst that the team wasn't allowed to talk to was a technical business analyst) and then erm, business business analysts - these were the people who actually talked to the business, to the people who actually wanted the software to be built. These two groups of business analysts also weren't allowed to talk to each other - they were only supposed to communicate via emailed documents. A substantial part of making our project possible was to gently insist that the business analyst who was working with us, should sit with our team and be allowed to directly talk to the customer!

So we've talked about stand up meetings and we've talked about retrospectives. There's one more Agile meeting where the "Pirate Ships" that are on the horizon might be detected. And that's the "Show and Tell." The "Show and Tell" is a meeting where the development team show the outputs of what they've been

working on in the last "Sprint" - typically two weeks. Who do they show it to? The product owner who is supposed to act as a representative of all the people who want the project to happen and any other interested stakeholders who want to come along.

After the minor victory of getting the business analyst to sit with the team, I then suggested that we start to have show and tell meetings. Fraud was an issue that affected employees of the bank all over the world. So for the first phone conference I could here just from the accents the geographical spread of interest. Northern Irish accents, Scottish accents, Cockney-sounding Southend accents and Indian accents from the offshore call centres in India.

In that week, the team had been doing some work on the user interface for one of the very early screens in one of the fraud detection journeys.

One of the developers put up the login screen and then clicked through the first screen and started to talk it through. Even though most people who were on the conference call were on mute, I thought I detected some kind of change in the silence. Finally someone on the line with a cockney accent said: "Erm, I though we'd agreed that we were going to arrange cases by account name rather than by account number?" The developer who was demoing the screen looked blank. The business analyst who we'd only just set eyes on and had only just joined the team started to look worried "No, it's in the requirements that cases should be arranged by account number." "But that makes no sense" someone with a Northern Irish accent joined in. "It's people who are victims of fraud, not just individual accounts" chorused in someone with a Glaswegian accent.t "We need to see all the accounts that someone holds, and see the activity across all of them for this to make any sense." said someone with an accent I didn't recognise.

Yes, that's right. The structure of this project was fundamentally wrong. This problem hadn't been detected through months and months of analysis. But, literally five minutes of putting the working software (OK, I'll admit, it was only a front end) in front of the people who might use it had found the problem.

If you do daily stand-ups with your team, if you do retrospectives and show and tells, even if you do these things *badly* - you will still find out what the problems are that are facing your project. Not having stand-ups, not having retrospectives and show and tells is the equivalent of shouting "Scrub the decks, don't look at the pirate ship!" at your team.

So, what does understanding this notion of "agreed activity" mean for our overall aim of delivering the impossible?

Well, it means something very basic. If you work to discover the problems that your project is facing and then work to solve those problems, you may well be able to transform seemingly impossible projects into projects that are not only possible, but actually get delivered.

This seems so obvious as to be laughable. Why then, in project after project

have I found teams that aren't articulating their problems and aren't tackling them? In improv terms rather than looking at the pirate ship and doing what needs to be done when it arrives, they are scrubbing the decks and in the process are making possible projects impossible.

Why? Because problems are scary. Problems are humbling. Problems cause conflict. Professional people who are hired to do a job are supposed to be able to do it aren't they? What does it mean if they openly admit that there are parts of the job that they can't do? Maybe it means that someone hired the wrong people.

Highlighting the problems that a project unearths can be threatening to the sponsors of a project. What if the problem that you unearth is something that they haven't though of and they don't know how to fix? They will be tempted to avoid addressing it, possibly by attacking or threatening the team for even daring to raise the issue.

Too, too often, when someone on the stage shouts "Look a pirate ship." It's the director off stage that shouts something like "You're wrong, it's not there," or "I'm tired of this negativity" or "Maybe you're not up to the job if you think that's a pirate ship."

And that's why people don't raise problems and instead just scrub the decks.

Oh dear. Well. You probably picked this book up because you wanted to deliver (seemingly) impossible projects. I'm delivering on that promise. I'm not delivering on the promise that delivering these projects is a walk in the park or a day at the beach.

Even though dealing with the problems that you find and making them clear to your team and your sponsors can be a rough ride, demanding tenacity, tact, diplomacy and a willingness to be called not very nice names, there are two reasons why you still should do it.

Reason number one is that, despite the resistence you might encounter, solving these problems is still the best chance you've got of delivering this a seemingly impossible project. That's a really good, solid positive reason.

But reason number two is possibly for me, just as motivating: I know what happens when you don't.