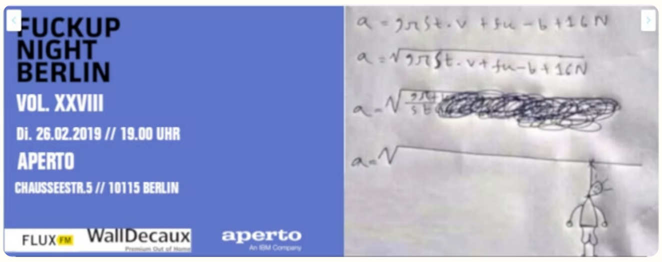
5 stupid things to do when building a software business

I would love to change the world, but they won't give me the source code.

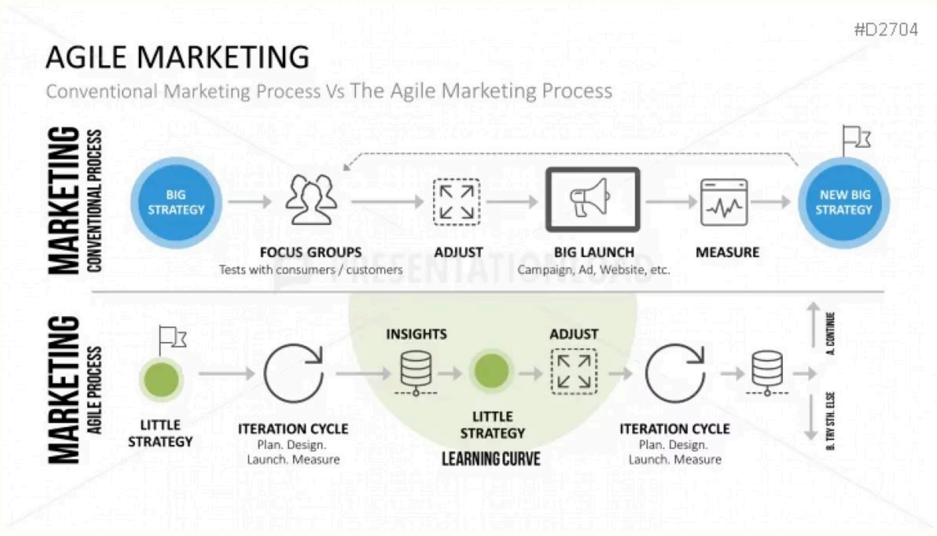
While waiting for the code, it is better to partly adjust to the world by learning from failures. In particular, the failures of others. To do so, I sat down with fellow tech leader Migan who not only worked as a programmer and tech lead at large corporations but also launched her own social enterprise and software business.

Why the obsession with failures?

Migan: When I look for a source to learn how to do things better, I find and read an enormous amount of success stories. Successful people seldom share their failure stories because it's not sexy. That's why I immediately fell in love with the Berlin Fuckup Nights initiative, where they invited founders and professionals to tell their fuckup stories and laugh with the audience about their past mistakes.



I want to see this attitude everywhere, not just on stage at Berlin Fuckup Nights! We are all human and we succeed because we failed and learned from our mistakes. Wouldn't it be great if we all shared our failures and celebrated them? Failure is just as important as success, and yes, a good winner is a good loser!



STUPID THING #1 - FORGETTING AGILE

We have all been "raised on" agile development and the new generation of software developers often don't even know any other way to work. The first article on Scrum, "The new new product development game," was published in 1986, nearly 40 years ago. Even so, we still sometimes lose track of the main point of agile development and do parts of the business development in a more rigid way. This could be quite dumb, especially if it is a part which is fast moving, such as market research for small SW products.

Migan: We did some market research to find the USP for our software and started implementing it. But wait! In a rapidly evolving technology market, you cannot do market research in a waterfall format! Every day you should check the market status… and re-evaluate your decisions, your business model and your pricing model. In our case, while we were laughing at low-code/no-code solutions, they grew rapidly and took over the software market. Even though our software offered immense customization at every level, which was initially highly desired by the customer, they eventually decided to go with a less flexible but cheaper solution!

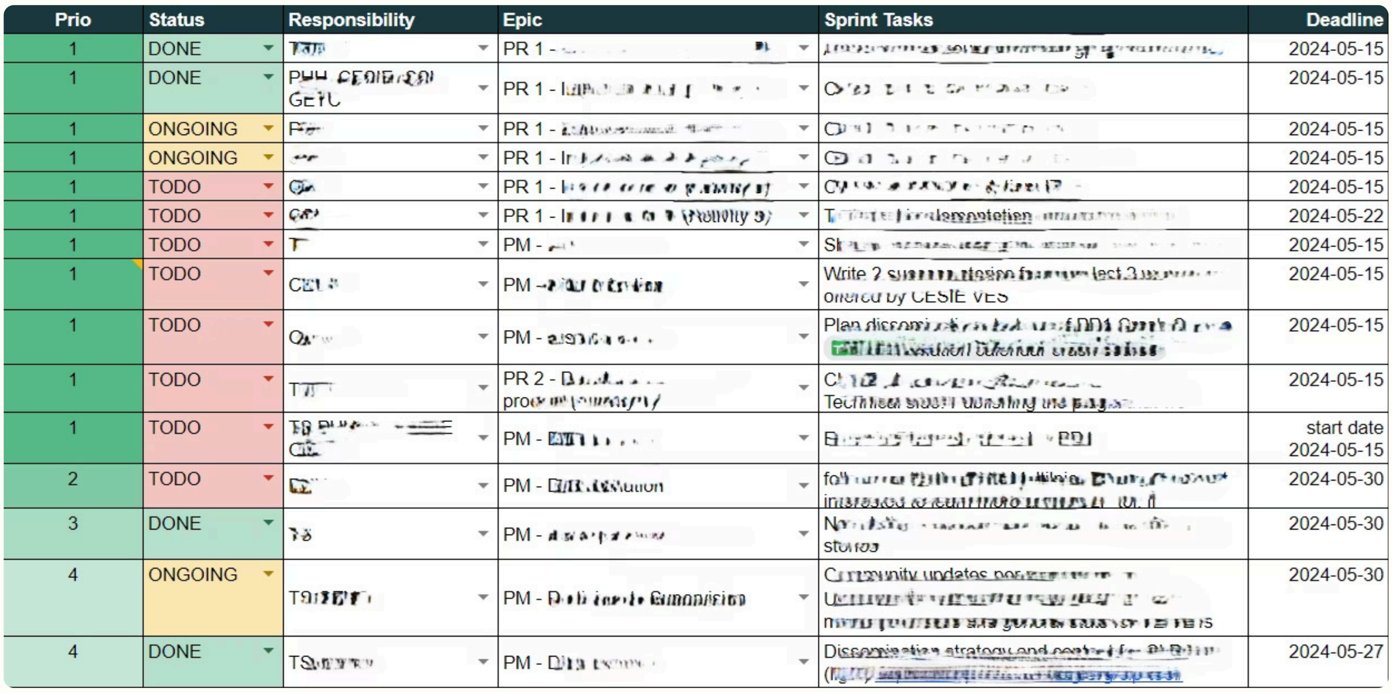
If I could go back, what would I do differently? I would do ongoing and iterative market research and encourage an agile mindset at all levels of the company, including market research and business development. Just a quick note, though -doing agile doesn't just mean implementing sprints in your marketing team, having everyone write sticky notes or Jira tasks and that's it, you're good to go. Doing agile is about the mindset of moving forward iteratively and failing fast. At each iteration, evaluate the outcome and decide if you are on the right track or if you need to try something else. You can all shake your heads because this is so obvious, but in practice many companies get so lost in setting up agile processes that they forget about the core purpose of it!

STUPID THING #2 - GETTING THE PARETO WRONG

Let's move on to something even older than agile development - "The Pareto principle" from 1896, sometimes referred to as the 80/20 rule - meaning that 80% of your revenue comes from 20% of your features (in a software example). The trick is to understand what those 20% are upfront.

Migan: Should the customer be dictating that? Or should it be the market research and what we can build as a USP? You can go wrong by trusting the customer to prioritise the 20%... They often end up wanting all the features and it's hard for them to understand that we need to do the 20% as a first priority... At least in my experience,

customers want it all... and in the priority table we ended up with a lot of epics marked as priority "1"!



Prio	Status	Responsibility	Epic	Sprint Tasks	Deadline
1	DONE	[illegible]	PR 1 - [illegible]	[illegible]	2024-05-15
1	DONE	[illegible]	PR 1 - [illegible]	[illegible]	2024-05-15
1	ONGOING	[illegible]	PR 1 - [illegible]	[illegible]	2024-05-15
1	ONGOING	[illegible]	PR 1 - [illegible]	[illegible]	2024-05-15
1	TODO	[illegible]	PR 1 - [illegible]	[illegible]	2024-05-15
1	TODO	[illegible]	PR 1 - [illegible] (Priority 3)	[illegible] implementation [illegible]	2024-05-22
1	TODO	[illegible]	PM - [illegible]	[illegible]	2024-05-15
1	TODO	[illegible]	PM - [illegible]	Write 2 [illegible] offered by CESIE VES	2024-05-15
1	TODO	[illegible]	PM - [illegible]	Plan [illegible]	2024-05-15
1	TODO	[illegible]	PR 2 - [illegible]	[illegible] Technical [illegible]	2024-05-15
1	TODO	[illegible]	PM - [illegible]	[illegible]	start date 2024-05-15
2	TODO	[illegible]	PM - [illegible]	[illegible]	2024-05-30
3	DONE	[illegible]	PM - [illegible]	[illegible] stories	2024-05-30
4	ONGOING	[illegible]	PM - [illegible]	Community updates [illegible]	2024-05-30
4	DONE	[illegible]	PM - [illegible]	Dissemination strategy and content [illegible]	2024-05-27

For example, in one project, the core product was a matching platform with a comprehensive customizable recommendation system, but there were many other features with lower strategic value that were pushed for and requested by the customer. We ended up reducing the effort in testing and customising the recommendation system and tried to squeeze in as many features as possible! In the end, a big spaceship with a lot of features was launched, and it was only then that we started to test the matching algorithm with real data, and - surprise! The recommendation algorithm behind the spaceship needed a lot of adjustments. If I could have convinced the customer at the early launch to focus on the feature which generated the core business value, which was the recommendation system (our holy 20%), and launch that 20% and test it with real data, we would have had a happy end user and a happy customer. And we would have still been able to continue building the spaceship in an iterative way, taking into account not only the customer's opinion, but also the end user's wishes.

Petra: I heard a couple of interesting comments with respect to not letting the customer decide on the wrong things. One was, "Do not outsource your strategy to your customer"- you should know your core and the value you believe you are delivering to your customers.

STUPID THING #3 - TOO MUCH TRANSPARENCY

Right now I feel there is a trend among managers for transparent leadership, transparent decision making, transparent whatnot. In theory, this is a good approach to have, and we should have fewer secrets and more transparency, but in practice you need to consider whether the people you are communicating with can handle the provided information.

Otherwise it's not really "transparency" - it's just dumping shit on them!

Migan: Can I make a quote out of your last line and frame it in our office? Yes, I agree that in practice too much transparency can be overwhelming. Transparency has always been important to us internally towards our team, but I feel like we have been too transparent with them and that can hurt their motivation! If customers are complaining and we communicate the complaints word-for-word to the team, over time they could lose their motivation... Transparency should also have its limits... if you explain to the customer the technical reasons why a feature is expensive to develop, maintain, or upgrade, you will lose them in the technicalities of the discussion! How to solve this? Well, in this particular example I think you should spend more time translating the technical discussion into human understandable language, giving it the right level of abstraction. Of course, this is easy to say, but we need more practical tips on how to do this. What is the right balance between transparency and opacity?

STUPID THING #4 - NOT MANAGING EMOTIONS

What happened to all the talk about EQ? Did it go out of fashion or are we suddenly so much better at interacting with each other?

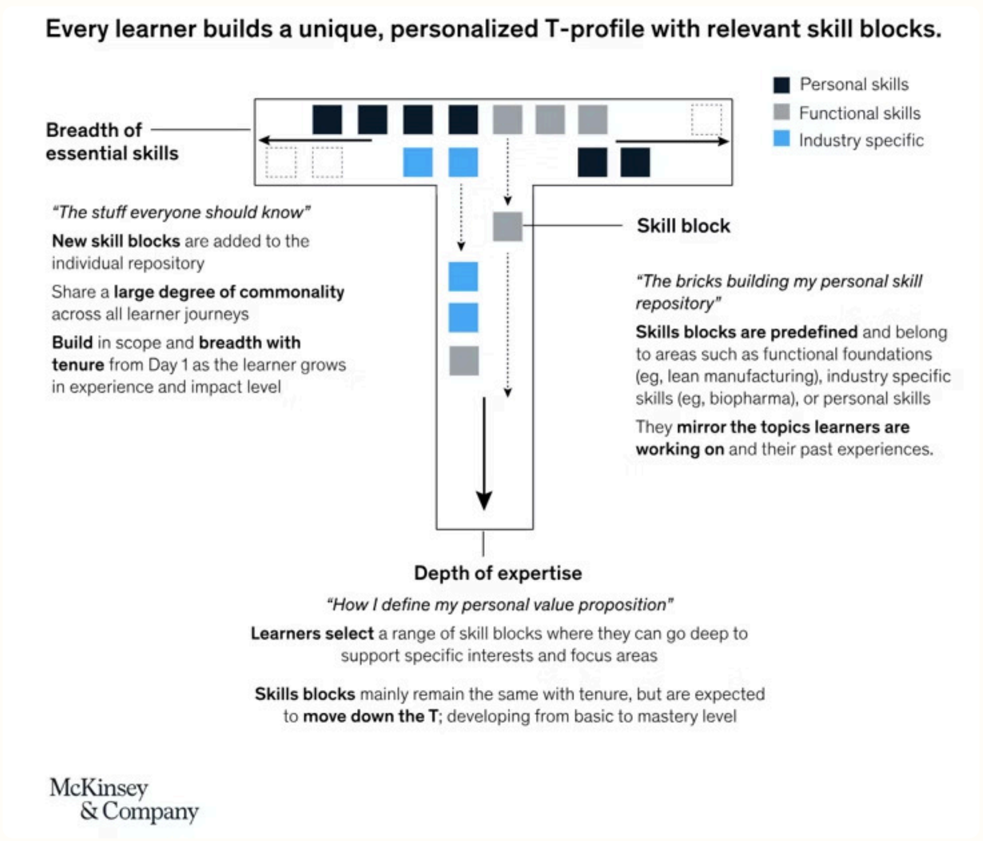
Migan: We develop software, but at the end of the day we are human beings and we have to get along with other human beings... Listening to them, understanding their fears, addressing their doubts... sometimes you do all that, but the chemistry just isn't there. The project manager may not get along with the client's representative. The client's representative may perceive that the project manager's approach doesn't fully meet their expectations. As a result, tension and frustration escalate throughout the project. Now, it takes experience to find the tipping point to decide whether to keep trying to fix this relationship or whether it's better to simply replace one of the two sides!

In the hiring process, it is common to conduct personality and teamwork assessments, let your team members be part of the interviews, and consider their opinion when hiring new team members. Why do we not do this when we start a new project with a new client? To me this preset installation sounds like a forced "arranged marriage" that, in all likelihood, cannot end well!

Petra: I remember when I worked at a large corporation going through the agile transformation. As part of the transformation long-lasting teams were established. This was quite a big change compared to working on projects, because then you knew that if you didn't particularly like someone, the project would eventually end and then you would be able to move on. One of the developers asked a manager - "what should I do if I'm in a long-lasting team with someone I don't like?" The manager recommended couple's therapy...

STUPID THING #5 - YOU

Petra: Do you remember when the trendiest thing was to be T-shaped? Even McKinsey has an article on T-shaped:


Every learner builds a unique, personalized T-profile with relevant skill blocks.
■ Personal skills
■ Functional skills
■ Industry specific
Breadth of essential skills
"The stuff everyone should know"
New skill blocks are added to the individual repository
Share a large degree of commonality across all learner journeys
Build in scope and breadth with tenure from Day 1 as the learner grows in experience and impact level
Skill block
"The bricks building my personal skill repository"
Skills blocks are predefined and belong to areas such as functional foundations (eg, lean manufacturing), industry specific skills (eg, biopharma), or personal skills
They mirror the topics learners are working on and their past experiences.
Depth of expertise
"How I define my personal value proposition"
Learners select a range of skill blocks where they can go deep to support specific interests and focus areas
Skills blocks mainly remain the same with tenure, but are expected to move down the T; developing from basic to mastery level
McKinsey & Company

When you launch a SW business you need to be strong in your tech domain, probably even hands-on, and you also need to be a leader.

Migan: Yes, exactly! This is a challenging requirement, where you have to do everything that is expected of a leader without losing touch with the technical debt. How can you manage these two roles, where one stretches you horizontally and the other vertically?

Another fact is that when you are new to something - for example, a technical person new to leading a business - you are more likely to make mistakes! When I started leading IT projects at my previous employer, it was a very comfortable situation. The large corporate setup gave me a lot of room to learn by doing and by making mistakes. But when you start your own SW business, the price of your mistakes can be very high and cost you your business. Before you start experimenting in the horizontal part of the "T", I think it will pay off in the long run if you take some time in the early stages of building your business to get some leadership and business training. Additionally, it can be very valuable to bring strategic/business/leadership advisors on board who advise you on real-world situations beyond your theoretical learnings.

Petra: It sounds like you will never be done. And actually, building a business is often compared to athletes and their sports achievements - you should work hard, do your best, run until you puke...

Migan: I don't think you should always give it your all... because then if you get rejected you break hard! This is not a love story based on romantic ideals - it is a Software development story. For one particular customer, we went above and beyond, pushing all our limits to meet their needs. While it's natural to want to deliver exceptional service, if you consistently go beyond your boundaries, it can set unrealistic expectations. Over time, the other party may come to expect that level of effort without fully realising the extra work behind it. It's important to establish healthy limits from the start. This way, people understand the expectations, and when you occasionally go the extra mile, it's genuinely appreciated and creates a positive surprise. In psychology, this is referred to as *"The norm of reciprocity"* combined with *"habituation"*.

CONCLUSION

There is a common theme in the scenarios and examples discussed here - the difference between following a principle or guideline in theory and implementing it in practice. It is important to learn how to fine-tune your approach to fit the real world and not fall between the rigid lines of a checklist. We have to factor in the fact that we are human and that even when you are building a business or a software product, you and everyone around you is ultimately driven by human emotions.

Migan: Let me share a line from the Berlin Fuckup Nights website: *"Sometimes you win. Sometimes you learn"*. Unfortunately, the best way to learn is by doing - like toddlers learning from every fall and bruise, we learn from every situation we encounter and gradually develop an instinct that helps us avoid failure. However, even though nothing can replace direct experience, we can still help each other by sharing stories of our mistakes and what we learned from them. And, trust me, you fail more gracefully when you share your story and help others to rise!