Why We Need Buffers

I think everyone knows Murphy's Law: *Everything that can go wrong, will go wrong.*

I agree in reality, it isn't so absolute, but **unless we recognize and proactively mitigate the factors that make things go wrong, the probability is very high.**

Why am I bringing this up now?

Because today, I witnessed something, once again, blow up (more specifically almost every time there's a release, something goes wrong). **Why?**

One reason, **those directly responsible, who should've foreseen this and prevented it, didn't.** They're not the ones feeling the pain and dealing with the consequences. They have no incentive or motivation to improve.

The other reason, **there are no buffers.**

A Simple Example

If it takes you, 30 minutes to get to an appointment and you leave home 30 minutes before the scheduled time, what are the chances you will be late?

Well pretty high if we're just talking about a few minutes… but once in awhile you're going to hit a traffic jam and be late by 15 minutes… and on rare occasions, your car completely breaks down and you have to cancel.

What we are interested in is the expected cost:

*Sum(Probability(x)*Cost(x))*

The possible costs are basically: trivial, bad, very bad

But the probabilities of each scenario is very different. Something like 80%, 19%, 1%

So in this case, it's probably advisable to leave **just 40 minutes** before the appointment.

What if the first 2 probabilities are flipped? Then you'd probably want to leave **at least 60 minutes** earlier.

That's common sense right? Well what about picking a time where you're pretty sure there will be no traffic jams and making sure your car is well maintained? *For this case it's a sort of overkill but leads into my point.*

You can actually take proactive steps to change the probabilities and lower the chances that the expensive cases will occur. **You can lower the expected cost.**

And what is one way to do this? Allot enough time to make sure things are done correctly and checked.

Building a Buffer and Keeping It

What I see day to day is that we accept too much work and their requirements keep changing.

There's way too much work (multiple changes) for the deadline we need to meet (3 week/sprint). This includes design, coding, and testing… for each change.

So usually coding and fixing bugs takes 80% of the time; 15% for testing, including setup overhead; 5% for design and that's being generous.

Also the requirements are unclear and keep on changing until the last minute, literally… which makes testing rushed or basically moot. It's a check the box formality.

So what does this add up to?

Poorly designed and inefficient code that is hard to change, with high probability of bugs. And even though it's hard to change, changes are needed… at the last minute…so we have to resort to making hacks. Hacks that are probably not well tested given the now lack of time.

Basically everything is rushed and extremely fragile... if not already broken.

What could possibly go wrong?

1. What do you think the probabilities would be for: smooth, some minor problems, a catastrophe?
2. The expected cost?

And yet somehow managers don't see this.

They make everyone give their 100% all the time with tight deadlines, high pressure, and constantly changing requirements. *That in itself drives the probability of a mistake up by **A LOT***

Oh and in addition to coding, we have to deal with production issues… **But wait!** There's no time for that! In their mind, fixing problems doesn't take time!

Why do we have all these issues anyway?

In addition, they keep bending the rules and making exceptions…

A few specific ones:

- We have hard cut-offs
- We are supposed to have Sprint planning **which should involve developers.** We do but it's an managers only affair… *WTF? Don't even get me started on this but you know in Agile, PMs should only set the backlog; developers decide what to actually do in a Sprint*

Why do we have these rules?

The answers the 2 questions are in some form:

- everything is rushed because **there is no buffer** and therefore quality drops like a rock
- **to ensure there is enough buffer**

Why don't they understand this?

It seems they are optimists, not realists... and they don't learn.

They aren't feeling the pain… and they don't take input from the people who do.

Whenever there is a problem, they just hope it doesn't happen again rather than taking steps to keep it from happening again... and we all suffer for it…

And it's sort of ironic that I have to tell them this, I'M NOT A MANAGER!
I am an avid reader though…

I've actually told them to read *The Phoenix Project*, which points this problem out clearly.

And they say they have read it and completely understand it but still… nothing has changed…