Getting excited about maintaining legacy systems

Blanca Garcia Gil
LDX3
June 17th, 2025

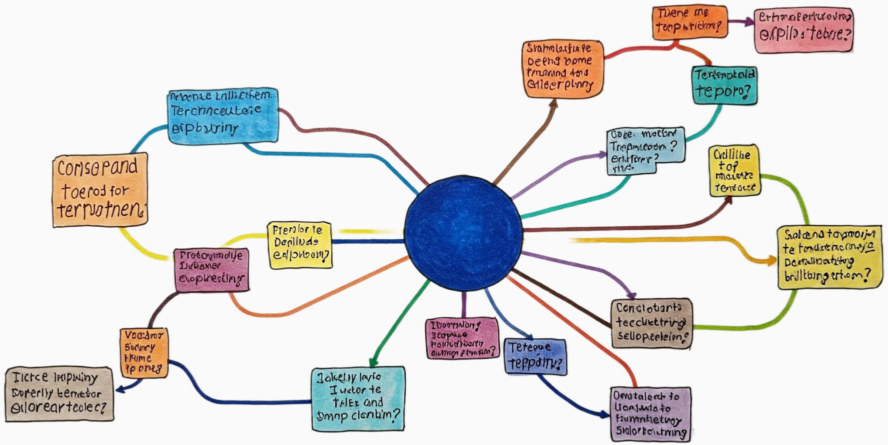
Imagine you join a new company and you get your first piece of hands on work.

You are a software engineer

Your first thought:
“This code base makes no sense”

You secretly start blaming whoever worked on this before you .

You visualise a future where you have a clear framework for how to proceed into the unknown.



Maintaining software is different to working in greenfield.

Let’s look at three foundational skills that will pave the way for success

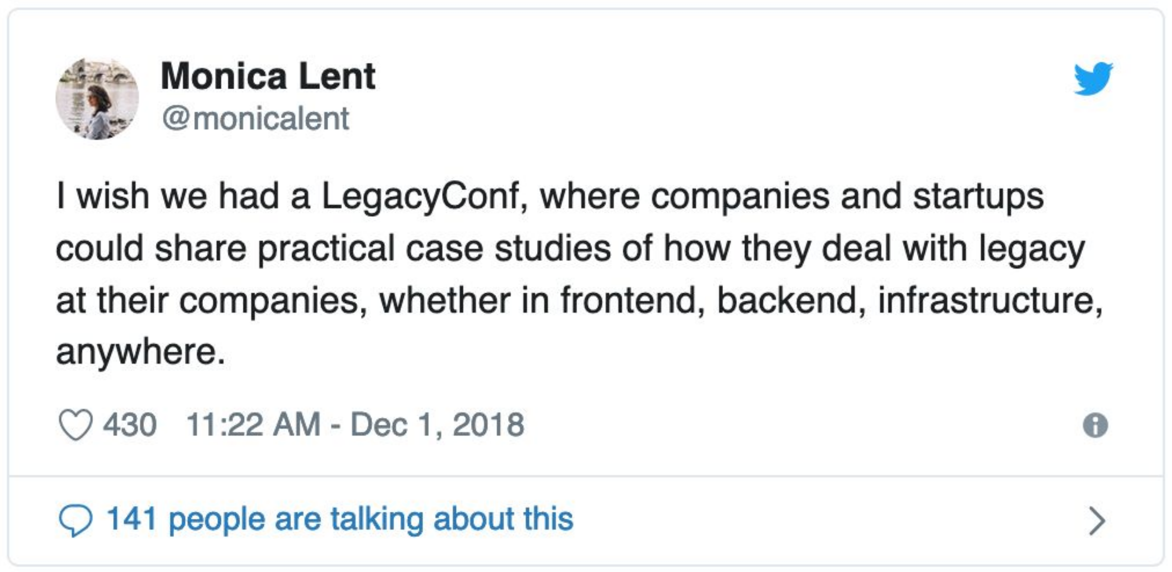
Blanca Garcia Gil (she/her)

Principal engineer and transformational coach

Founder staffplusengineer.com

Legacy code has been part of every job I’ve had

Turns out I'm not the only one excited



Monica Lent
@monicalent

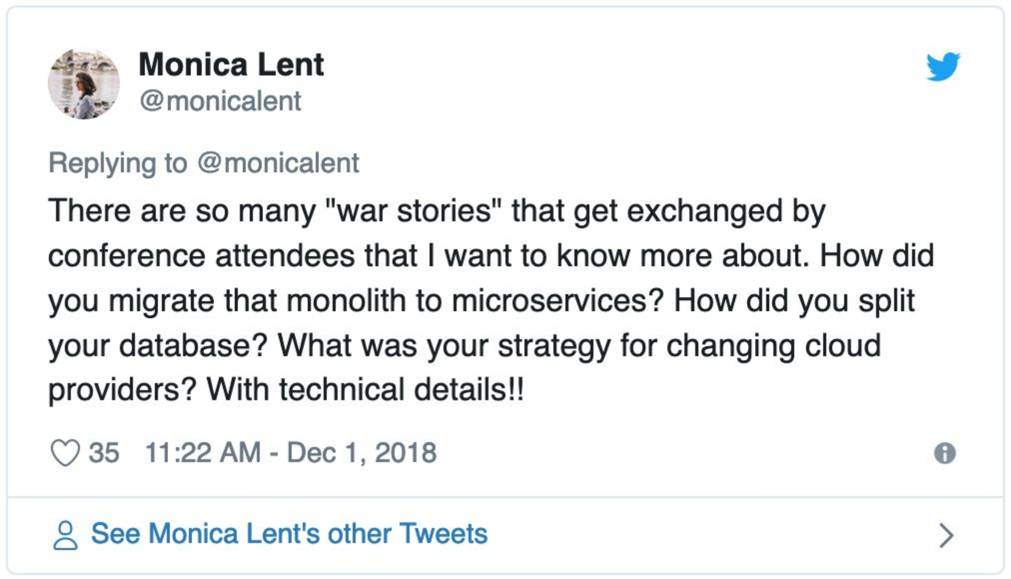
I wish we had a LegacyConf, where companies and startups could share practical case studies of how they deal with legacy at their companies, whether in frontend, backend, infrastructure, anywhere.

♡ 430 11:22 AM - Dec 1, 2018

141 people are talking about this

https://twitter.com/monicalent/status/1068812692931178496

Legacy provides an opportunity for learning



Monica Lent
@monicalent

Replying to @monicalent

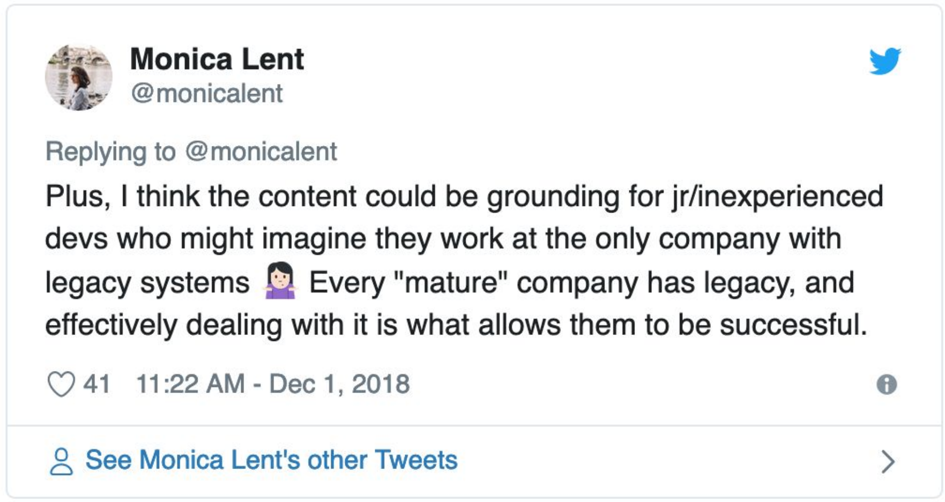
There are so many "war stories" that get exchanged by conference attendees that I want to know more about. How did you migrate that monolith to microservices? How did you split your database? What was your strategy for changing cloud providers? With technical details!!

♡ 35 11:22 AM - Dec 1, 2018

See Monica Lent's other Tweets

https://twitter.com/monicalent/status/1068812694084562944

The legacy landscape



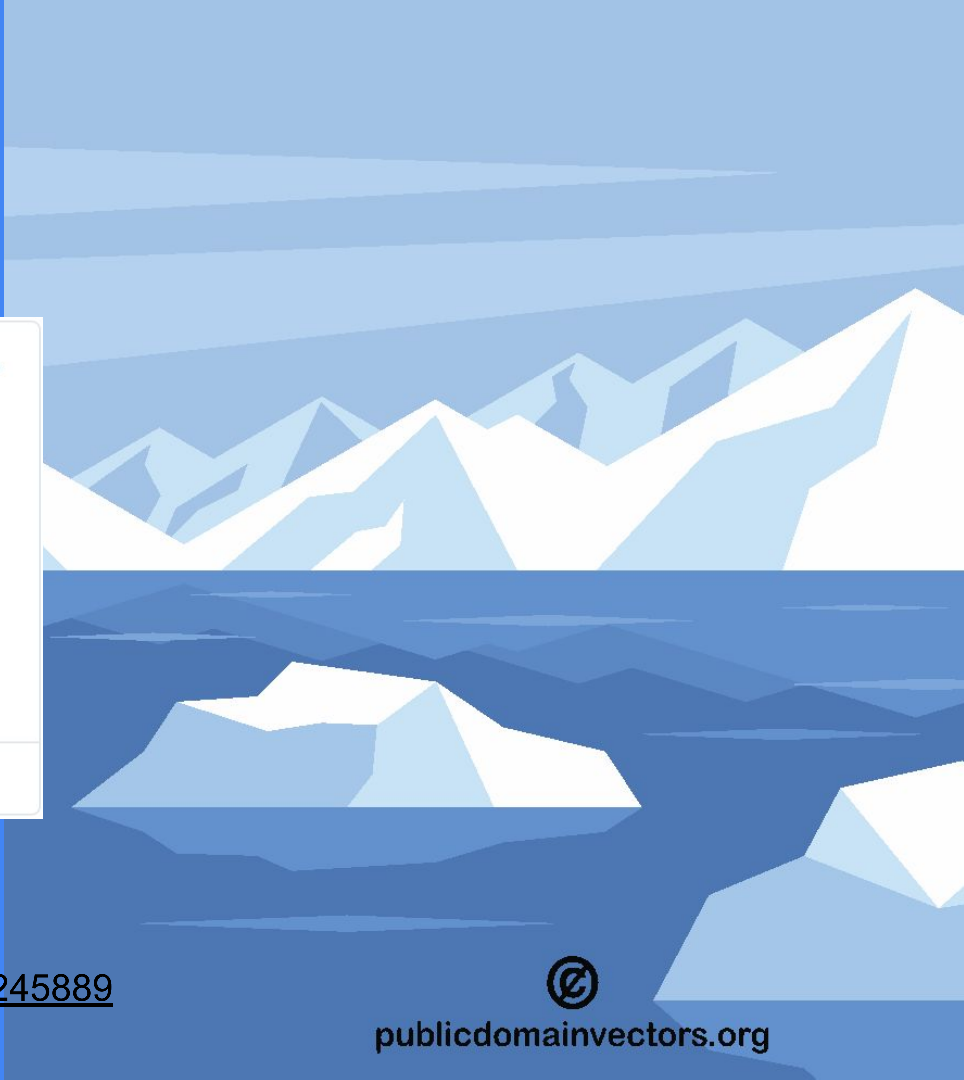
Monica Lent
@monicalent

Replying to @monicalent

Plus, I think the content could be grounding for jr/inexperienced devs who might imagine they work at the only company with legacy systems 🤷🏻‍♀️ Every "mature" company has legacy, and effectively dealing with it is what allows them to be successful.

♡ 41   11:22 AM - Dec 1, 2018

https://twitter.com/monicalent/status/1068812695502245889



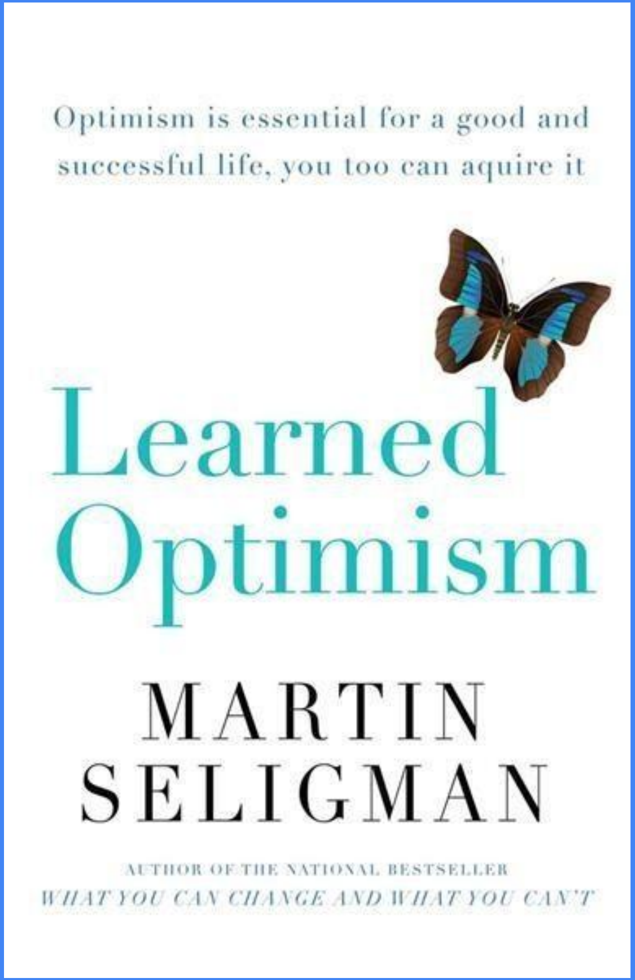
Agenda

1. Introduction
2. Cultivate a learning mindset
3. Create a map of the software system
4. Predict and prevent possible pitfalls when implementing new systems
5. Key takeaways and wrap up

~~**1. Introduction**~~
2. Cultivate a learning mindset
3. Create a map of the software system
4. Predict and prevent possible pitfalls when implementing new systems
5. Key takeaways and wrap up

1. Introduction
2. **Cultivate a learning mindset**
3. Create a map of the software system
4. Predict and prevent possible pitfalls when implementing new systems
5. Key takeaways and wrap up

Changing our mindset as a starting point



Optimism is essential for a good and successful life, you too can aquire it

Learned Optimism

MARTIN SELIGMAN

AUTHOR OF THE NATIONAL BESTSELLER

WHAT YOU CAN CHANGE AND WHAT YOU CAN'T

Analytical thinking is what we do most in our jobs

But we don’t spend enough time building empathy or emotional intelligence skills

As senior, staff engineers or architects we lead the way for others

“Operating under constraints, is the key to creativity and fun.”
- Ian Bogost

https://ideas-ted-com.cdn.ampproject.org/c/s/ideas.ted.com/want-to-be-less-distracted-try-this-find-the-fun-in-tedious-tasks/amp/

Why does technical debt matter beyond code quality?

1. Developer productivity and happiness

2. Reliability of our systems

What are the consequences on the build up of technical debt over time?

Inability to fix bugs or make improvements can lead to:

software event horizon

Doing nothing is not an option

Face our fear of opening the closet and seeing how many skeletons there are

… A story of how avoidance doesn’t solve anything

1. Introduction
2. Cultivate a learning mindset
3. **Create a map of the software system**
4. Predict and prevent possible pitfalls when implementing new systems
5. Key takeaways and wrap up

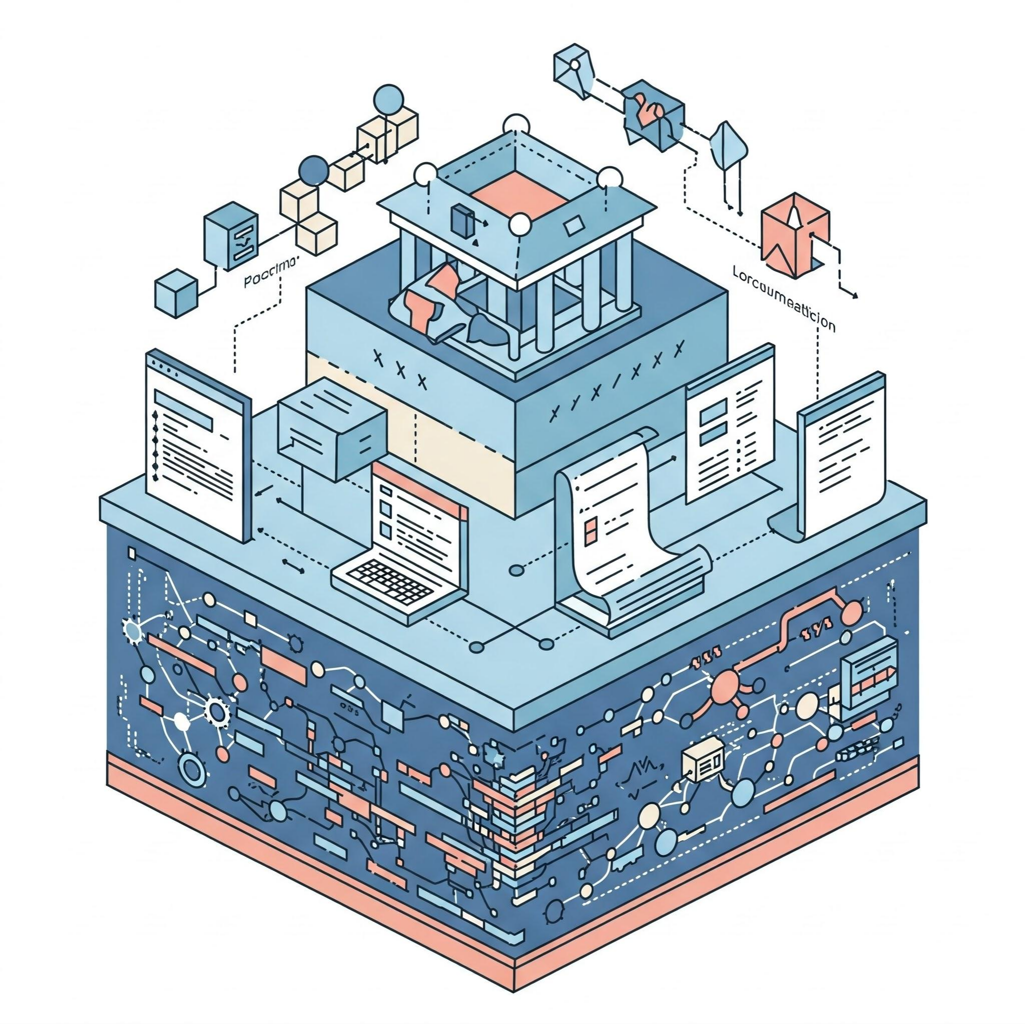
Use our **systems thinking** skills to take a step back

Creating a map will help us connect the dots

Understand what the software system is meant to do

Peel off the layers:

- Architecture
- Documentation
- Code base & tests



“Fun is the aftermath of deliberately manipulating a familiar situation in a new way.”
- Ian Bogost.

https://ideas-ted-com.cdn.ampproject.org/c/s/ideas.ted.com/want-to-be-less-distracted-try-this-find-the-fun-in-tedious-tasks/amp/

Understand the team context

Speak to people in a variety of roles to get a bigger picture view

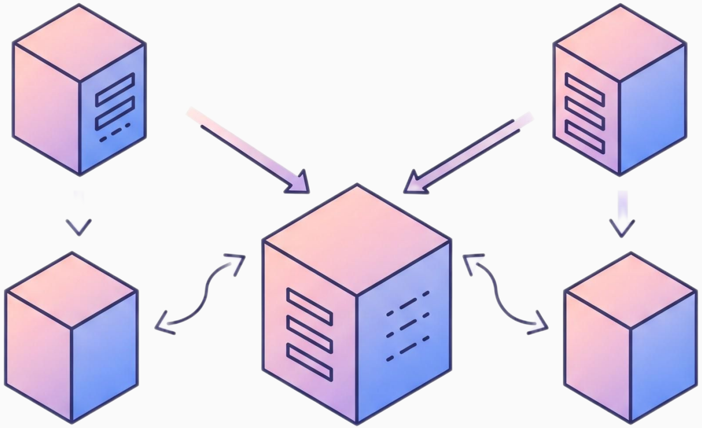
Product manager

Business analyst

Sales person

Test engineer

Look at the edges of the system



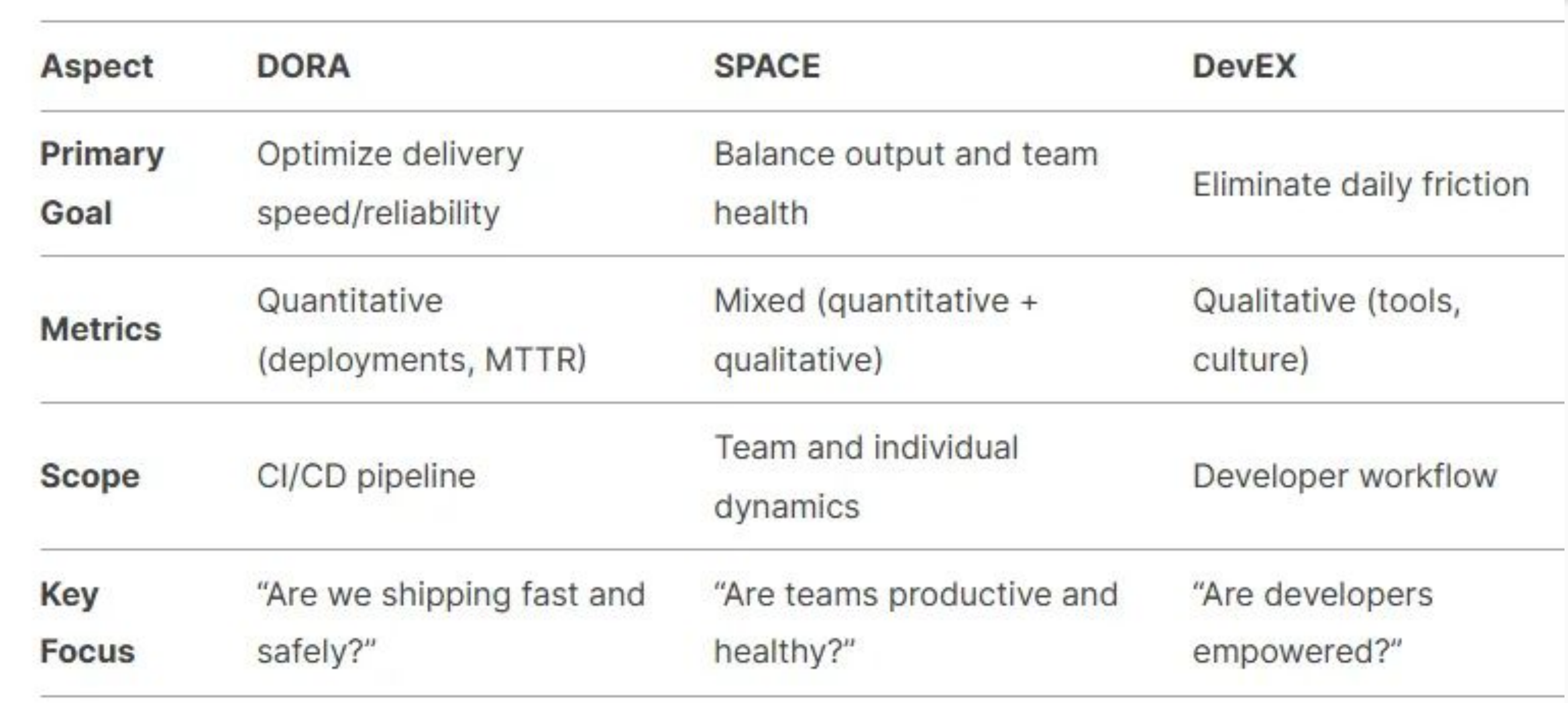
Look at what other systems it is connected to

Are there any contracts in place? (API specification, testing contract, data contracts, etc)

1. Introduction
2. Cultivate a learning mindset
3. Create a map of the software system
4. **Predict and prevent possible pitfalls when implementing new systems**
5. Key takeaways and wrap up

Document incident reviews… a personal story

Track where tech debt might be creeping into your systems



Aspect	DORA	SPACE	DevEX
Primary Goal	Optimize delivery speed/reliability	Balance output and team health	Eliminate daily friction
Metrics	Quantitative (deployments, MTTR)	Mixed (quantitative + qualitative)	Qualitative (tools, culture)
Scope	CI/CD pipeline	Team and individual dynamics	Developer workflow
Key Focus	"Are we shipping fast and safely?"	"Are teams productive and healthy?"	"Are developers empowered?"

https://medium.com/code-factory-berlin/unlocking-engineering-excellence-dora-space-devex-and-beyond-10a2a1d844d4

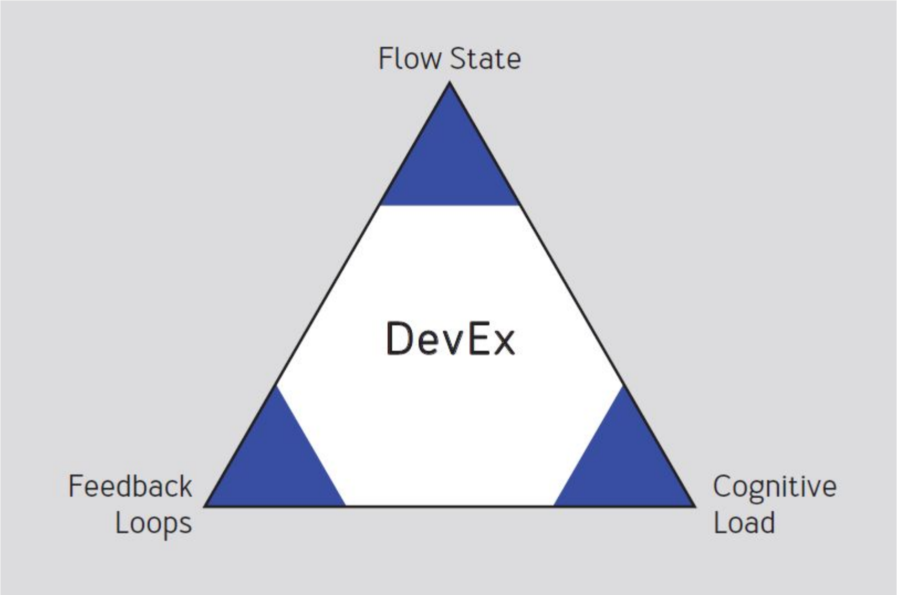
DevEx (2023)

Flow state - balance between challenge and capability

Feedback loops - tests, dev environment setup, pull-request reviews

Cognitive load - documentation, ADRs (architecture decision records)

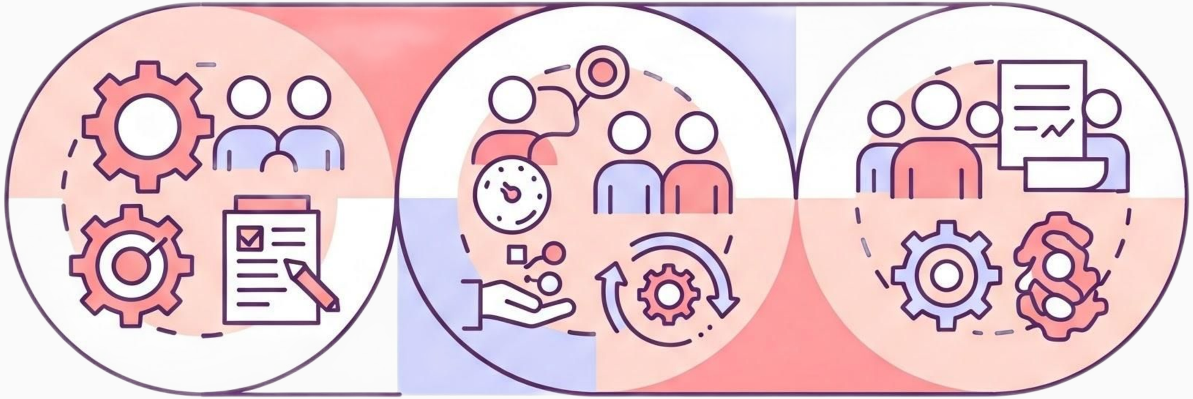
FIGURE 1: **THREE CORE DIMENSIONS OF DEVELOPER EXPERIENCE**



Develop healthy practices to prevent technical debt growing

1. Prioritise regularly work dealing with technical debt

3. Be intentional when you incur technical debt, document and communicate it.



2. Adopt good team practices: documents, commit log messages, architecture decision records, etc

1. Introduction
2. Cultivate a learning mindset
3. Create a map of the software system
4. Predict and prevent possible pitfalls when implementing new systems
5. **Key takeaways and wrap up**

Key takeaways: learn **three skills** to deal with technical debt more effectively

1. Cultivate a learning mindset

2. Create your map of the software system

3. Know how to predict and prevent possible pitfalls when implementing new features and changes in the code base

Motivation isn’t something you either have or you don’t.

“It’s something you build - by understanding your desires, facing your fears, and designing actions that feel good to take”

- Moni, my coach

THANK YOU!

Connect with me:

staffplusengineer.com

bgarciagil@gmail.com

linkedin.com/bgarciagil

blanquish.bsky.social