Legacy Code: Sunk Cost or Opportunity?

CTOCRAFT Winter 2022 Lightning Talks

Richard Bown
Independent Software Consultant
@bown_rw

About Me

- Independent Software Consultant
- Software Developer -> Lead -> Config -> PO -> Head of Engineering
- 25 years in industry (Banking, Telecoms, Insurance, Energy, FMCG, SaaS Products, Open Source, Startup)
- DevOps, Automation and Re-Use
- Systems Thinking, Conway's Law, Team Topologies FTW
- i.e. the-team-is-the-software approaches

Disclaimer

- This talk didn’t exist at this time yesterday
- Pictures are raw and WIP
- Too much text and not enough inspirational quotes
- First delivery - you are therefore lucky people!
- **Please** provide feedback

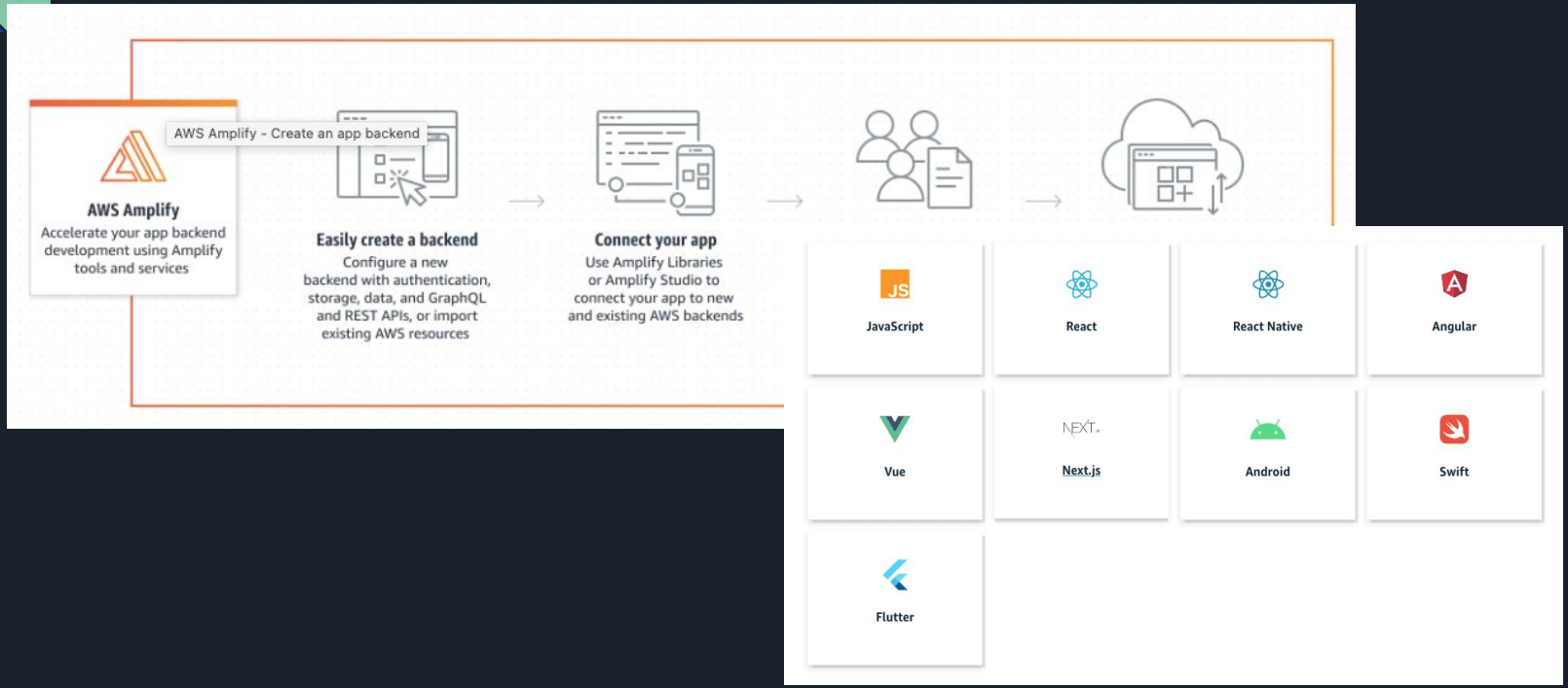
Technical Debt

- Technical Debt - is a broad definition or catch-all (see Wikipedia)
- Technical Debt is about decision making in all parts of the process.
- Coding standards, architecture, repository setup, branching strategy, build pipeline, library and dependency use, deployment method, infrastructure design, testing strategy, change management, policy etc.
- Things that we've not considered ahead of time i.e. lots of things
- Systems vs Software
- For the purposes of this presentation it's the same or interchangeable with Legacy

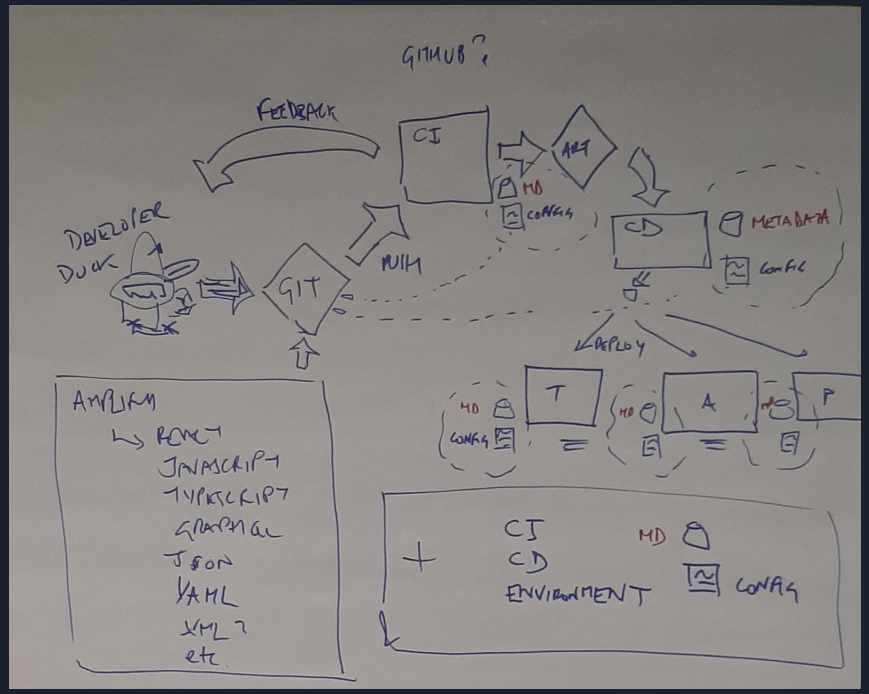
When Does Software Become Legacy?

- We build from assumptions
- No model is perfect
- When it finally goes in front of users - they make it more complicated
- Deployment makes it inflexible and brittle
- Feedback is suddenly fast
- Architecture gets extended
- Whole systems get created, re-written, deleted
- Compliance and perhaps security are (sometimes) afterthoughts

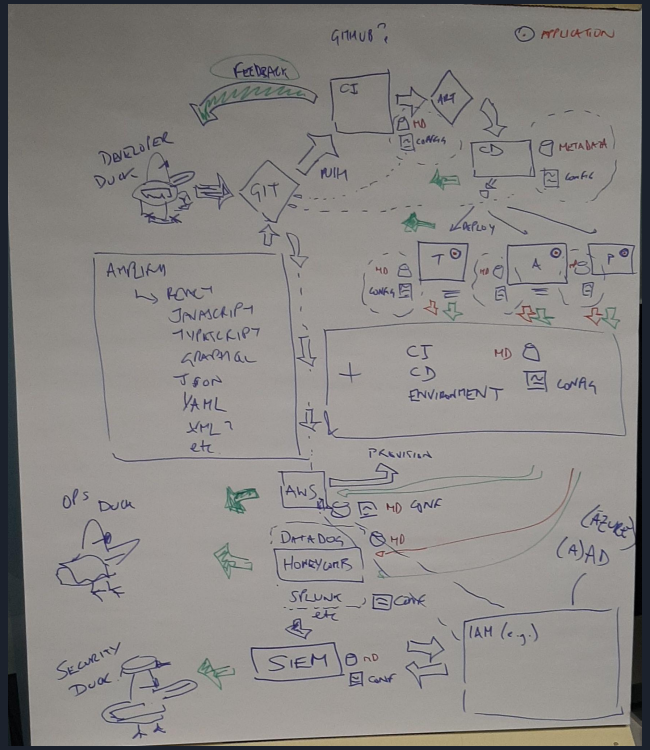
Example with Amplify Framework



How does this look?



And then deploy it...



Twitter: @bown_rw

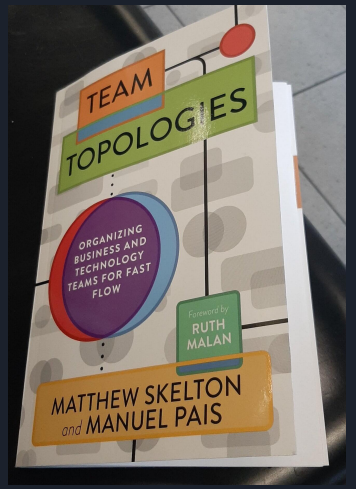
Richard Bown

ctoproblems.com

Cognitive Load

In cognitive psychology, **cognitive load** refers to the amount of working memory resources used. There are three types of cognitive load: *intrinsic* cognitive load is the effort associated with a specific topic; *extraneous* cognitive load refers to the way information or tasks are presented to a learner; and *germane* cognitive load refers to the work put into creating a permanent store of knowledge (a schema).

See Team Topologies for more on cognitive load.



Feels Like Legacy

“Technical debt is what you feel the next time you want to make a change”

Ward Cunningham (2003)

Legacy is Immediate

- Something in production means we have legacy
- All future decisions will be built from this baseline
- Users influence our decision making
- How can we optimise the above for continuous change?

Instead often:

- Put more layers of complexity on top of our system
- Slow down the pace of change
- Grow and find it harder to realise what we're trying to deliver

Legacy is Everywhere

- Enterprise, scale-up, start-up
- In every major or minor decision we make
- Both inside and outside the codebase
- The supporting systems becomes part of our critical infrastructure (ticketing, pipelines, IAM, infrastructure, SIEM)
- Having multiple (redundant systems) is the simplest and cheapest option in the short term (so we duplicate systems like Jira, Azure DevOps, Gitlab)
- Silos form immediately around tooling and expertise

Why Does Legacy Hurt Us?

- We have customer imperative (we are in a rush)
- It can quickly become difficult to accomplish even small changes
- Legacy systems are unpopular because they are harder to work with
- We often don’t spend enough effort on the mundane things to make it simple

Being Realistic With New Technology

- It will be microservices
- It will be Kubernetes
- It will be fully decoupled, easier to deploy
- It will be a destination for engineers who will flock to work on it
- We will open-source portions of it
- We will become a platform for the rest of the business to re-use
- We will finally be able to retire the monolith

..or we have a realistic plan that will enable us to implement it alongside existing platform

How To Breathe New Life Into Legacy?

- Strategic and tactical approaches are available - it's a business decision
- Refactoring along fracture boundaries (*Domain Driven Design - Eric Evans*)
- *Michael Feathers - Working Effectively With Legacy Code*
- *The Unicorn Project - Gene Kim (The 5 ideals of Software Development)*
- Investment in test coverage and automated testing
- Investment in platforms (consolidating/expanding)
- Investment in skills
- Realising effect of Conway's Law (your organisation impacts your software products)
- Considering Team Topologies

Or...?

If Legacy is immediate...

..engineers should always be invested supporting business value.

How can we make sure that everyone feels like we are supporting business value every single day?

What Can We Do?

- Accept that Legacy is Everywhere and Immediate
- Empower teams to own it
- Equip individuals with the skills to deal with it
- Change our organisation to align with our software products
- Make new products and technology a part of our legacy story
- Allow strategy to dictate not reporting lines
- Don’t stop challenging ourselves and each other

Thank you!

LinkedIn: https://www.linkedin.com/in/richard-bown-a0762b2/

Twitter: **@bown_rw**

Websites: https://richardwbown.com

https://ctoproblems.com