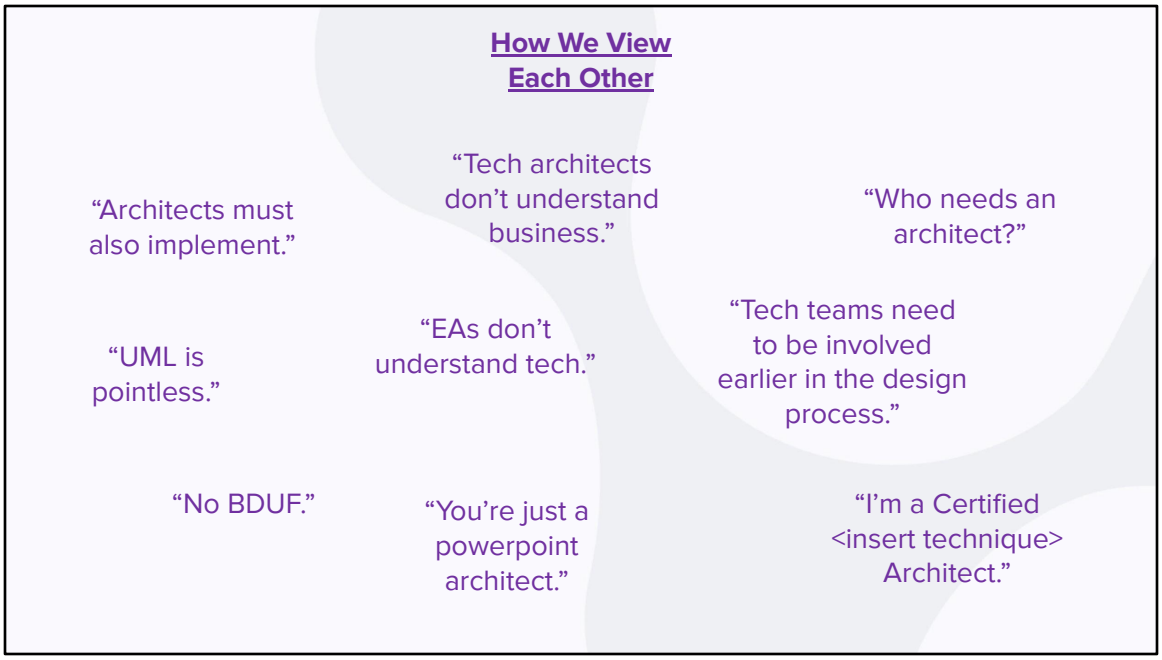
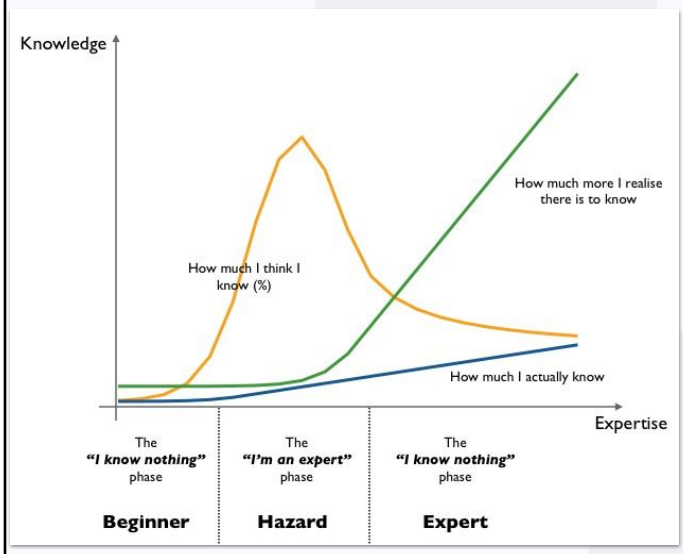

- Been working design of single apps to EA design for multi-national companies. Want to talk about the different kinds of architecture I've experienced if you transition from technical architecture to enterprise / business architecture.
    - What's different and what stays the same

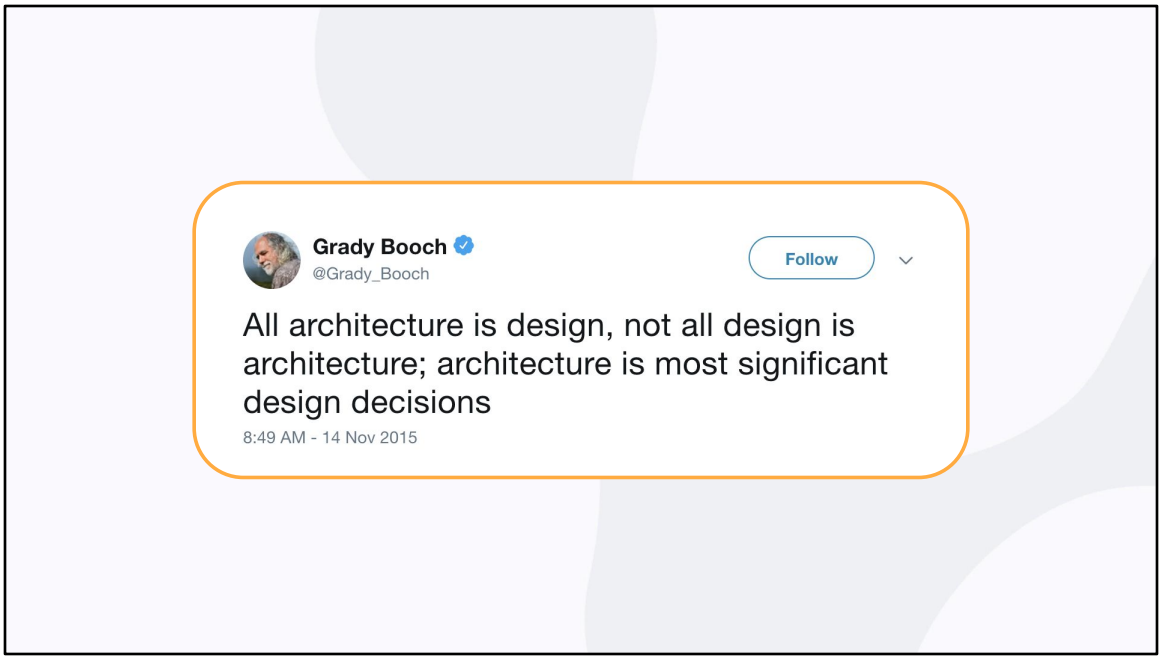



- Observation - we don't understand each other and can be a bit disparaging
  - Observe that lots of ideas in architecture (talks, twitter, blogs) take a very narrow focus on what architecture is
  - "Architects must also implement", "you should use UML / C4", "technique X is _better_ than technique Y", "I'm a Certified <insert technique> Architect"
  - Learnt that there are certain similarities and differences
  - These all tie back to different aspects of design

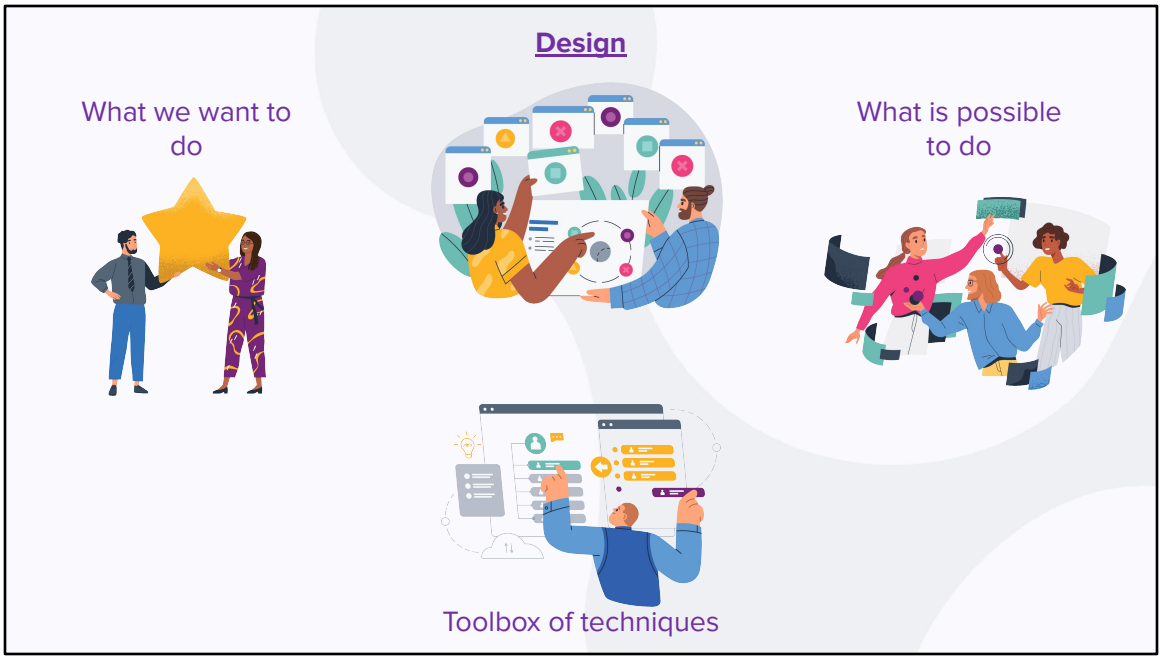



The three stages of expertise
https://blog.gardeviance.org/2008/04/three-stages-of-expertise.html


- Context - don't claim to know the answers
  - experience has taught me that sw arch covers a whole range of things and how little I know about it all
  - Worked with many different types of architecture during my career and looking back an observation is that I've become less opinionated - well, I still have very strong opinions - but I'm sharing them less because I realise more and more how hard it is to have an opinion that covers all the different aspects of sw arch. So, I want to reflect a little on why that it is
  - (I know Simon Wardley originally published this as a joke graph but it captures pretty well my understanding of architecture through the years)




- What is arch - a definition to talk around
  - This is the point where a talk like this introduces their definition of architecture
    - No shortage of definitions. You certainly don't need a new one from me. Never found the right one
    - Such an overloaded term.
    - Everyone who doesn't fit in another category seems to put architect as a suffix to their role
    - Miss the old CMU/SEI page of architecture definitions.
    - Lets go with Grady Booch. When I started, OO and architecture were all the rage. And he's provided lots of great material on both.
  - Its about design - structurally and strategically important design that is hard to change
    - If you’re not doing design, you’re not doing architecture – even if architect is somewhere in your title.
    - Not about when you do it. you’re doing it continuously




- What do we do when we're doing design
  - What do we want to do <-> What is possible to do <-> toolbox of techniques to resolve these two
    - What we want to do
      - want to do want to achieve. the outcomes. functionality, qualities (NFRs), emergent properties
    - What is possible to do
      - What are we building with. Starts with code constructs but becomes so much more. Systems, capabilities, teams, processes, operating models
    - Toolbox of techniques
      - methods, patterns, tools, stack overflow answers
      - How do we analyse what we want to do so that it starts to match what's possible to do?
        a. How to we explain what's possible to do so that it guides how we think about what we want to do?
  - Whole variety of architecture situations that are different in each of these three areas
    - Makes all-encompassing statements about architecture work - like "architects should implement" and "you should use C4 / UML" - seem ridiculous. They come from such a narrow perspective on what we all do

- We spend a lot - A LOT - of time talking about what's possible and the techniques
  - The building blocks and the techniques are non-ambiguous. more deterministic
  - what we want to do can be situation-dependent. Hard to generalise
  - hard to talk about the bit in the middle. agreeing on the right tradeoffs can be hard. Nebulous.
    - have found some things that work. E.g., Patterns. I miss the focus on patterns we had in the mid 2000s.
  - But when we lose sight of these aspects then we fall into cargo-culting
  - Being certified in a technique or technology doesn't make you an architect. Might be a good technologist or process-driver though.

## What We Want To Do: "The Business"

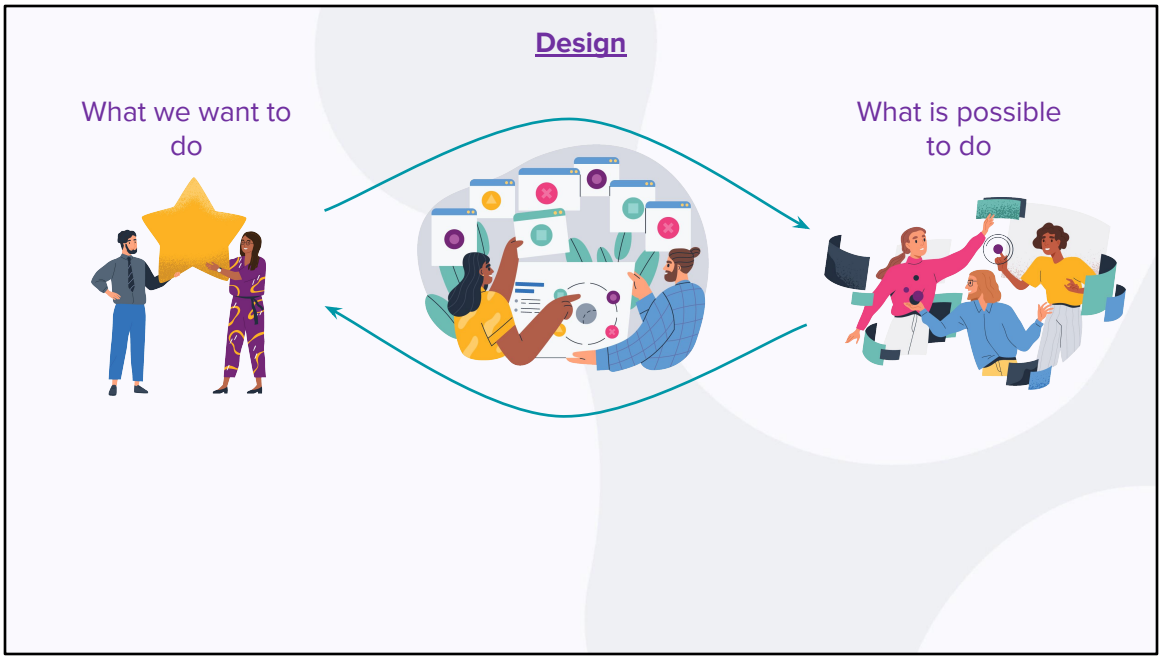
What we want to do

"I want to be the link between Business and IT."

- Every Architect's CV in the industry (including mine)

- The Business - our weakest understood area
  - Read lots of CVs, ALL of them, including mine, have some version of "I want to be the bridge between business and IT".
  - One of things I've realised over time is that I've had a very poor understanding of what business is and the different types of things that they want to achieve
  - Perhaps obvious to those that enter EA/BA from business analyst approach but wasn't for me entering from the technical side
  - Lets consider the opposite (a good technique against bias)
    - If I was the business domain expert on an IT development project, I could claim to be the link between business and IT.
    - but its such a narrow view of IT. What about IT management. What about operations. What about IT quality assurance.
    - IT people would never assume that such a person "understood IT" and could be the link between business and all facets of IT
  - Tech Archs talking about the importance of migrating to a new tech because it will attract other developers. Which is true. But for a business that has to live with this app for 15 years. That may not be the best choice.
  - Or a Bus Arch arguing for a complicated metamodel or framework compliance without understanding how hard it is to get 1000s of people in a distributed company to move in the same direction.

- Understanding business in a more sophisticated way is the biggest thing I've had to learn. Is what I'm doing going to generate revenue or reduce cost.
  - Might be a really interesting tech idea. But what is the cost to make it happen? Is it really going to move the needle for this company? What are the long term costs/benefits?




- They feed each other
  - what you want to do challenges to find new possibilities
  - new possibilities trigger new things you want to do
  - Design is the synthesis of these things
    - where the tradeoffs happen
    - if you're not combing what you want to do with whats possible to do, then you're not doing design - and by extension - you're not doing architecture
    - Tech / technique competence is highly skilled but not architecture. If you're a technologist with extensive knowledge of code / tech constructs and consequences but you don't know what the business is trying to achieve. Then you're not doing architecture.
    - Business competence without understanding the possibilities and constraints of whats possible to do it also not architecture. If you're a business architect designing how processes should happen in a company but you have no idea about whats possible to do with in the system landscape and the teams that make use of those systems - then you're not doing architecture. Might be a really good business analyst - and thats useful - but you're not doing Business / EA
    - The key here is the Design Concept. I'm going to come back to it later.
  - Will give four exemplars and look at what it means to “work with business” and “what do we build with"

- Not Business, Enterprise, Solution, Application
  - focus on different types of design that I've done in those roles

## System As Application

### What we want to do

- The relationship between **value, requirements, application architecture, implementation, and NFRs**
- Business: product owner, domain experts, users

### What is possible to do

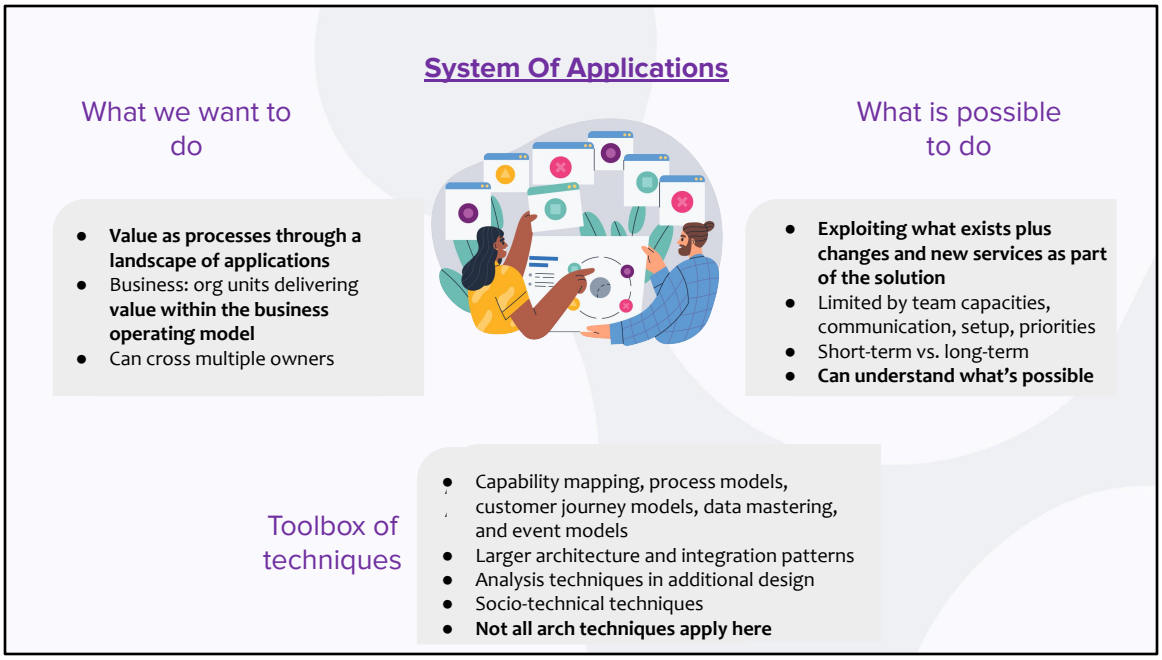
- **Concepts provided by the technology**
- Small number of aligned teams
- Best way to learn is to DO – by implementing with that technology

### Toolbox of techniques

- Techniques are based around building blocks:
  - CRC cards, UML / C4, design patterns, DDD
  - Stack-overflow answers
- **Majority of what we talk about**

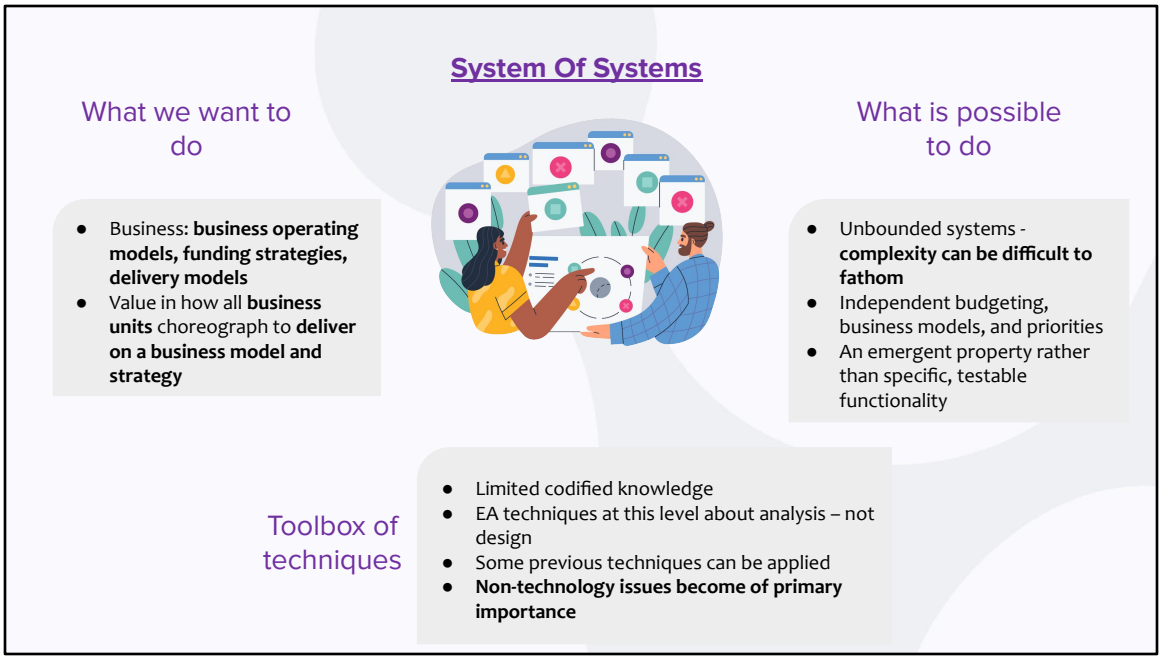
- System as Application - usually start here
  - Intro (if time)
    - Started with designing applications.
    - might have integrations other places but to some extent a standalone system
    - Often building something from scratch or updating a single app
    - Standalone app with one or two teams working on it - or nowadays the set of microservices providing a function within a bounded context
    - business has shared understanding because you can talk about the functionality in terms of what they see in the interface
  - What we want to do:
    - Work with the business meant understanding the relationship between value, requirements, application architecture and implementation
    - Qualities of the application.
  - What is possible to do:
    - We build with the concepts provided by the technology
    - The best way to understand what is possible is obviously by implementing with that technology
    - Cynefin obvious: sense–categorize–respond
  - Toolbox:
    - Our primary architecture techniques and tech specific techniques are based

- around this area. (Visualisation, design patterns, etc)
- CRC cards, domain modelling



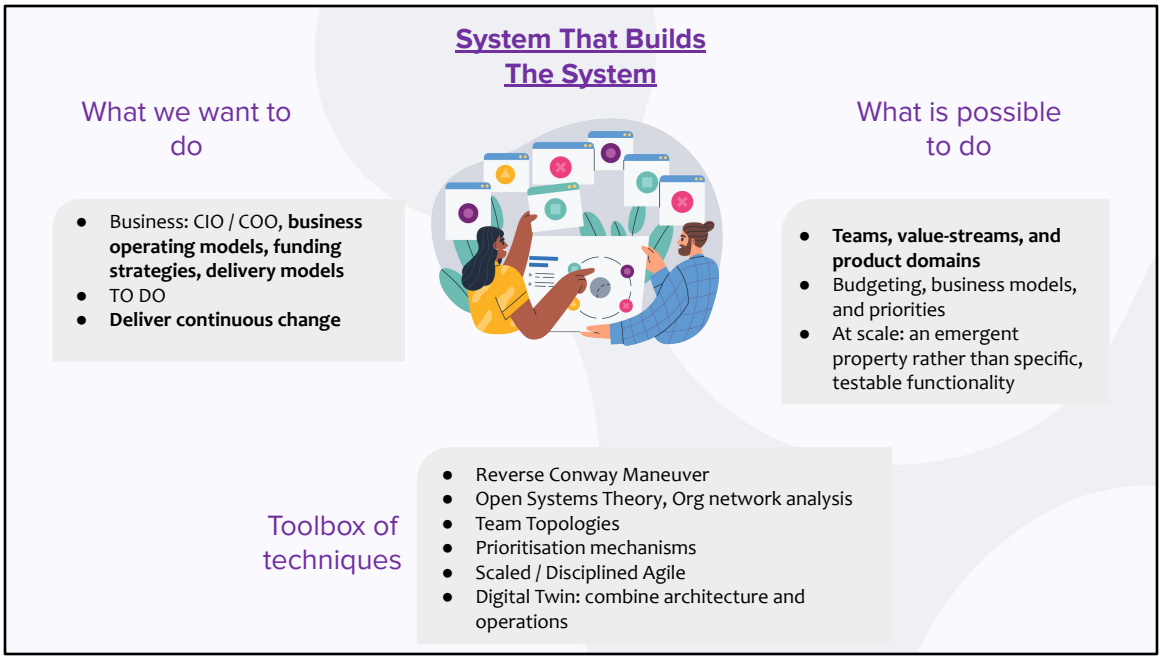

- System of Applications - starts getting more complicated
  - Intro
    - No conscious decision to move away from coding and app arch, just started working with solutions that span multiple applications.
    - Often building solutions in an existing landscape of applications in some enterprise. Or a core system replacement that is central to many business domains
    - Landscape is bigger and specific teams often don't see the whole picture because they're so focussed on their app(s)
    - Still possible to understand the whole but you need to be dedicated to it.
    - generally start working earlier in the process and crossing more of the landscape.
    - More analysis work. What are the impacts of this design
    - Collaborating with application architects
  - What we want to do:
    - usually defined by business units rather than a single owner.
    - Working with the business means finding out how initiatives map to applications. Business doesn't just want the value provided by one app, they want value as journeys / processes that flow through multiple solutions. Often different owners with different flows through the same solutions
    - Business doesn't understand how it all hangs together and neither do most

        - IT people
    - Whats possible to do:
        - Often a mix of exploiting what exists as well as building small extensions or new pieces that take part in the solution
        - not just code, its concepts / abstractions for the business but limited to what's possible in tech.
        - Processes, events, mastering of data, gaps & overlaps between applications, quality of the whole.
        - Properties of multiple individual applications added together to provide an end-to-end solution. End up with multiple solutions across many applications.
        - Historically we've been breaking down monoliths that did all of this into smaller apps, usvcs that do pieces. Great that they can evolve more independently but its harder to understand how the whole hangs together.
        - Possibilities now limited by team capacities, team communication, team setup, team priorities (when and how)
        - Sociotechnical aspects become more visible
        - Cynefin complicated: sense–analyze–respond
        - Not primarily based on concepts provided by tech. Therefore, keeping up through implementing is not as important
            - But still need to understand and work together with app arch teams
    - toolbox:
        - also get into capability mapping. Long term change. Short-term v long-term approaches. Intentional technical debt
        - Capability maps. UML can still be relevant (especially the behaviour parts). Process models, customer journey models
        - Archimate, TOGAF techniques.
        - Toolbox is more about bridging applications and analysing the what we want to do
        - Toolboxes for design techniques are limited. Not much industry advice.
    - Lots of opinions about architecture (and agile) often come from system as application and blindly assume that they are relevant for systems of multiple applications. And they don't fit



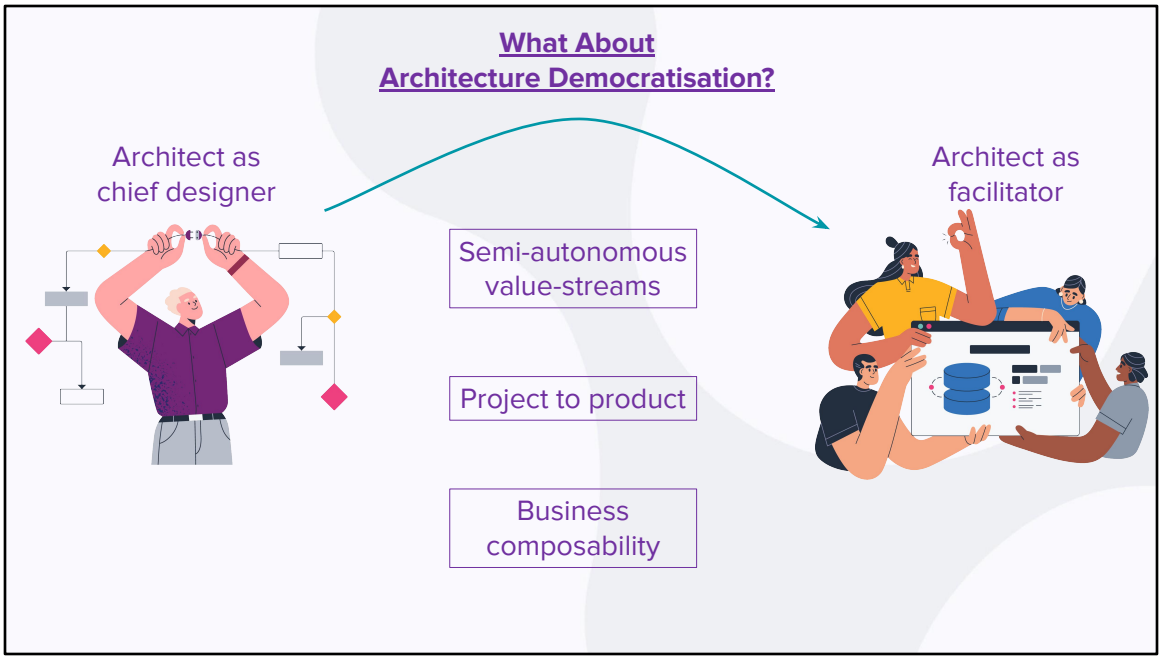

- Intro
  - Previous job spanned 6 countries with about 20 distinct business units and brands.
  - Global strategy moved from completely independent countries and business units to operating as s single company across the region - “can you go and figure out how to change the IT architecture for that?"
  - same principles apply for companies with a lot of M&A activity or independent business units / product areas.
- What we want to do:
  - working with the business includes business operating models, funding strategies, delivery models
  - Not one individual customer journey for one business unit. But how do all business units choreograph to deliver on a business model and strategy
- What is possible to do:
  - Level of complexity is difficult to fathom - unbounded systems. One person can’t understand how it all hangs together
  - Have to deal with more autonomy between the systems - can have totally independent budgeting, business models, and priorities.
  - Meeting ‘what we want to do’ is often an emerging property rather than specific, testable functionality
- toolbox
  - Toolbox for system as a single app is (almost) irrelevant when working at this scale

- Not much codified knowledge
  - Enterprise architecture work at this level if often about analysis - not design.
    - Which is useful, but not complete
- Product families, product lines, multi-tenancy
- SEI system of systems work
- Some techniques also scale up to this level - e.g., DDD. Techniques that are based on theories that can be linked to cognitive models
- Non-technology issues become of primary importance - sociotechnical, human-factors. But its not codified in a way that's directly useable for architects
  - No real software architecture body of knowledge



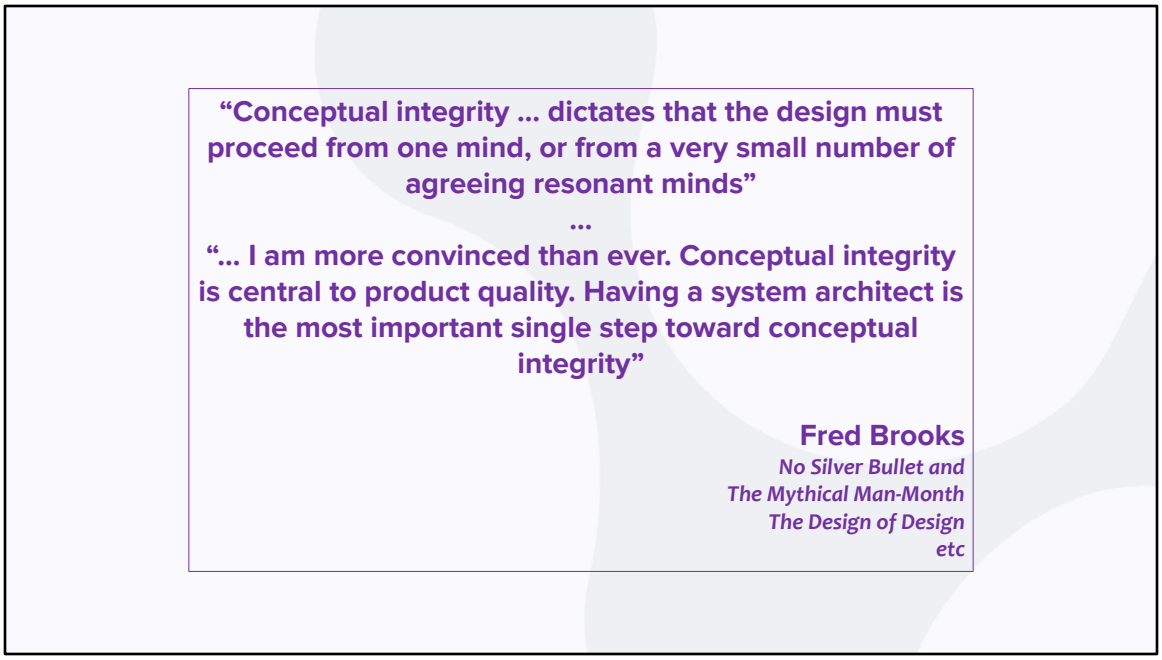

- System that builds the System - becoming more prevalent
  - intro
    - Architects working more and more with Way of Working. Agile approach to EA. Try, learn, improve
    - Design of the enterprise to improve change delivery. Especially companies that do a lot of their own development rather than procurement
  - what we want to do
    - adapting to continuous change -> away from plan-build-run
    - Throughput, time-to-value
    - employee empowerment, autonomy, and productivity
    - improve quality. People with the best knowledge make the decisions
    - also about architecture. You knowledge deepens on all these aspects (possible to do, techniques, want to do). You can't know it all up-front. Working in an enterprise for measured in years.
  - what is possible to do
    - teams, value-streams, domains. Building blocks of how we organise work and measure its progress (events)
    - prioritisation mechanisms
    - investment mechanisms
    - similar to previous but more about the dev/maintenance rather than operation

        - mechanisms for type of work - cognitive load
    - toolbox
        - reverse conway manoeuvre
        - open sociotechnical systems
        - open systems theory (or sociology tools)
        - organisational network analysis
        - Team topologies
        - scaled agile techniques (Disciplined Agile, SAFe, etc)
        - Digital Twin of the Organisation for visualising on the ent arch
        - real danger with cargo-culting here.




- How does the trend of Architecture democratisation affect this?
  - Builds on the System that build the System section
    - what is consequence of moving away from plan-build-run?
  - number of trends in the industry
    - Project to Product
    - semi-autonomous, x-functional teams teams or value-streams
    - business composability for rapid change
  - Do these invalidate any of these types of design.
  - All these trends have an underlying principle ( moving from a single, centralised decision-making architecture function to a decentralised architecture function where decisions are made.
    - Also removing the strong business / IT distinction. The people who operate the capability / value-stream / domain are also responsible for its evolution
  - many tweets, blogs, linkedin articles talking about the trend of the centralised architect / team moving from chief designer to facilitator / coach for the arch work happening in these de-centralised teams.
  - Supporting the democratization of EA.
  - What does this mean for the anatomy of design that has been presented.
    - What happens is still the same, but when and how it happens is evolving.
    - The key is understanding that teams are semi-autonomous, they are not autonomous. What are the dependencies?

- Architect as facilitator is not a complete shift, more of a change in primary emphasis
  - Not everything fits nicely in a value-stream org model Teams need to know where they fit into the whole. Where they can make decisions and where it impacts others.
  - "The architecture" for the company goes from centrally produced and consumed to being something that is mass-consumable and self-service creation.
  - Process goes from inside-out to outside-out.
- Architect is providing structure to collaborations rather than just structure to the building blocks
- But one thing that doesn't change - I mentioned the "What" of architecture must remain




- The role of architect is to provide the conceptual integrity
  - I've done all these types of design in my career and every single one of them comes back to this. You might be on a similar career path. Don't fall into the trap of thinking building blocks and techniques are the most important thing. They're a toolbox but they are not what you're trying to build.
  - always good to quote Brooks. Also The Design of Design
    - people will look at you sagely
    - and there is enormous depth in his aphorisms.
      - Brooks law. and No Silver Bullet.
    - One of the most important things he's written doesn't get enough air-time. Its all about what happens in the synthesis of the design aspects we've spoken about.
      - The important of conceptual integrity in design.
  - Design must be logical consistent and cohesive
    - must make sense and fit together
  - Brookes: "I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor or representing it and testing the fidelity of the representation"
  - Brooks: "I am more convinced than ever. Conceptual Integrity is central to product quality. Having a system architect is the most important single step toward

- conceptual integrity”
  - think this is in the forward to the 20th anniversary edition of MMM book
- This part is harder to to talk about than building blocks, BOKs, and techniques. More fluffy. But it is absolutely the key part.

**"To understand a problem means to understand its difficulties; and to understand its difficulties means to understand why it is not easily soluble – why the more obvious solutions do not work. We produce the obvious solution and then criticize them, in order to find out why they do not work. In this way, we become acquainted with the problem, and may proceed from bad solutions to better ones – provided always that we have the creative ability to produce new guesses, and more new guesses.**

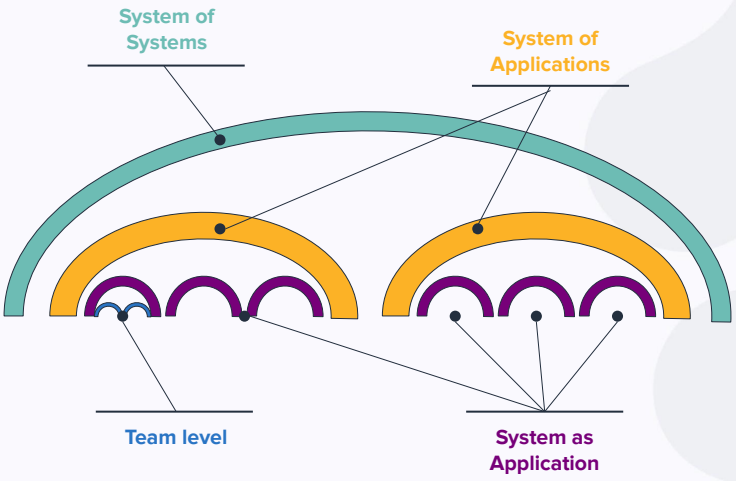
**...**

**If we have been working on a problem long enough, and intensively enough, we begin to know it, to understand it, in the sense that we know what kind of guess or conjecture or hypothesis will not do at all, because it simply misses the point of the problem, and what kind of requirements would have to be met by any serious attempt to solve it. We begin to see the ramifications of the problem, its subproblems, and its connections with other problems."**

**Popper, K. R.**
***Evolution And The Tree Of Knowledge.***
***Objective Knowledge: an evolutionary approach***
***Oxford University Press. 1979***

- Popper - how do achieve conceptual integrity
  - never seen a prescriptive process of SW design that helps. Being certified will not help you.
  - This is the only thing I've read that comes close. From the philosophy of science. How do we build explanations about the world around us.
  - Its just a continuous process of understanding the problem domain, the solution domain, and testing them against each other.

## summary




Design happens at different levels and can cross different business / IT scope and impact different spans of time

Each has a different consequence for cost and benefit

They all require conceptual integrity for the design you are creating

Adapted from https://twitter.com/ruthmalan

- Seen lots of different types of architecture
- Difficult to give software arch advice that is applicable to the the whole range of software architecture situations
    - When giving your advice / opinions, think about the context you're in and which contexts you haven't yet experienced
    - There is always a bigger context to architecture that your current situation.
    - When commenting on other types of architects consider whether or not you're been in that context
    - Consider whether you are focussed on what the business wants to achieve or only on making use of the techniques and building blocks


ardoq
Design
from Application Architecture
to Enterprise Architecture
Jason Baragry
Chief Enterprise Architect