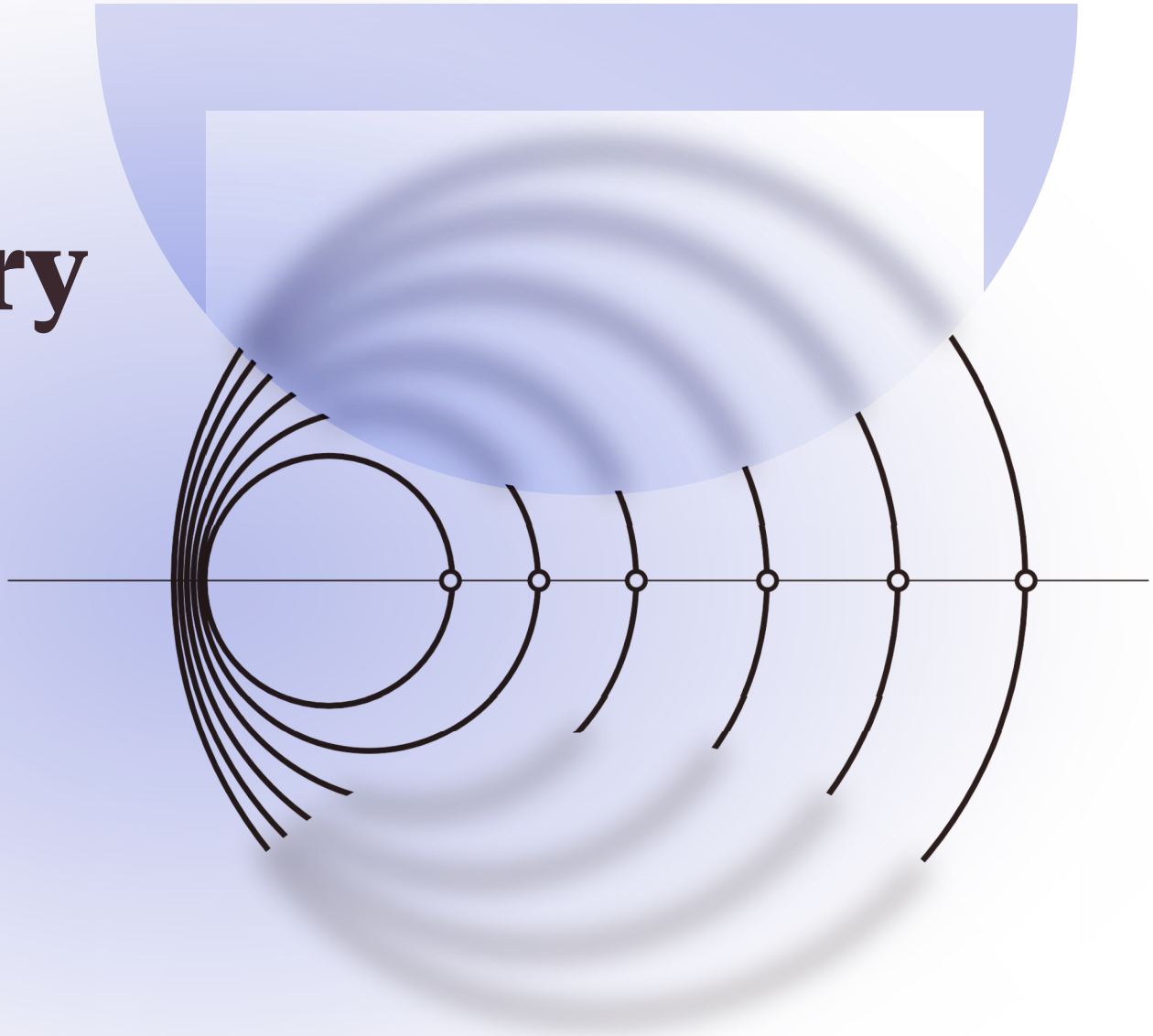
SCENARIO BOOK

Contemporary Architecture



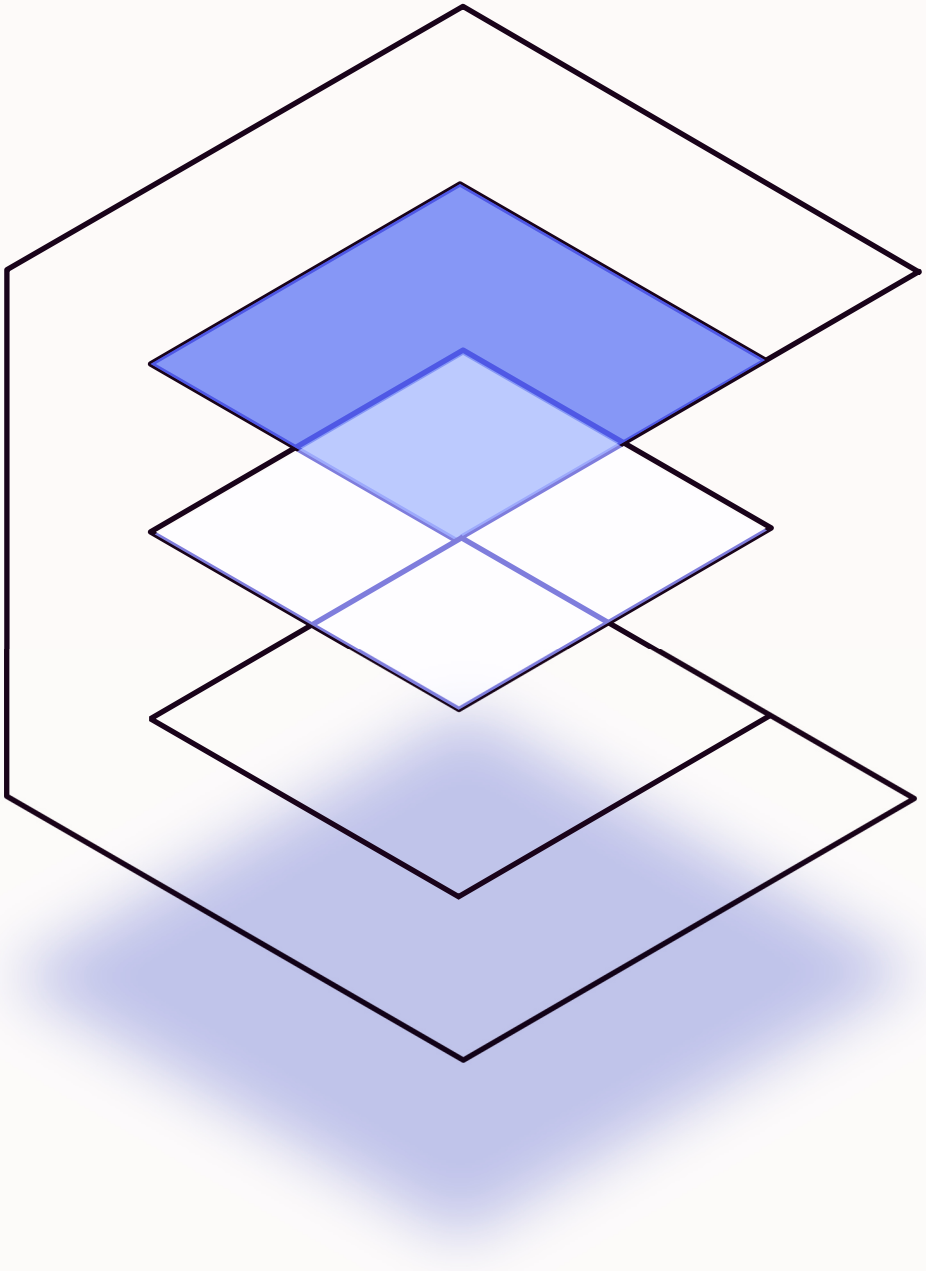
Architecture can be one of the most significant constraints on the flow of value and business growth. How do you make sure that your architecture constantly evolves to meet customers' needs?

HYPR

Your scenario?

- You realise that architecture has a major effect on how quickly your enterprise can adapt to the market and general environment
- You're concerned that your architecture is failing your customers because everything seems to take so long to deliver. It's taking months to release new value and changes to existing software takes ages too
- You've noticed that testing happens at the end of delivery, exposing fundamental errors and security flaws too late in the day
- It's obvious that your system cannot be easily maintained. You can't recover issues rapidly and the system uptime is starting to drop
- Deployment feels 'dangerous' and makes you nervous

"We need architecture that doesn't get in the way of delivering value to customers at the speed they expect. We're at the point where delivery feels sticky and sporadic and when we do deploy, we're always 'on edge'"



Why does architecture become a problem?

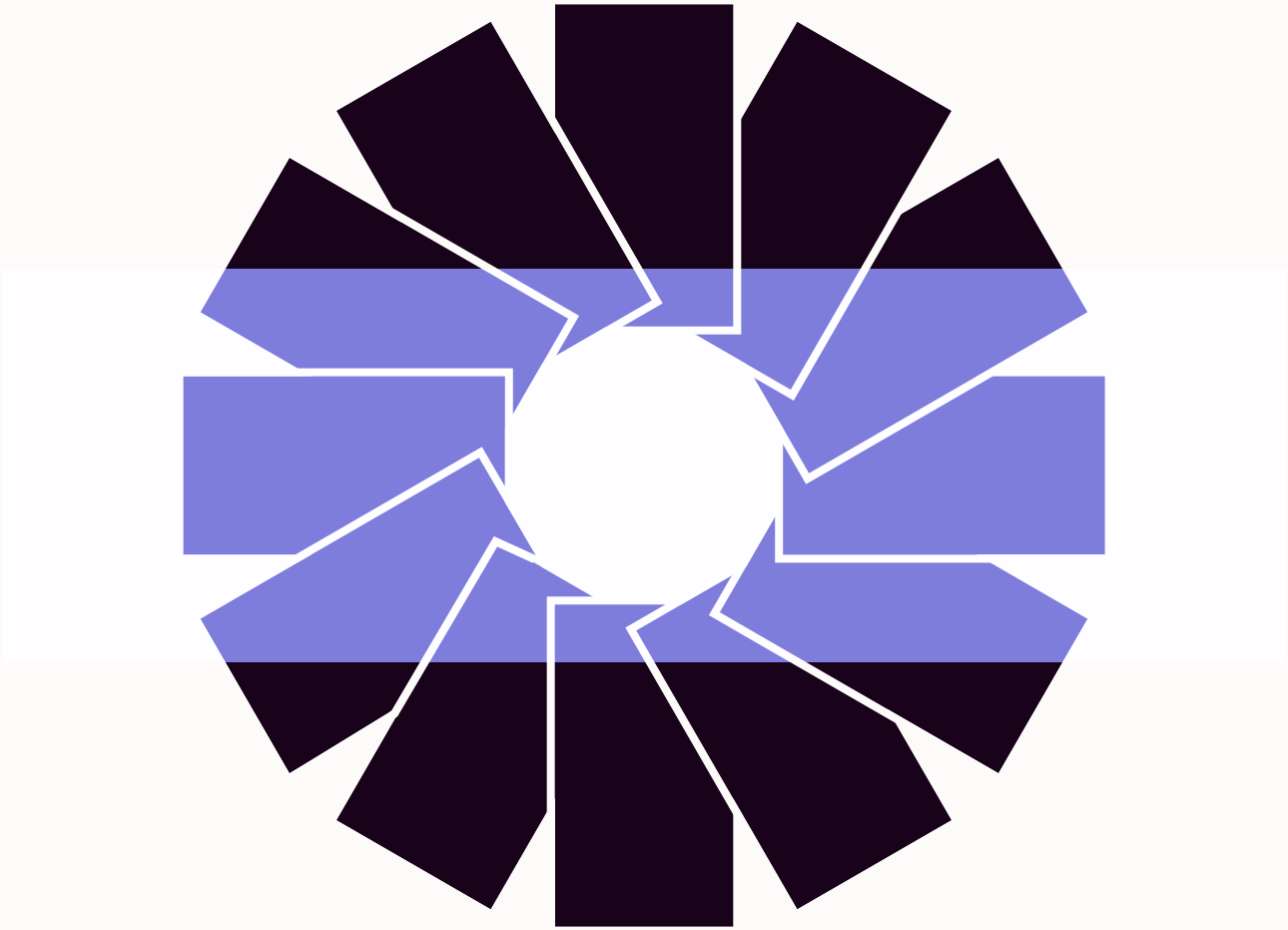
As technology has evolved, there have been significant changes in the architectural patterns that must be used to enable systems to adapt effectively to customer demand. Many enterprises have not modernised their architecture with the new patterns with the following results:

- The system was built around rigid technologies and architecture which could not adapt easily to change. Monolithic architectures – which were completely appropriate at their inception – have become brittle. They are creaking under the strain, it's harder to deploy safely and the risk of failure increases
- Businesses continue to prioritise the meeting of immediate customer needs over investing in good engineering practice
- Delivery of initiatives under resource pressure and at a tactical level simply adds layer upon layer to the architecture, increasing complexity and constraints
- The technical debt burden has reached unsustainable levels. It's becoming harder to realise value. Workarounds and band-aid approaches store up problems for later
- Enterprises often struggle to make the right investments to modernise architecture because boards and CEOs are conflicted by the high costs of change and the time it takes. An architectural journey of evolution isn't easily understandable to them
- Delivery teams get demotivated working on old tech in old ways

What good architecture looks like

The modern architectural patterns that should be used to enable systems to adapt effectively and at lower cost include:

- The chosen architectural style should meet the constraints of the system and anticipated performance and scale required to support customer growth
- Applications should be cloud-native, organised around business domains with appropriate distribution. Event-driven architecture supports larger systems and reflects separation of concerns and scale
- Containerisation should be used to support developer productivity and fast feedback
- Applications should be deployed using modern CI/CD and progressive delivery techniques with automated infrastructure, deployment, security, compliance, testing and release
- Quality, reliability, observability, modularity and security should be built into the system rather than treated as an 'add on'. For example, in respect of quality, building in layered testing automation as work progresses
- Low coupling, high cohesion and excellent dependency management help ensure the architecture can continue to be evolved rapidly
- Any specific algorithms should be relevant and well-implemented
- For highly-scalable systems, patterns to allow resilience to failures or outages of subsystems or third-party systems must be in place



Making a case for change

The technical nature of moving to a contemporary architecture often makes it hard to sell in to boards and CEOs. And that's before you start talking costs and timeframes! In our experience, it's much to do with the language used and a lack of focus on business outcomes. Here are the high-level themes we recommend in making a case:

- When it comes to accelerating the flow of value to customers and supporting business growth, the number one priority should be ensuring the architecture is up to it. Architecture is your superpower
- Architecture is getting in the way of things you need to do today and this will only get worse tomorrow
- The architecture does not allow you to release working code safely to production at will
- The architecture cannot adapt to the changes in technology so you're missing opportunities to deliver value
- The architecture is unable meet your anticipated growth
- Existing architecture is not organised around your business context or business domains. Making it so will allow you to both improve the value you deliver and make it much easier to link that value to business outcomes. Imagine if your system could be understood from a business and technology perspective using a ubiquitous language
- You have significant technical debt that you need to actively manage and pay down. It's time to stop the workarounds
- The cost of maintaining and operating your architecture is rising
- The risks you face in the system are increasing

Helping you make change happen

We're experts in helping enterprises transition from old to contemporary architectures. Through our work, we have matured a set of patterns and approaches that help us understand what good looks like in your context, the journey you need to take and how, together, we can navigate to the destination.

Discovering your context

We use Situational Analysis to discover, among other things:

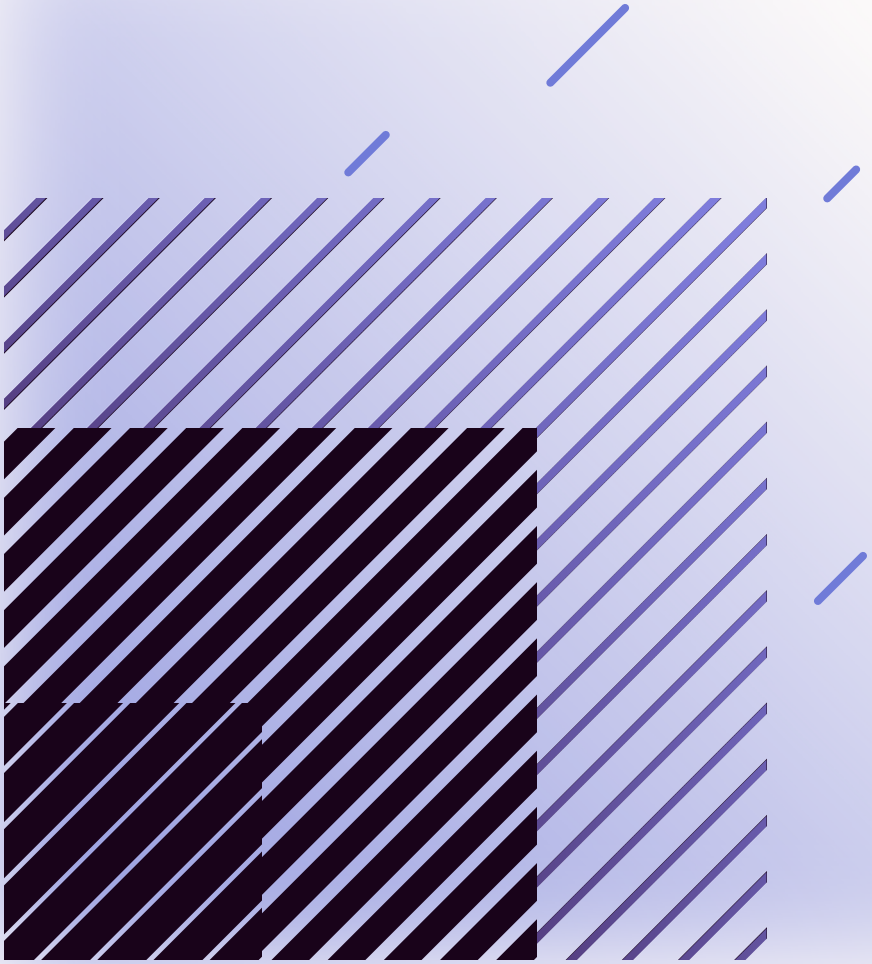
- Your vision for the business and how tech investments are aligned to it
- What your system and architecture needs to deliver for your business
- Constraints in your existing architecture and how it compares to modern systems
- Your technology stack and toolset
- Your code, data and quality practices

We use our Improvement Model to baseline your system (and practices) against benchmarks. It shines a light on the things you need to act on.

Identifying your priorities. Finding the place to start

There may be many things to act on, but there's always a priority. Your starting point for changing architecture should be a high-value business service or domain that represents the 'fastest path to value'. We help you find this 'slice' of value.

We then create a high-level model for that service or domain (based on a product or Value Stream) and identify the dependencies and constraints impacting it across the tech stack. This model shows you the path for transition...



Transitioning your architecture

We take a transition approach to change that carefully and systematically extracts you away from older technology. We have a toolkit of patterns and techniques that help do this in the right order.

We usually retain at least some elements of existing architecture and augment it with new approaches, techniques and technologies. On rare occasions, we might recommend building from scratch.

Our goal here is to prove material improvements in time-to-market and quality. Once one domain has been extracted, we move to the next.

Features and benefits of the transition approach

Listening to reality

Traditional software development functions are now being planned in smaller, less risky iterations with feedback loops being relied on to 'listen to reality' and guide direction. Your architecture needs to go the same way. Rather than developing large, expensive, upfront plans for architecture based on previous experience or assumptions, our transition approach allows you to learn and adapt as you go. Architectural change only happens when reality tells us it is needed.

Experimenting at low cost

Experimentation and evolution over time is less expensive than planning an upfront architectural design for what might happen. With some key feedback loops in place such as test automation, CI/CD, system observability and automated benchmarks, you can run small experiments of small incremental changes to see reality.

Containerisation and cloud computing such as Azure and AWS make this experimentation faster and far less expensive. These experiments enable your delivery teams to develop the capability to evolve architecture in ways that focus on the delivery of value to your customers.

Allowing ambiguity to create a sense of direction

Our transition approach respects the concept that architecture has a type of 'archaeology' – evidence of previous technical practices and architecture are left behind as valuable artefacts. To support evolution and experimentation, architecture must be *allowed to diverge* and, in doing so, develop some degree of ambiguity. The antidote to this ambiguity is a strong sense of direction. Everyone knows where they are currently taking the architecture and where they definitely don't want to take it.

Meeting ongoing, changing demands

Even with just a few priority services or functionalities extracted and moved to a contemporary world, the hybrid system can already be reconfigured to meet changing demands. This unlocks additional options for the business to choose from.

Focus on customer value

Architecture is evolved when it is required to deliver value to end users. Delivery teams are empowered to tackle the constraints that are slowing them down.

Better planning

Future architectural changes become business-as-usual for software delivery teams. The costs for changes are included in estimates for planned work.

Reduced risk

Small experiments, iterative architectural change, JIT delivery and maximum buy-in from developers significantly reduces delivery team dependencies and possible bottlenecks.

Technology staff retention

Valuable learning opportunities for teams, reduced frustration through increased empowerment to make significant change, higher level of team engagement with less centralised architectural decisions.

Encourage creativity

Reduced cost and risk of experimentation, increasing the speed and accuracy of feedback and an accepted level of inconsistency allow for faster evolution and more creativity.

Is our approach for you?

Evolutionary architecture can be applied to most systems. For example:

01. Agile technical practices such as Test Driven Design promotes lean code, simplicity, separation of concerns, encapsulation, layering and indirection which enable the evolution of architectural patterns in code

02. Automated Testing and Automated Architectural guide rails run as part of a CI/CD delivery pipeline can be used as timely feedback to developers on the fitness of every incremental change to a system

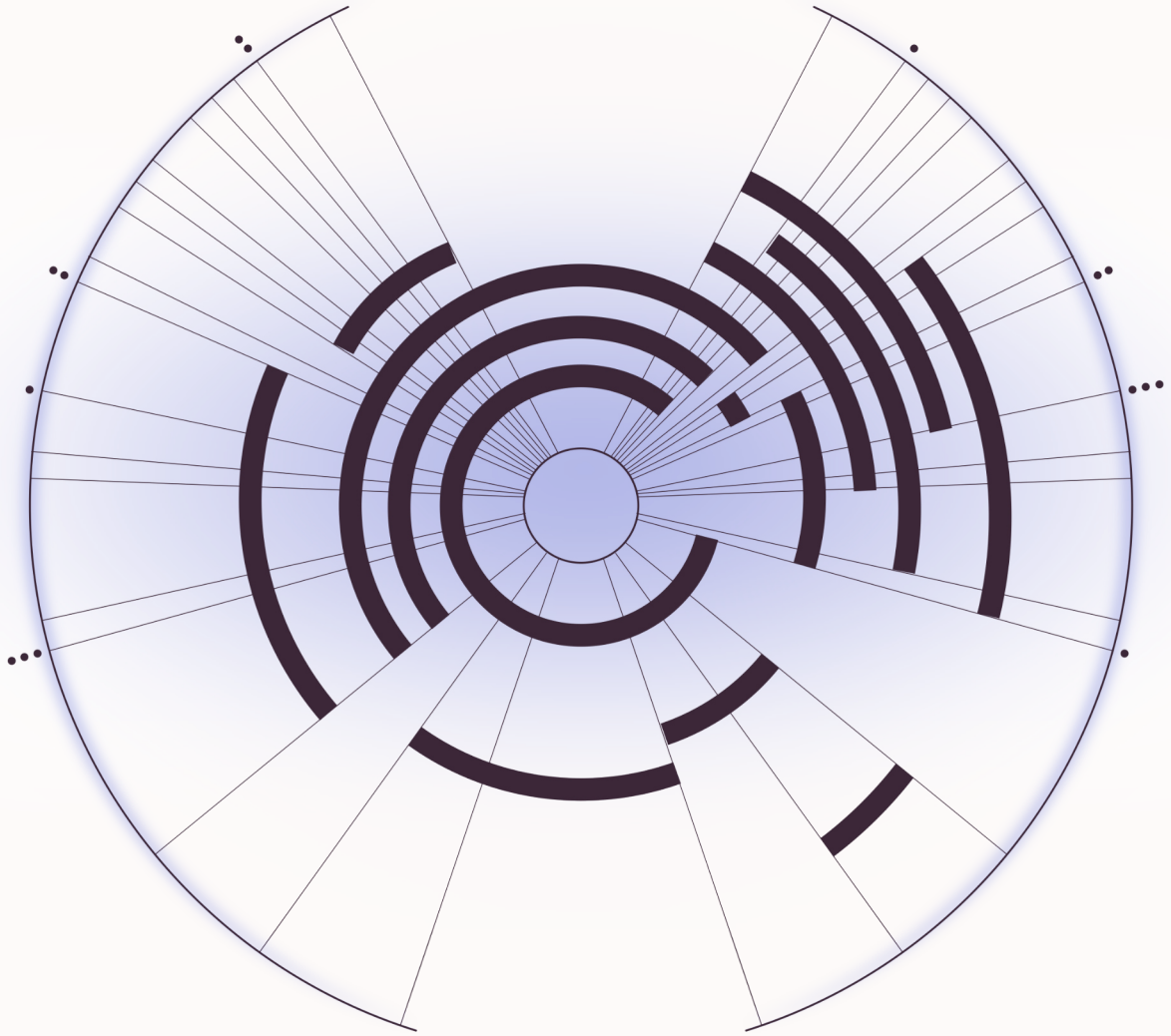
03. Ongoing architectural experimentation can be planned as business as usual, with learnings used by delivery teams to make informed decisions and set architectural direction

It's more than likely that evolutionary architecture can work for you too.

Outcomes

Identifying and extracting slices of customer value in this way transitions you over time to a contemporary architecture while continuing to deliver for customers. We find that over a period of months (not years) that we can unlock a significant amount of value.

- You are able to release high-quality, working code at will and improve your ability to adapt to new ideas, new technologies and new customer demands
- Your new architecture is free of significant constraints and is no longer 'in the way'. Flow of customer value is significantly accelerated
- Finally... 'The plumbing is nothing without the house' and 'the architecture is nothing without the teams of people using it to build platforms and products that make customers happy'. *Architecture is the beating heart of any technology estate and it requires the nervous system and cardiovascular system to work too. We cannot stress enough the importance of people and technical architecture in unlocking value*



Why HYPR?

We can help you keep you on the right side of technology change and make the decisions that ensure your system accelerates the flow of customer value. Call us now...

What makes us different?

Focus on flow – Progressive enterprises are focusing on finding and removing delays from their system through the practice of Value Stream Management (VSM). We're a leading VSM consultancy helping enterprises in NZ and Australia.

Systems thinking – We take a systems-thinking approach to avoid local optimisations that contribute little to the whole.

Focus on your people – Technology and people are one system and two sides of the same coin. We focus as much on the social constructs and human networks as we do on the tech.

Transition not transformation – Your enterprise operates in a VUCA (Volatile, Uncertain, Complex, Ambiguous) world. It needs to keep flying while making changes. We know from experience that transition is the only way you can do both.

Our people – We're a diverse team with shared purpose and values. We have extensive skills across our consulting lines, from the very best software engineers to strategic experts able to engage at board level. They have lived at the coalface of change.

We're ready to help

HYPR

Is your architecture constraining the flow of customer value? Are you worried about the costs of moving to a contemporary system? Our transitionary approach can help. Call us now...

Justin Tomlinson – CEO
justin@hypr.nz
+64 221 693 359

Gillian Clark – Director
gillian@hypr.nz
+64 21 642 079

Gareth Evans – Director and Chief Engineering Officer
gareth@hypr.nz
+64 21 0227 6744

Ajay Blackshah – Chief Practices Officer
ajay@hypr.nz
+64 21 747 633

Written by our team
Edited by Sian Hoskins
Design by Anam Berdugo, HYPR Creative Director

Thanks to all the clients and 'Friends of HYPR' who provided feedback and the pioneers of ideas and models that help us see things in new and different ways.

HYPR INNOVATION

GENERATOR NZ
Level 1, 22-28 Customs St East
Auckland Central
PO BOX 106-229
Auckland City 1143

www.hypr.nz