From restrictive monolith to event-driven powerhouse

Supporting a national retailer to adopt best-in-breed technologies and development practices, through a stepped transformation that maintains continuity of crucial business processes.

EQUAL EXPERTS

Every day around Australia, many thousands of shoppers visit this retailer: whether in a physical location or through their online store.

These shoppers browse and buy from an enormous inventory of items with constantly evolving variations in size, style, and localised availability. Stock levels are constantly fluctuating, and every shopper—and store within this retailer's vast retail network—relies on immediate, accurate visibility of what they can buy, when they can expect to receive it, and how it will be shipped.

Operating a leading retail business at this scale requires sophistication, immediate and reliable data, and robust, scalable platform design. By supporting their transition to an event-driven architecture based on microservices, we're proud to help **consolidate the team's reputation as one of Australia's biggest, most beloved, and forward-thinking national retail brands.**

This case study will help you to understand:

The crucial role that event-driven architecture can play for large scale retail businesses

How large organisations can gradually embrace new technologies without completely disrupting historical business processes

The value of stepped change in revitalising technical systems and the development practices that support them

Table of contents.

01

ABOUT THE RETAILER

Part of Australia's largest retailer group, this partner is a major player in the 'discount department' shopping category.

Stocking a broad range of over 100,000 individual stock-keeping units (SKUs)—with products from high-profile national and international brands like Apple, Samsung, LEGO and FIFA—this retailer is a one-stop shop for clothing, health and beauty needs, electronics, furniture, homewares, and more.

With massive customer visitation across all channels—and a rapidly growing ecommerce presence in the wake of the COVID-19 pandemic—their technology needs are as varied as their impressive product inventory.

4 billion

annual turnover.

180

stores around Australia.

22,000

employees.

179

stock pickup locations.

02

Originally, there was an implementation of SAP Hybris as a monolithic core system.

While SAP was a fast and cost-effective way to initiate a move to e-commerce in 2008, their needs had evolved over the course of a decade. Having been heavily customised over time—and reaching its scaling capacity—the SAP Hybris system had begun to approach end of life, causing pain due to the limitations of its monolithic architecture.

For example, in peak sales periods like Christmas and the annual toy sale, the entire architecture suffered under extraordinary pressure. As a result, there were frustrations across the board, with consequences for customers, as well as their tech, sales and marketing teams.

While the monolithic architecture offered useful early-stage capability in e-commerce, it no longer matched their requirements for step change of innovation. As a variety of sales channels grew, it became harder for the team to quickly create new services or solutions, or flexibly integrate with third-party platforms to save on green-field development efforts.

Constrained by the limitations of the monolith, rapid change was a challenge.

Provisioning change was time-consuming and laborious. Working within the monolith, a typical deployment lifecycle involved two-week production releases, followed by significant UAT, regression and performance testing. All told, this translated to a three-to-four-week cycle for all deployments. The painstaking process was required for any change: even small typos in code.

To provide first-class customer service, they need flexibility, the ability to scale and pivot in response to rapidly evolving business requirements, and an architecture to facilitate growth and evolution.

Almost every release required database schema changes, which meant the website would need to shut down entirely for any release. This typically occurred from 11pm to 4am every fortnight.

Aside from the interruptions to customer experience and impact on prospective sales revenue, these outages created an unexpected—though equally crucial—consequence: the potential for team burnout.

With our partner's technology team working around the clock—often to deploy relatively low-impact, routine changes—the team was at real risk of burning out. While this is a longer term and less direct consequence than something like loss of revenue, it's a crucial consideration for morale, productivity and talent retention. It's also unsustainable from a purely economic perspective: left unchecked, ongoing burnout can ultimately create a negative impact on any business' bottom line.

03

Some of the crucial, and most constant, questions for every store are: do we have this stock available to promise to our customers? Where is it available? How will it be shipped to them?

In addition to being difficult questions to answer, each proved a massive strain on the existing technical architecture.

The sophisticated rules involved in answering these questions (referred to as Availability, Sourcing and Fulfilment capabilities) were extremely resource-intensive for the legacy monolith. For example, around 40% of the total calls processed in the monolith are related to product availability queries.

The answer to the crucial availability question—'is this product available?'—is constantly changing. This can be impacted by a range of factors, including:

- Successful purchases made both online and instore, and the subsequent impact on stock levels
- Digital shoppers placing items in their cart, then exiting the checkout process before completing the purchase
- People grabbing items off the shelf in physical stores and either purchasing the items or placing them elsewhere in the store without completing checkout
- Items being lost, damaged, or potentially stolen

The vendor or location of a particular stock item. For example, our partner —like many large retailers— use drop-ship vendors for certain items. In some cases, products may be held in alternative third-party locations (a vendor might hold and deliver all trampolines to help warehouses save on space, for example), but still need to be catered for within the available inventory. Some stores will use a warehouse attached to a retail centre, while others have stand-alone distribution warehouses.

In short, the retail metric of 'availability to promise' (ATP) is a constantly moving target. Regardless, it is absolutely crucial that it's accurate. **Event-driven architecture offers real-time stock visibility** with immutable precision, while securely surfacing data for multiple business intelligence units.

Accuracy and rapidity of information is critical. If the ATP number is too low, products will be incorrectly listed as 'unavailable for purchase', needlessly foregoing potential sales.

On the other hand, if the ATP is too high, they risk:

- overselling,
- an inability to fulfil customer orders with correct items,
- cancelled orders, and
- general customer dissatisfaction.

To provide an accurate representation of inventory levels, the partner was using a feed of real-time events pulling various datapoints from point-of-sale, warehouse stock levels, and other data sources throughout the business.

These feeds—called event streams—help to calculate a real-time, constantly updating the representation of stock, defined as "total inventory, minus confirmed orders minus buffer" ('buffer' counts for people with items on the way to checkout that have not confirmed a complete purchase, for example).

Thanks to the event-based nature of the architecture, the data is immediate, can be consumed by multiple microservices and capabilities, initiates a wide range of automated business processes, and is **immediately served to multiple business intelligence units**—in the format they require— for necessary action or processing.

The architecture runs within the Microsoft Azure cloud platform, with events managed via Microsoft Event Hubs. Consumed events are written to a NoSQL database, Azure Cosmos DB, with a cache layer over the top to ensure the solution remains highly performant.

In terms of the architecture itself, the main storefront for ecommerce is the customer facing website. The customer website calls into the monolith—or at least the remaining capabilities contained within the monolith, while we're in the process of strangling individual capabilities out as microservices—with all of the logic.

As part of the strangulation process, we've created a method for partitioning and processing subsets of customer traffic based on specific criteria. For example, we can implement rules
to stipulate that all orders being fulfilled within South Australia are processed via the new microservices capabilities, while all other states are managed via the legacy capabilities housed within the monolith.

This approach—while providing granular control—allows the team to effectively 'learn in production' with comparatively lower levels of live orders, and thus minimal risk to both the end-customer and the retailer. This is crucial; even with enterprise level-quality assurance processes, you can never discount the value of seeing systems and architectures in practical application. You need data; you need live orders to process as a means to validate and identify anything that requires re-thinking or iterative enhancement.

Inevitably, the best way to endorse a system is to use it within a controlled environment.

04

For many organisations with this scale and profile, there's typically a requirement to maintain some legacy infrastructure, technical systems, and business processes. Change is a gradual and guided process.

To strike the right balance between major change and maintaining continuity—a common prerequisite in enterprise-level digital transformation —we collaborated with the team to refine a list of driving principles and evolve their architecture through a strangler design pattern. Learn more about prioritising effort in software migration.

Together, we co-designed a list of objectives for the new solution and technology team's production dynamic. Our objectives identified that the new environment and any associated development practices should:

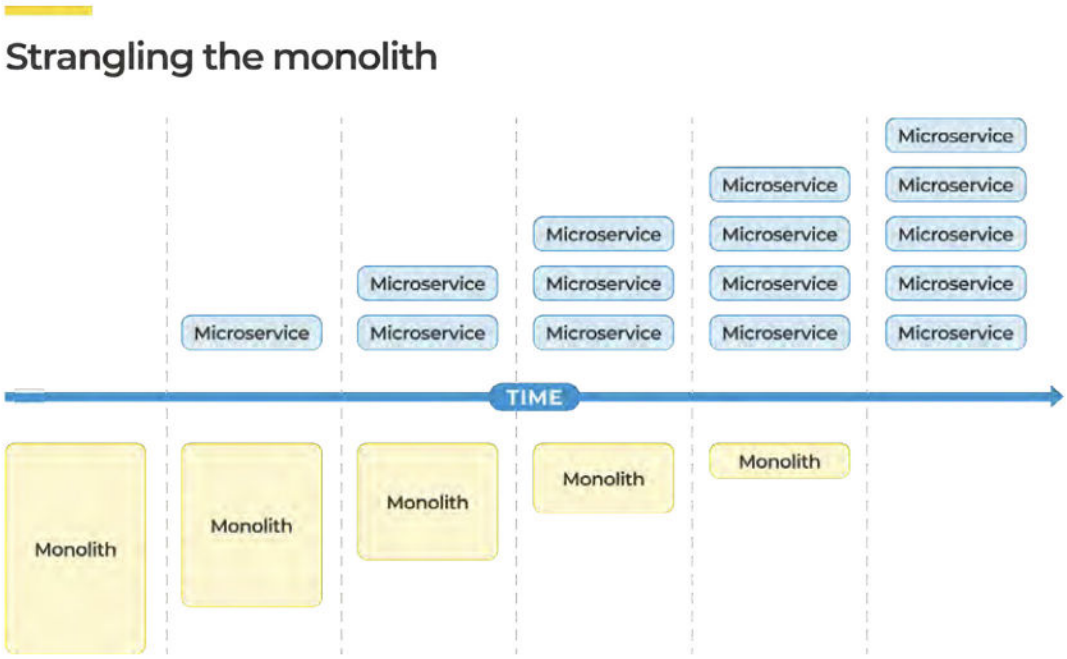
- Drive innovation based on what matters to customers
- Stay pragmatic and commercially focused
- Borrow, buy or build as necessary, depending on requirements
- Be directionally right, but specifically wrong
- Inherently maintain the ability to evolve, test and deploy

05

As with any major system transformation, there were decisions to be made around buying off-the-shelf, building green-field solutions from scratch, or borrowing from existing componentry within the wider group.

Buying an entirely new overarching system to replace the existing one was always an option our partner could consider. However, it was discounted relatively early in the process as it would limit the team's ability to innovate quickly and test new services. Ultimately, this was too great a tradeoff for a brand striving to consolidate its reputation at the forefront of in-store and digital customer experience.

At this scale, there's a fundamental requirement to continuously test, trial, learn and improve. This dynamic is simply not feasible within the constraints of large, standardised product packages.



By engaging with our partner early, it became clear that a Strangler Fig design pattern was the ideal approach. This process involved gradually migrating capabilities and business logic from the legacy monolith to a more flexible, microservices based architecture.

The first capabilities to transition were those that provided visibility of Availability, Sourcing and Fulfilment, while other capabilities with less demanding compute and resourcing requirements— like Personalisation, Payment, Promotions, and Order Management—remain within the monolith.

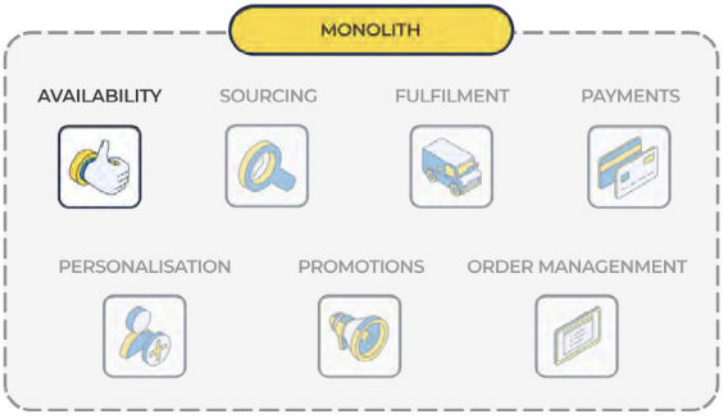
THE STRANGLING MIGRATION APPROACH

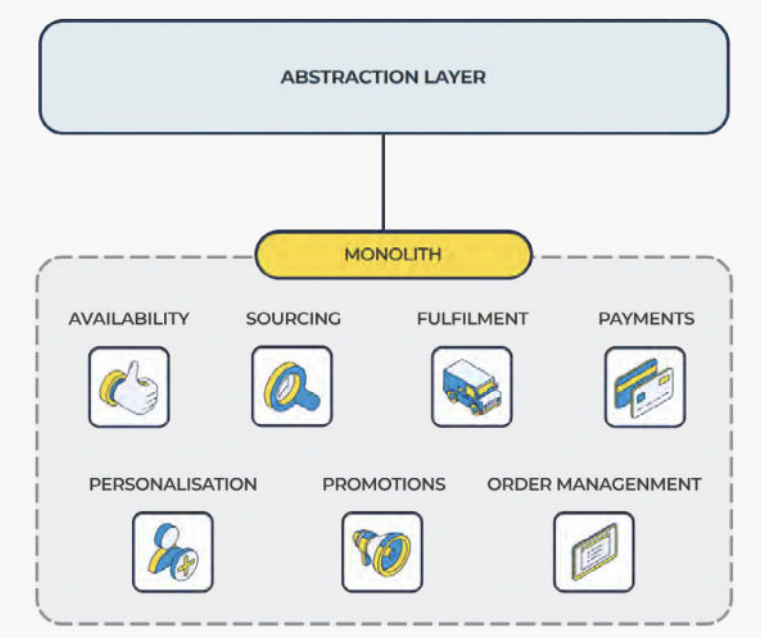
The transition involves strangling out each capability out from the monolith, one-by-one, using a multi-stage assessment process.

To maintain continuity of key business capabilities throughout the transition, we follow four steps:

1. Identify the service

For example, Availability, which highlights store-level availability of an item through the 'availability-to-promise' metric. This service was formerly running in SAP Hybris and was moved to AWS running on Kubernetes clusters.



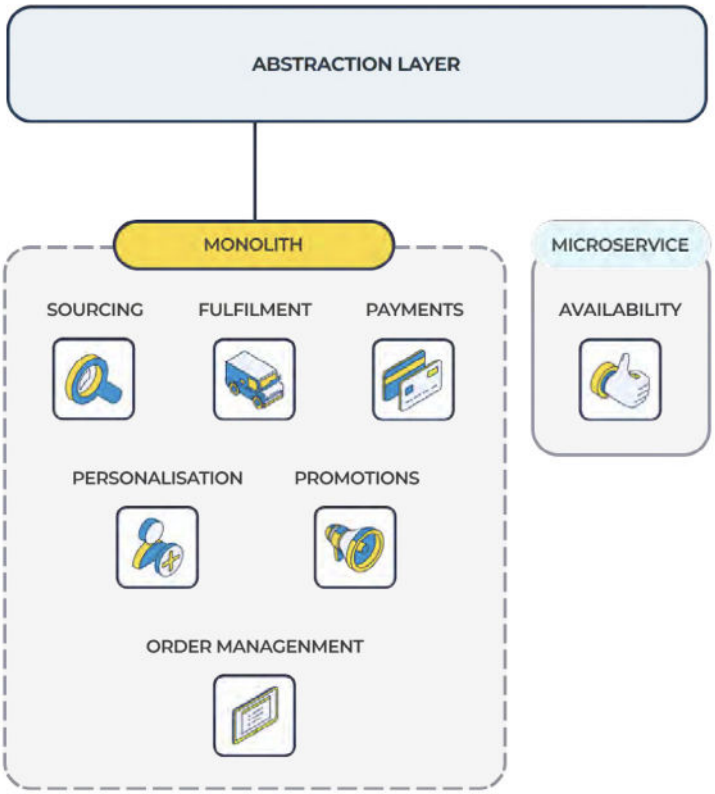


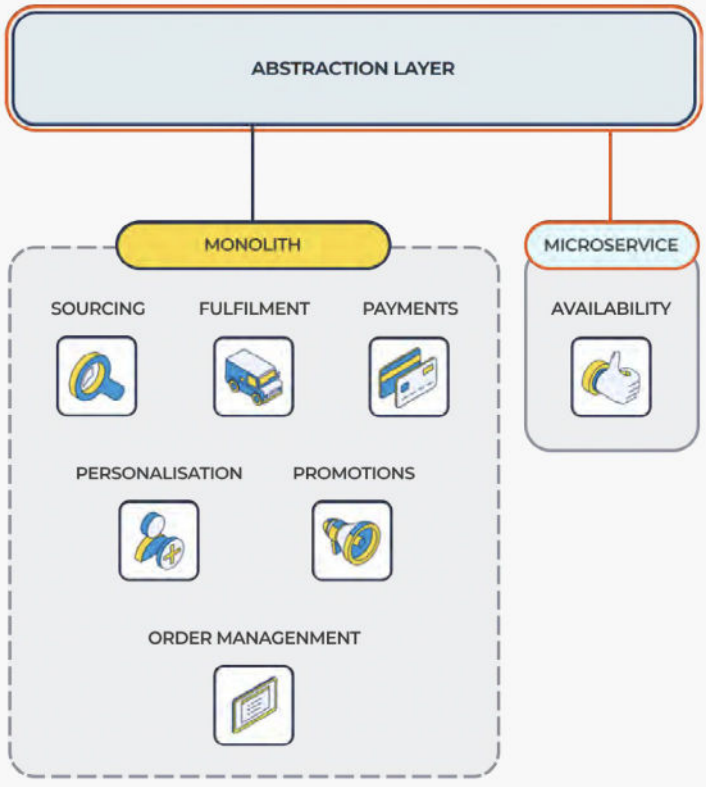
2. Create an abstraction layer

For the capability in SAP Hybris. This allows all the remaining capabilities in Hybris to continue to behave normally, unaffected by the capability being removed from the monolith and reimplemented as a microservice (or series of microservices).

3. De-couple

With the abstraction layer in place, we are able to decouple the capability with a microservice alternative. There is no risk to any other capabilities.





4. Leverage

Now, we are able to make sure any new services point to the newly de-coupled solution, separate to the monolith.

THE BENEFITS OF MOVING TO MICROSERVICES

By moving to cloud services with a microservice-based architecture, we facilitate the necessary scalability and flexibility that our partner requires, both now and in future.

Microservices and event-driven architectures, by their nature, provide a wide range of advantages in comparison to a monolith. These include:

SCALABILITY

A nimble microservices architecture allows the retailer to rapidly scale in response to the traffic and infrastructure demands associated with large seasonal retail events in a way that simply isn't possible in a monolithic architecture.

FASTER DELIVERY

Unlike monoliths, microservices readily lend themselves to less onerous and more independent deployment processes. As a result, the team can innovate at pace with access to more modular, independent components and a vastly reduced deployment effort.

GREATER EXTENSIBILITY

From a system perspective, event-driven architectures make it incredibly easy and efficient to slot in various third-party systems—as well as building out or 'annexing' additional functionalities and capabilities—as business requirements evolve over time. Learn more about flexibility and extensibility in event-driven architectures.

IMPROVED TALENT ACQUISITION

A revised approach to systems and development practice means we're able to provide better tools and processes that appeal to a premium tier of technical team members. As a result, their people and culture departments are now in a position to attract stronger talent to the digital team.

TECHNOLOGIES AND TOOLS

Throughout the project we have implemented a variety of best-in-breed third-party technical solutions:

Want to know more?

Are you interested in this project?

Or do you have one just like it?

Get in touch. We'd love to tell you more about it.