

When building modern e-commerce platforms or enterprise digital products, one of the biggest architectural decisions is choosing between a bespoke composable architecture and adopting standardised frameworks. This choice carries significant implications for scalability, delivery velocity, accountability, and long-term support. Leading companies like **Netguru**, **Lab Digital**, and **DEPT** have all tackled this decision in different ways, leveraging **MACH principles** and API-first strategies to guide their approaches.



In this blog, I'm going to dig into how you can decide between a bespoke composable architecture tailored to your specific needs versus adopting a proven standard pattern or framework. We will explore key factors like architectural ownership after launch, delivery posture, integration discipline, and the importance of phased migrations to ensure scaling consistency and minimal downtime. As someone who's delivered headless rebuilds, OMS integrations, and multi-market rollouts for over a decade, I'll cut through the buzzword soup to help you land pragmatic, accountable decisions.

Understanding the Landscape: Bespoke vs. Standardised Frameworks

The debate is often framed like this:

- **Bespoke composable:** Architecting each module, API, and integration to your exact needs, building a unique mesh of components that deliver tailored experiences.
- **Standardised frameworks:** Using ready-made platforms or frameworks that provide standard patterns, integrations, and workflows out-of-the-box, often with the benefit of an ecosystem and vendor support.

Let's clarify some common misconceptions first:

- *"Composable" ≠ "Headless"*: Headless means decoupling front-end and back-end, but composability emphasizes modular architecture with interchangeable components adhering to MACH principles (Microservices, API-first, Cloud-native, Headless).
- *Standardised frameworks aren't inflexible*: Many frameworks allow extensions and API integrations, but trading off customisation for faster delivery and scaling consistency.
- *Tooling isn't a silver bullet*: Choosing the latest MACH-compliant tooling doesn't guarantee success without disciplined integration and clear ownership.

Key Themes to Weigh When Choosing Your Path

1. Architectural Ownership After Launch

One of the biggest mistakes I see post-launch is a lack of clarity on who owns the architecture. You can build a brilliant bespoke architecture with hundreds of APIs stitched together, but if no one is accountable to maintain, document, [continuous delivery for ecommerce](#) and evolve it, technical debt quickly spirals out of control.

Ask early: Who owns the architectural roadmap after delivery? Is this a handover to an internal platform team, a vendor-managed service, or a blended model? Companies like **Netguru** emphasize embedding architectural ownership with internal product teams, ensuring knowledge doesn't leave with the agency.

With standardised frameworks, architectural ownership is more straightforward because vendor SLAs and upgrade paths are baked in. However, this can blur lines around customization requests and platform upgrades—so clear escalation paths and SLA definitions are vital.

2. Delivery Posture and Accountability

Delivery posture refers to how teams approach delivery velocity, quality gates, regression safety, and stakeholder engagement.

- **Bespoke composable:** Delivery is often complex and requires tight integration discipline and cross-team coordination. It's easy to fall into "we can do anything" traps — which I never accept as an answer. Over-promising flexibility without accountability creates rework.
- **Standardised framework:** Delivery benefits from repeatable release cycles and vendor support, but teams must resist customization for customization's sake, which could cause costly forks.

Industry insight: **Lab Digital** advises integrating delivery teams early with architecture owners to ensure accountability remains high during phased rollouts, preventing "integration dump" scenarios post-launch.

3. Integration Discipline Beats Feature Checklists

Whether bespoke or standard, a common pitfall is chasing feature checklists without integration discipline. APIs define the contract between components—if these are poorly versioned or lack governance, your delivery team will spend more time firefighting than innovating.

API-first approaches underpin both bespoke and composable standard patterns. Rigorous documentation, testing, and contract monitoring are non-negotiable.

DEPT stresses that integration discipline must become part of the delivery culture, with CI/CD pipelines enforcing contract validation and backward compatibility. It's not enough to have MACH-compliant tooling if teams cut corners on API governance.

4. Phased Migrations Limit Downtime and Risk

Large-scale migrations—particularly from monoliths to composable architectures—are inherently risky. Incremental, phased migrations enable:

1. Real user feedback on small slices of capability
2. Risk reduction by isolating new features from legacy systems
3. Scalability testing in production with real traffic
4. Continuous learning to refine integration points and delivery processes

Case Study Note: Successful multi-market rollouts I worked on always involved a migration roadmap that carefully decoupled front-end and back-end transitions to align with business priorities and minimize customer impact.

Decision Table: Bespoke Composable vs Standardised Framework

Factor	Bespoke	Composable	Standardised Framework	Architectural Ownership	Requires
Ownership	dedicated internal/product team	ownership; knowledge retention risk if not well documented	Vendor provides baseline ownership, but customization ownership may vary	Delivery Posture	High complexity; needs disciplined coordination and accountability
Repeatability	Repeatable delivery cycles; some flexibility	limits for faster velocity	Integration Discipline	Critical to enforce API governance and testing rigor	Governance provided by vendor; need to manage custom API extensions carefully
Customization	Highly customizable, tailored to unique business needs	Limited customization, focused on standard use cases	Scaling Consistency	Can be customized to optimize scaling; risk of inconsistencies if not disciplined	Built-in consistency and updates from vendors
Migration Approach	Phased approach critical to minimize downtime and manage complexity	Can leverage vendor migration tools; less control over timing	Tooling Support	Wide choice; dependent on team familiarity and ecosystem	Vendor tools tightly integrated; faster ramp-up

Guidelines to Make Your Final Call

So, how do you decide? Here are my key guidelines based on years of delivery experience and vendor engagement consulting:

- 1. Identify your core business differentiators:** If your product requires unique workflows or integrations critical to competitive advantage, bespoke composable approaches might be worth the investment.
- 2. Assess long-term ownership capabilities:** Do you have the team capacity and architectural discipline to maintain a bespoke mesh, or would vendor ownership with standardisation provide safer scalability?
- 3. Evaluate integration discipline maturity:** If your delivery culture embraces API-first, contract testing, and continuous monitoring, bespoke compositions can thrive; if not, start with standard frameworks and build maturity.
- 4. Plan for a phased migration:** Whichever path you choose, avoid big-bang launches that increase risk. Prioritize small incremental releases for feedback and troubleshooting.
- 5. Clarify accountability upfront:** Don't accept "we can do anything" without a delivery model and escalation matrix. Know who owns architecture and delivery post-launch — this accountability gap kills projects.

Conclusion

Choosing between bespoke composable and standardised frameworks isn't binary. It's about aligning architecture, delivery, and business strategy while embracing MACH principles and rigorous API-first integration discipline. Companies like **Netguru**, **Lab Digital**, and **DEPT** each demonstrate that accountability, phased rollouts, and clear ownership are far more crucial than chasing the latest composability buzzwords.

Successful platforms are built on sustainable practices that scale consistently with clear governance post-launch. Whether bespoke or standard, demand clarity, discipline, and a delivery posture that tolerates no ambiguity around "who owns what" — because the build is only half the story.



Got questions or want to discuss your architecture strategy? Feel free to reach out – I always ask who owns the architecture after go-live, and I won't settle until that's crystal clear.