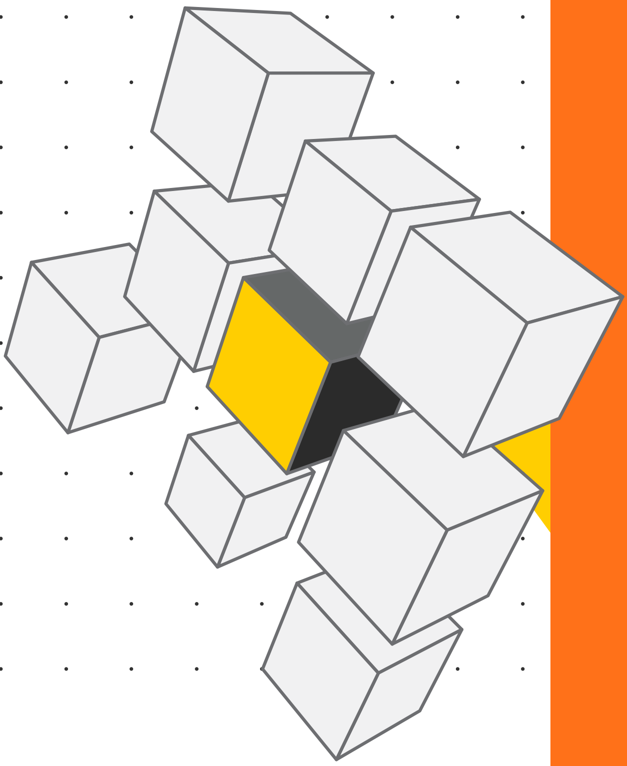
9 Composable Commerce Pitfalls

And How To Avoid Them

KIBO builder.io PEAKACTIVITY

Table of Contents

Composable commerce is a commerce architecture where instead of having one large monolithic backend that packages everything from your catalog, inventory, search, cart, checkout, content, and analytics into one giant backend, you have a series of smaller, loosely coupled backend capabilities for each. Shifting to composable commerce architecture accelerates fast-growth companies and defends incumbents from “death by a thousand cuts” as many new entrants slowly work to pick apart their business with optimized omnichannel commerce experiences by:



- **Improving time to market for new features, campaigns, and products**
- **Lowering the total cost of ownership for commerce technology and infrastructure**
- **Speeding up performance**
- **Enabling teams to create differentiated digital experiences**
- **Making launching and iterating omnichannel experiences much easier**
- **Improving scalability with fewer resources**

Although shifting from a monolithic approach to a composable one is necessary for many teams, it comes with risk. Teams in composable commerce architecture must stitch together multiple systems and uproot workflows that have “worked” for years. And, because you can customize everything, it leaves room for teams to make the wrong decisions accidentally.

Between Builder.io, Kibo Commerce, and PeakActivity, we have helped more than fifty commerce businesses shift to composable commerce architecture.

In this ebook, we outline the most common pitfalls we’ve seen teams fall into when going to a composable commerce architecture and what you can do to avoid them.

1

Not Defining a Clear Re-platforming Business Use Case

Composability is a highly flexible way to build commerce technology, but a lack of clear direction and business use case could result in a solution that takes longer to implement and is more costly to manage unique business requirements in the long run.

How to avoid the pitfall:

▷ Make sure re-platforming aligns with strategic company goals

Clearly articulate how re-platforming aligns with a company's goals by highlighting how a composable solution can improve agility, innovation, operational efficiency, and the customer experience.

▷ Collaborate on written objectives with business and technology teams

Commerce use cases must be well-defined by business and technology teams to ensure that business teams can work autonomously. They should be documented in simple language with functionality and the goal of the function.

▷ Scope to a specific use case

Drafting a business use case helps estimate the human and financial resources needed to implement and maintain the composable commerce architecture successfully.

▷ Examine if it will minimize technical bloat

Emphasize the modular architecture of a composable commerce solution and how this approach allows you to select and integrate only the specific components you need, avoiding unnecessary features that contribute to technical bloat.

▷ Create a cost analysis

Present a cost analysis comparing the expenses of maintaining a legacy system versus adopting a composable solution. Highlight potential cost savings regarding development hours, maintenance efforts, and system performance.

▷ Review industry leaders that are making the move to composable

Composable commerce is a buzzword for a reason — mention industry trends and explain how many leading companies are moving towards composable solutions to achieve greater agility.

2

Failure to Get CEO/Executive Buy-in

Adopting a composable commerce strategy will impact several business functions — it will affect their existing processes, workflows, and the technologies they work in. Without getting CEO and executive team's buy-in, you could be met with resistance from employees who are accustomed to working a certain way.

How to avoid the pitfall:

▷ **Make sure re-platforming aligns with strategic company goals**

Clearly articulate how re-platforming aligns with a company's goals by highlighting how a composable solution can improve agility, innovation, operational efficiency, and the customer experience.

▷ **Examine if it will minimize technical bloat**

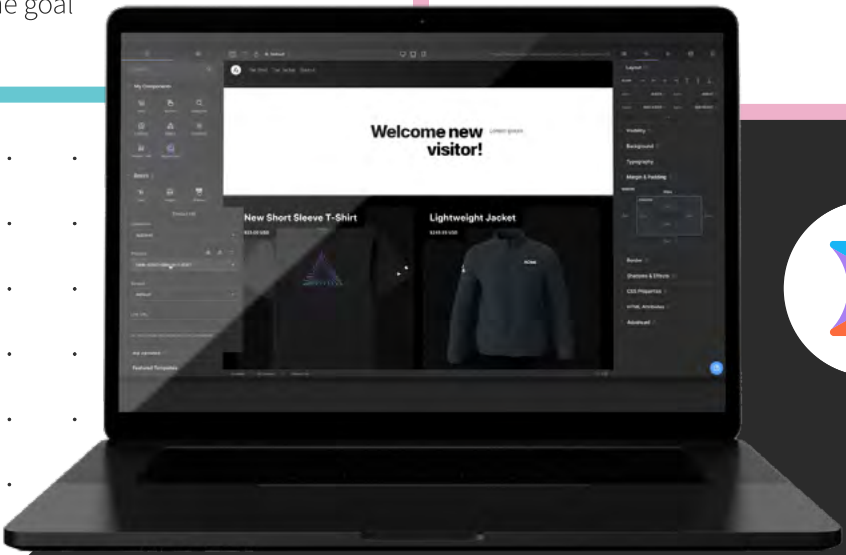
Emphasize the modular architecture of a composable commerce solution and how this approach allows you to select and integrate only the specific components you need, avoiding unnecessary features that contribute to technical bloat.

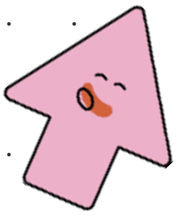
▷ **Collaborate on written objectives with business and technology teams**

Commerce use cases must be well-defined by business and technology teams to ensure that business teams can work autonomously. They should be documented in simple language with functionality and the goal of the function.

▷ **Create a cost analysis**

Present a cost analysis comparing the expenses of maintaining a legacy system versus adopting a composable solution. Highlight potential cost savings regarding development hours, maintenance efforts, and system performance.





3

Doing It All In One Bang

Replatforming an entire commerce platform all at once rather than in phases can lead to scope creep, compromised quality control, resource constraints, missed adaptability opportunities, extended time-to-market, and a challenging user and team adoption.

How to avoid the pitfall:

▷ Invest in training and skill development

Encourage team members to acquire expertise in composable commerce through training programs, workshops, and certifications. This investment will equip them with the necessary knowledge and skills to contribute effectively to the migration process.

▷ Hire composable commerce experts

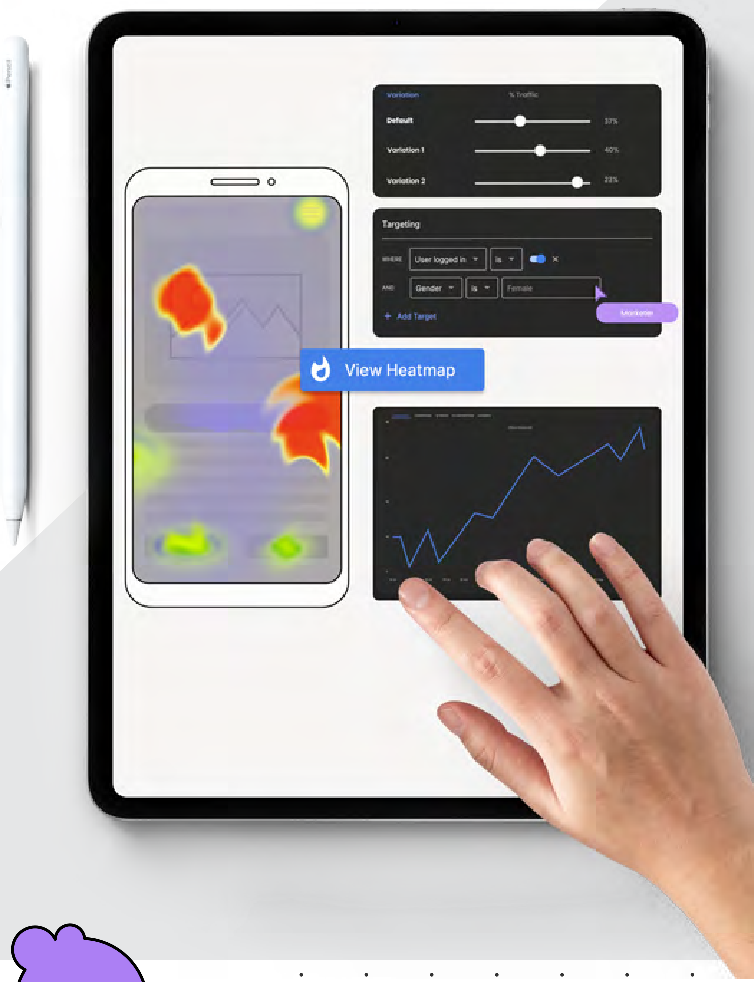
If in-house expertise is limited, consider hiring specialists with experience in composable architectures. Alternatively, work with external consultants who can guide the team and provide specialized support during the migration.

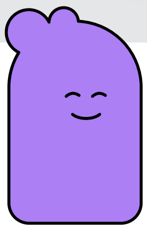
▷ Build cross-functional teams

Form cross-functional teams comprising developers, architects, security experts, data specialists, project & product managers, stakeholders/decision makers, and outside consultants. This collaborative approach ensures a holistic understanding of the migration requirements and facilitates effective decision-making throughout the process.

▷ Engage in pilot projects

Before committing to a full-scale migration, undertake pilot projects with a limited scope. This allows the team to experiment, learn from the experience, and fine-tune their approach under the guidance of experts.





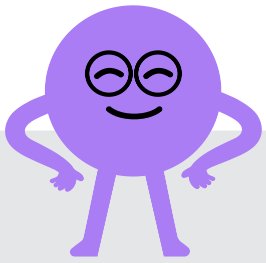
KIBO builder.io PEAKACTIVITY

4

Not Having the Right Skills for the Shift and Ongoing Operations

Underestimating the importance of having experts in composable commerce during the migration can lead to negative outcomes. Without their expertise, teams may struggle to design, implement, and manage a composable architecture effectively. Also, teams may overlook critical aspects of the migration, leading to security vulnerabilities, inefficient service compositions, or inadequate scalability.

How to avoid the pitfall:



▷ Invest in training and skill development

Encourage team members to acquire expertise in composable commerce through training programs, workshops, and certifications. This investment will equip them with the necessary knowledge and skills to contribute effectively to the migration process.

▷ Hire composable commerce experts

If in-house expertise is limited, consider hiring specialists with experience in composable architectures. Alternatively, work with external consultants who can guide the team and provide specialized support during the migration.

▷ Build cross-functional teams

Form cross-functional teams comprising developers, architects, security experts, data specialists, project & product managers, stakeholders/decision makers, and outside consultants. This collaborative approach ensures a holistic understanding of the migration requirements and facilitates effective decision-making throughout the process.

▷ Engage in pilot projects

Before committing to a full-scale migration, undertake pilot projects with a limited scope. This allows the team to experiment, learn from the experience, and fine-tune their approach under the guidance of experts.

5

Not Architecting Data Workflows Before Migration

If data workflows are not properly architected prior to migrating to a composable commerce architecture, it can lead to significant challenges and complications in the new system. Without careful planning, data may become fragmented, scattered across various services, and difficult to manage cohesively. This can result in data inconsistencies, redundancy, and loss of integrity, leading to erroneous information and unreliable processes. Additionally, the lack of a well-defined data architecture can hinder communication and synchronization between services, causing bottlenecks and performance issues.

How to avoid the pitfall:

▷ **Define clear data models and standards for the new composable architecture**

This includes naming conventions, data formats, and data relationships.

▷ **Assign data ownership responsibilities to specific individuals or teams**

Implement data governance practices to ensure data is managed, accessed, and updated appropriately. This prevents data fragmentation and enhances data integrity.

▷ **Design APIs to facilitate data communication between services**

Consider using event-driven architectures or message queues to enable real-time data updates and reduce the chances of data inconsistency.

6

Not Architecting Commerce or Order Management Workflows Before Migration

Adopting a composable solution for omnichannel without considering the complexity of integrations and workflows between the ecommerce functions and distributed order management solution is like embarking on a long road trip without a map or clear directions — you could encounter delays, dead ends, and unnecessary backtracking.

As a result, you could experience extended implementation time, scalability concerns, disorganized integrations, and a bad user experience — for both internal teams and customers.

How to overcome this pitfall:

▷ Map commerce and order management workflows

Pre-planning the workflows allows you to align the workflows with their existing processes, optimize them for efficiency, and adapt them to future changes without disrupting the overall system.

▷ Architect with scale and performance in mind

By architecting the workflows ahead of time, you can design the system to handle increased traffic and order volumes and ensure the system is optimized for performance and will grow with the business.

▷ Align to metrics for each department

Well-defined and logical workflows lead to smoother coordination among departments such as sales, inventory management, customer support, and customer experience. This translates to quicker issue resolution, accurate order processing, and ultimately a more efficient internal user experience that drives a better customer experience.

▷ Plan data governance and standardizations

Architecting workflows helps identify potential data synchronization challenges between ecommerce and order management systems—allowing you to maintain data integrity and avoid any discrepancies in inventory or order information.

7

Not Architecting Content and Merchandising Workflows

Getting products, promotions, and experiments to market faster and with fewer resources is a major reason why teams shift to composable commerce, but if your marketers, merchandisers, and product team members can't use your systems without engineering support or switching across a dozen tools to recreate workflows that worked on a monolithic system then much of the benefit of composable commerce is rendered moot.

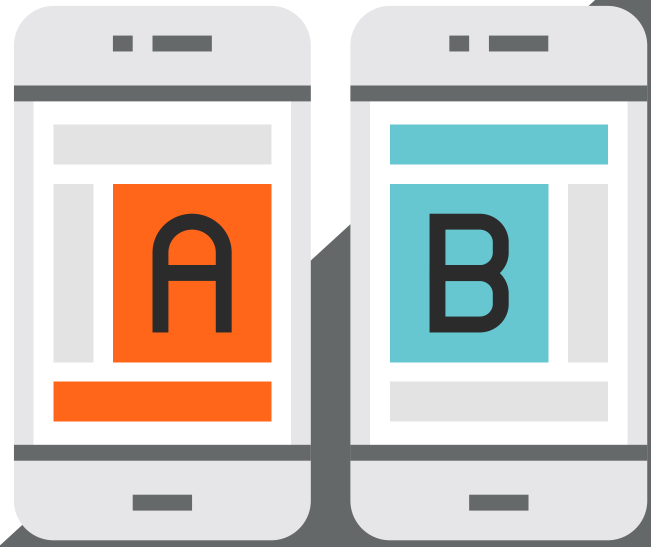
How to avoid the pitfall:

▷ Identify and map the dependencies and gaps in the internal content workflows

Start with a comprehensive journey mapping of your existing internal content and merchandising workflows. This means scrutinizing each step, from content creation to its delivery, and identifying all dependencies and potential bottlenecks. For example, if the content approval process involves multiple stakeholders across teams and results in delays, this gap should be identified and addressed in the new composable commerce structure or matched with automated workflow alerts that make sure content doesn't get "stuck" in the pipeline. This exercise will reveal inefficiencies and help you architect the ideal workflow in your new commerce architecture.

▷ Map ideal A/B testing and experimentation workflows

Identify the stages of hypothesis generation, designing variations, running tests, analyzing results, and implementing changes. Recognizing how these stages integrate with your content workflows, analytics, attribution, and reporting workflows with a composable commerce architecture to ensure a seamless transition. For example, some composable commerce implementations include analytics in context of the content for how often each type of persona (i.e. customer/non-customers, city, gender, etc.) has visited each page so teams can prioritize and quickly test personalized variations in context.



7

Not Architecting Content and Merchandising Workflows (cont.)

▷ Invest time in team training and enablement

Plan to allocate time for hands-on training sessions and workshops to make sure teams feel confident and understand the full potential of what can be done or created. For example, your marketing team may need to understand how to use new tools for personalized campaign management, and your product team may need to learn how new workflow integrations get implemented.

▷ Plan how much design freedom each content role should have

One pro of composable commerce is that it gives content creators more freedom than ever, but one con is with more freedom comes potential for disjoined design and user experiences. To help your team stay on-brand and keep designs consistent create a plan to roll out more design freedom. For example, designers should have full access to custom styles, whereas, in the first month, marketers might only have access to custom components with a few inputs, and gradually get more access to expanded on-brand color swatches, etc. Over time, as you gain a better understanding of the workflow, you can refine these permissions for optimal efficiency.

▷ Leverage APIs and webhooks to automate workflows

Shifting to composable commerce often means that every service has APIs and there's a webhook for almost anything that exists in any of your systems. Webhooks trigger instances such as new products added to the catalog, new images associated with a product, or new promotions added to your commerce system. By leveraging webhooks to trigger actions, you can automate workflows across multiple systems without human intervention.

8

Using a Commerce Platform That Requires You to Jump Into a Full Composable Commerce Backend From the Get-go

Jumping into a fully composable commerce backend from the get-go could potentially add complexity and impose a steep learning curve on your organization. If the organization lacks the necessary expertise or resources, the learning curve can be overwhelming, leading to delays in implementation and increased training efforts. Moreso, investing in a full composable commerce backend could result in higher expenses compared to incremental adoption of the composable approach.

How to avoid the pitfall:

▷ Choose a commerce platform with a modular architecture

Similar to LEGO bricks, the components of a composable system function autonomously, allowing you to assemble them in various configurations to form a tailored solution. By taking a phased approach, you can build and expand your system as needed.

▷ Look for packaged business capabilities

PBCs can be added to an existing system or cherry-picked to create a new one. They make it (much) easier to incorporate new or custom functionality versus traditional monolithic systems.

▷ Select a solution focused on business requirements

Find a composable solution that centers around fulfilling business requirements—enabling swift adjustment of features and capabilities in response to customer input, market shifts, or internal projects.

▷ Make sure your commerce platform seamlessly integrates with other platforms

Composable systems are engineered to seamlessly interact with various systems and data sources, both within and beyond the organizational boundaries. This makes them much more flexible than traditional systems, which are often siloed and difficult or cumbersome to integrate.

KIBO builder.io PEAKACTIVITY

9

Using a Headless CMS Without a Visual Workflow

A content management system (CMS) is core to delivering content to the frontend user experience, and almost every team that shifts to a composable commerce architecture uses a headless CMS. The problem is that most headless CMSes only give business team members a form-based user interface that ultimately ends up with business team members not being able to visually create and iterate digital experiences, requiring them to constantly rely on engineering teams for small fixes such as adding a new field for one-off campaigns, adjusting image layouts, and altering text and fonts.

How to avoid the pitfall:

▷ **Choose a headless CMS that lets business users drag and drop to create experiences**

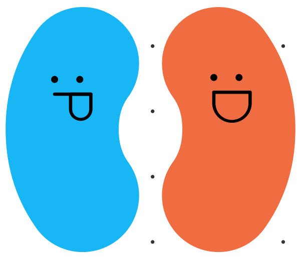
Plan to allocate time for hands-on training sessions and workshops to make sure teams feel confident and understand the full potential of what can be done or created. For example, your marketing team may need to understand how to use new tools for personalized campaign management, and your product team may need to learn how new workflow integrations get implemented.

▷ **Seek a headless CMS that lets marketers target personalized experiences and ship experiments on their own**

A headless CMS that allows product teams to target personalized experiences and ship A/B tests empowers your team to constantly optimize content for conversions and upsells. For instance, if your team is launching a campaign for a specific demographic group, they should be able to customize the experience for that audience and quickly implement A/B tests to determine the most effective content variations without needing to rely on the engineering team.

▷ **Select a headless CMS that enables developers to easily integrate third-party components into a visual editor**

The ability for developers to easily integrate components from third-party systems is a crucial feature that a headless CMS should provide. This allows business users to drag and drop these components onto their interfaces. For example, developers should be able to create a carousel component of recommendations that marketers can drag onto any page and customize for each persona.



Ready to Learn More?

Learn how Builder.io, Kibo Commerce, and Peak Activity can accelerate your commerce transformation and eliminate the risks along the way.

About Builder

About Kibo

About PeakActivity