THE FUTURE OF ECOMMERCE - EVERYTHING YOU NEED TO KNOW

HOW ORGANIZATIONS CAN FUTURE-PROOF THEMSELVES USING THE LATEST TECHNOLOGY

AN
ENGINESS
BUSINESS GUIDE

Table of Contents

Introduction

Technology has transformed how we live our lives in an almost infinite number of ways. Perhaps the starkest example of that is eCommerce.

The explosion of eCommerce has changed how businesses operate in a very significant way. From compressing margins and cutting out middlemen like Amazon and Alibaba and creating impossibly niche websites for impossibly niche products, to building platforms to sell anything online (Etsy, eBay and so on) — it has changed the way we shop.

Even at a business level, eCommerce is increasingly the transaction of choice. But that's everything that's been done.

So what's next for eCommerce?

How can businesses both meet the incredibly high bar consumers have set in terms of total customer experience now **and** future-proof themselves against future technology and improvements to eCommerce?

That's what we are going to answer in this ebook.

In this ebook, we are going to cover:

- Traditional vs headless eCommerce architecture
- The advantages of the headless architecture
- Why (we think) the future of eCommerce is headless
- How agile development applies to eCommerce projects
- The three stages of agile eCommerce transformation, and how to recognize which stage you are in
- How organizations can future-proof themselves using the latest technology.

By the end of this ebook, you will know what's next for eCommerce, where your organization is now, and what steps you need to take to advance your eCommerce experience to the next level.

About Us

Enginess is a Toronto-based digital consultancy. We empower businesses to get the most out of technology.

We shape strategies for business processes and deliver solutions that enhance customers' experience, improve efficiencies, generate new market opportunities, and redefine value creation for many different kinds of organizations.

We offer services that include:

- Technology Procurement
- Digital Business Strategy
- Revenue Growth Strategies
- Website, eCommerce and Custom Software Development
- Experience Design

Over the past 18 years, we've built a solid reputation based on our ability to consistently deliver high-quality customer service and handle complex digital strategy and development assignments.

The Enterprise.
Futured.

1.416.901.6151
info@enginess.io
enginess.io

@enginessio

@enginessio

@enginess

Chapter 1: Traditional vs Headless CMS for eCommerce

Ecommerce is nothing new. According to BigCommerce, it all started back in 1969 with CompuServe. However, if you are like us, you probably think of the first eCommerce starting in 1995 when Jeff Bezos launched Amazon.com.

That said, there are some significant differences between the platform Bezos built over two decades ago and what eCommerce is now.

Today, eCommerce is valued at $2.8 trillion and is expected to grow 71% to almost $5 trillion by 2021. To give you a sense of comparison, if global eCommerce was a country, it would have the 10th largest GDP in the world. So like any massive industry, the underlying infrastructure is massively complex. For CEOs, business founders, and the CMOs who are responsible for the eCommerce functionalities, it can be a bit daunting to dive into.

In this chapter, we are going to explore:

- What traditional eCommerce architecture is
- What headless architecture is
- The differences between traditional and headless
- Some of the benefits of embracing headless eCommerce.

By the end, you should have a clearer sense of what architectural options are available, and a better sense of which one will work for your business.

What are traditional and headless eCommerce architectures?

Traditional architectures are the bulk of eCommerce experiences. What this means is that the providers (think Magento or Shopify) offer a full stack solution. That is traditional architecture couples the backend data storage and management with the front-end experience. Essentially, it says: 'this is the data (from the backend) and this is the container and/or experience that it creates (the front-end).'

In contrast, **headless architectures** are, broadly speaking, architectures that decouple the back-end data from the front-end experience, and relies on REST/RESTful APIs to move the data from A to B.

Essentially, you split the codebase between the backend and the front-end. As Jess Amerson, the IT director at Ulta Beauty, put it:

"[headless architecture] splits the code base cleanly between server side (e.g. REST webs services) code which defines data and business logic, from client side (e.g. JavaScript) code which invokes the logic and renders the data… typically as part of an MVC pattern."

Here is an easy way to think about it.

Imagine your wrist is fixed. It can't bend. That's traditional architecture. Whatever changes in the backend (think your forearm) affects the front-end experience (your hand).

Headless architecture is as if your wrist can bend. You still have the backend data (your forearm)…

… but without changing the backend data, you can change the front-end experience (when you bend your wrist).

The reason is that the data stored in your server is only the content that your website or app is filled with. The buckets that the data (e.g. content) flows into is completely defined on the front-end using code like JavaScript.

The difference between traditional and headless architecture

Clearly, there are significant differences between these two architectures. But it is worth calling a few of them out specifically.

Traditional architecture defines both the front-end and the back-end experience
Traditional architectures solutions define both the front-end and the back-end. Shopify, for instance, offers a turnkey solution with all the basic eCommerce functions. But if you want to customize the front-end experience (e.g. change the experience on a mobile phone by only showing specific content), you might not be able to.

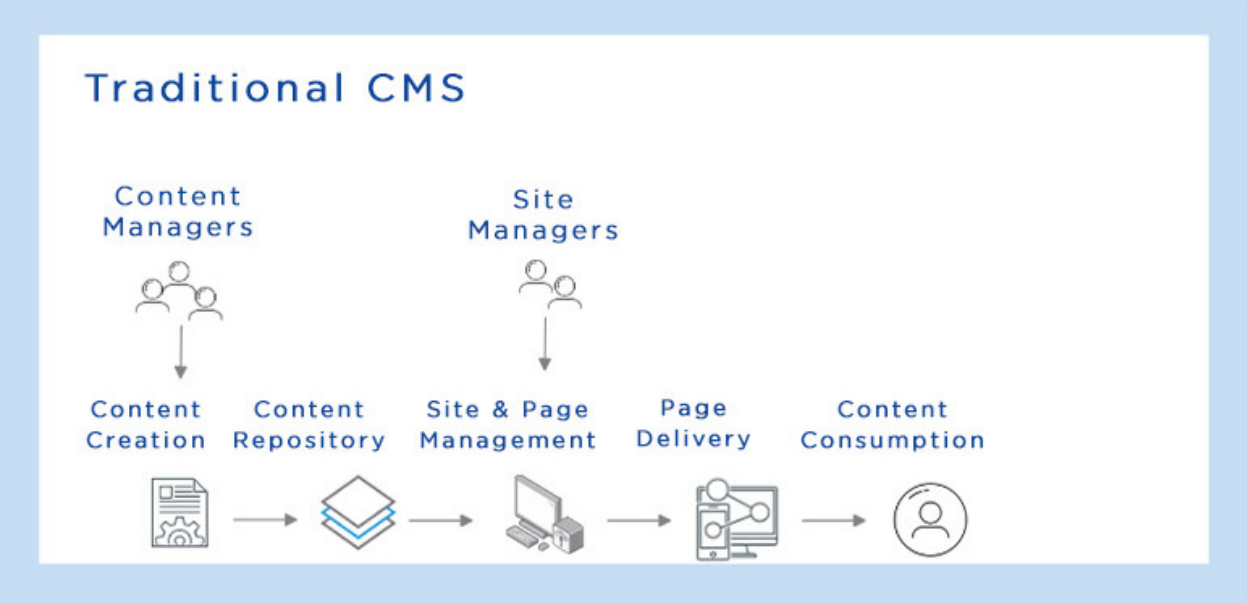


Image Source: *Advantage CSP*

Why?

Because changing the front-end requires a change to the back-end data, OR the writing of new code to mitigate the relationship between data and experience. And with a cloud, multi-tenant servers on a shared code base, Shopify can't change for a single customer. Hundreds or thousands of clients use the same codebase on that server!

In contrast, headless architecture only defines the back-end data structure. It's up to the end client to define how that data is used and displayed, and they can do that however they want. JavaScript is often used to do this, but there are other options.

This means that front-end developers can define whatever experience they want, be it on a screen, using VR / AR technology, or some other new technology we haven't even thought of.

Headless solutions open up the administration experience

One big (and often overlooked!) difference between headless and traditional eCommerce solutions is the experience of the website administrator.

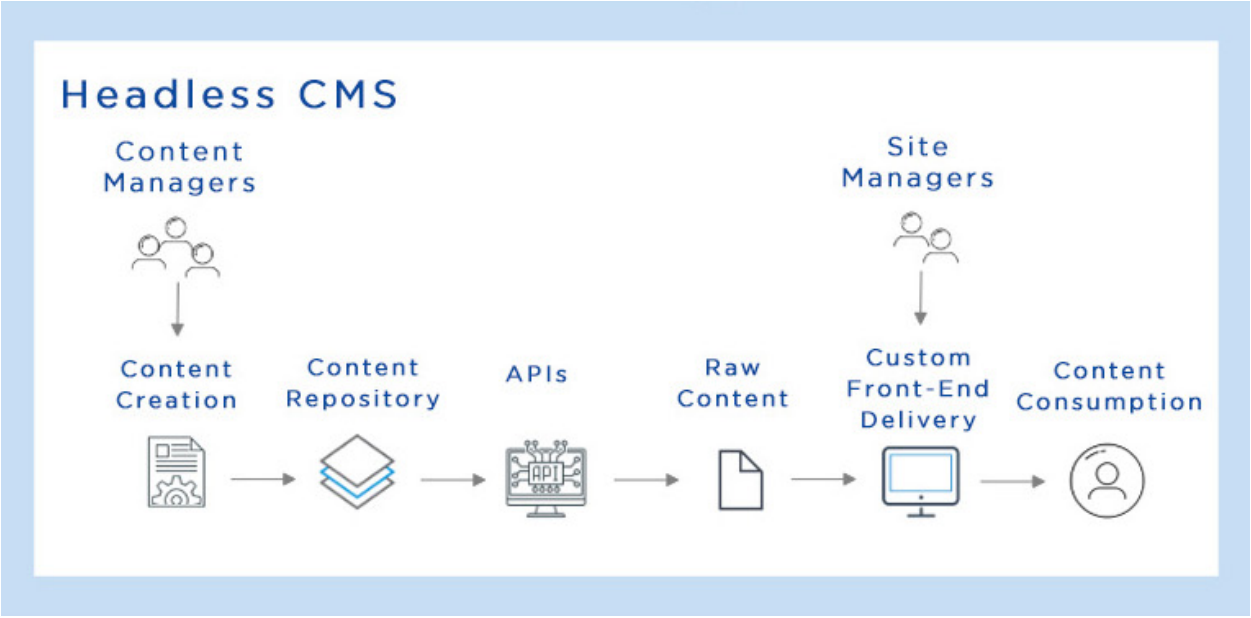


Image Source: *Advantage CSP*

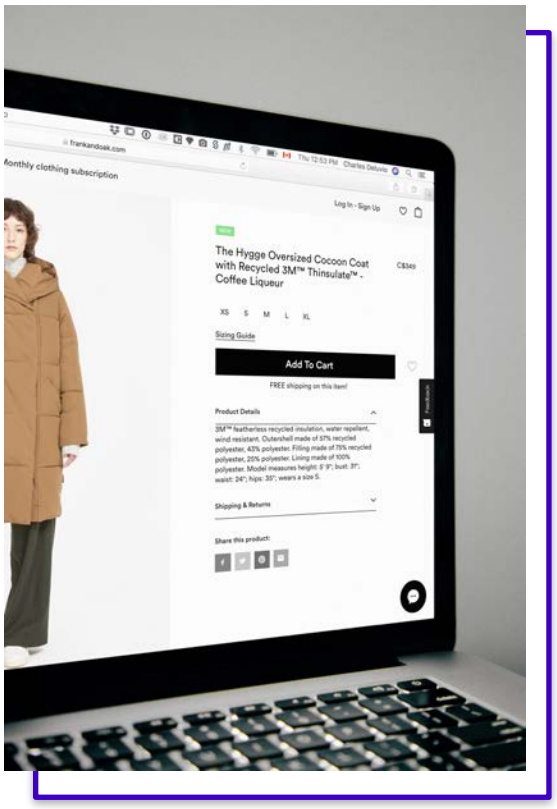
To borrow an example from the content management system (CMS) world, consider CMS. It's extremely flexible, and you can design almost infinite front-end experiences. However, it is still a traditional solution. And as a consequence, no matter what your front-end, the website admin experience is virtually identical for the hundreds of millions of CMS sites across the internet.

In contrast, headless architecture, for eCommerce or a CMS, offers a completely unique experience for both the end user and the admin, which is extremely beneficial if you have multiple user roles, functions, and requirements that might change extremely quickly.

The pros of going headless

So with these two solutions, is there a clear winner?

The short answer is: "It depends."



For small organizations who just want a super basic eCommerce site, then traditional eCommerce will work fine. There are tried and tested providers who can supply all the basics in a turnkey solution. Shopify is a great example of this.

However, the future undoubtedly belongs to headless. And there are a few reasons why.

Headless solutions are future-proof

We don't know what new devices we are going to use in the future. Maybe screens will be completely gone and we will all use voice activation only. Maybe we will want to stream eCommerce content directly into our eyeballs.

Who knows.

By completely decoupling the back-end data from the front-end experience, organizations can ensure that they're ready for customers, no matter where or how they

choose to engage. And by decoupling your web experience, it reduces complexity and improves performance.

Headless solutions are fast

The bulk of web development is defining business logic of what content to display, and how to display it.

But headless solutions resolve that problem, simplifying matters for the back-end and putting that business logic into the front-end language. This is generally much easier to manipulate and much, much faster to test and release, meaning new changes can deploy incredibly quickly.

Headless solutions make content/site management easy

Content management is always a struggle for eCommerce businesses, who regularly have thousands or even tens of thousands of SKUs.

Headless solutions make this job much, much easier. Why? Because that content can be stored effectively in the back-end and then manipulated to suit the environment on the front-end when needed.

Second, headless solutions mean far more things can be templated, both within and across various web properties and at both a content and website level. This makes managing multiple properties and a massive amount of content much easier and much faster.

Headless solutions enable personalization

Marketing is heading towards a completely personalized world. And headless eCommerce makes that possible by serving personalized content to users, wherever

those users happen to engage. Because the data is captured and centralized in the back-end, it can be served to whatever digital channel the customer is browsing in, and can be dynamically and proactively pushed to additional channels.

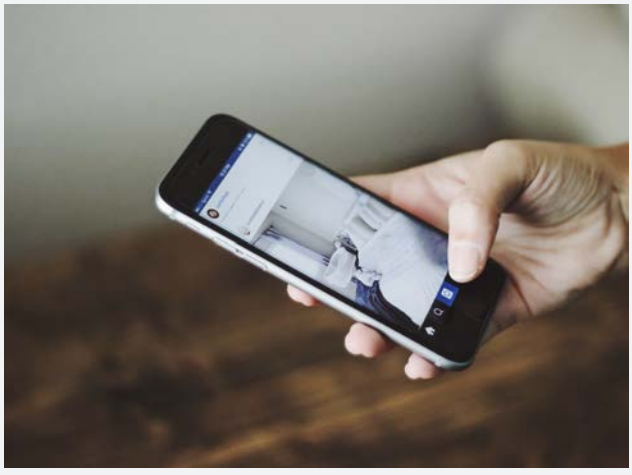
Headless solutions are easier to maintain
From a security perspective, on the front-end and database are publicly accessible — all the goodies in the back-end are locked away.

Second, there's more flexibility in design. And while the quality of the site might end up the same, it's often easier (and faster) if you don't have to shoehorn a design into an existing format or tool.

Third, headless solutions have less overhead. Progress web apps (PWAs) and other front-end technologies tend to be easier to use and deploy, thereby reducing how much time / staff are needed to build and maintain them.

The cons of going headless

There are not any significant technical limitations to headless architecture. However, there is one large, obvious problem: you now have two of everything.



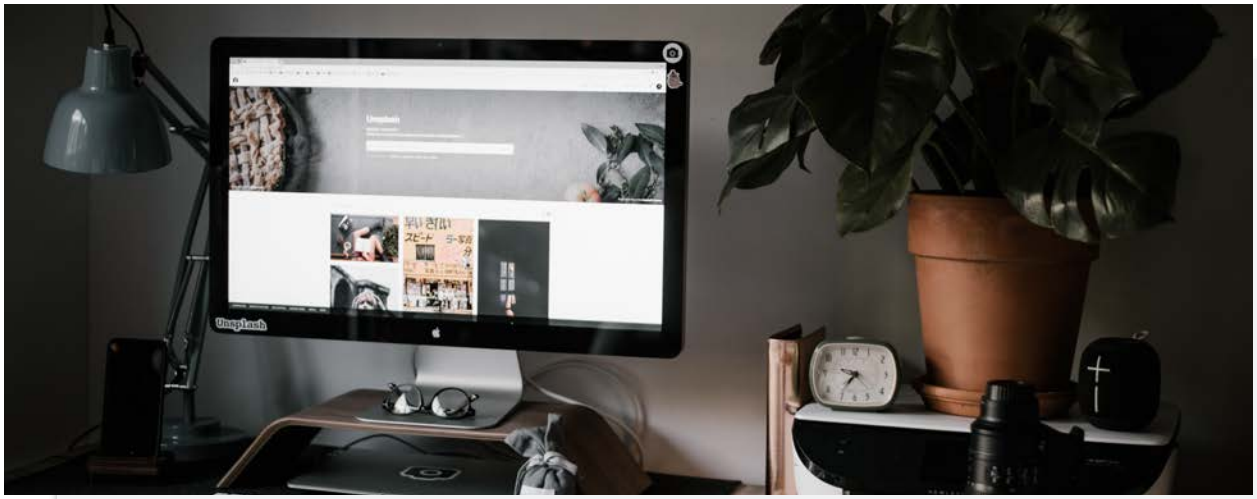
You're effectively maintaining two systems, including two databases and code bases. This means that you potentially need more teams to maintain them, or if you have one team, a larger team since some work will be duplicated.

This problem of workload has a couple of specific consequences worth calling out.

Total cost of ownership
While we maintain that headless solutions are easier to maintain, there are some increased cost during initial implementation, especially if you're migrating from an existing traditional eCommerce site.

Integrations can be hard
While the two systems you're now maintaining should essentially work independently, and connect via APIs, there's always going to be some applications that need both frontend and backend connections. These can present a thorny problem for developers, and might take longer to figure out.

No staging
Teams love staging in a CMS. It allows senior leads and stakeholders to see what a website or application will look like in situ — an incredibly powerful experience when you're reviewing something. No staging means that an organization has to both embrace the fact that there will be mistakes, and requires developers to be more diligent over what is pushed live.



Who should use headless architecture

Headless architecture is for anyone. There's no perfect use case, and as we said, it depends on the needs of the organization.

But there are a few examples where headless architecture makes an awful lot of sense.

- **Scaling companies.** The flexibility and future-proofing that headless eCommerce provides are especially appetizing to scaling companies. It allows them to build something now that they know is going to work in the future, rather than building something now only to need to rip it out and replace it 18 months down the line.

- **Content heavy eCommerce.** If you have a B2B eCommerce company that sells a commodity like screws, where the spec sheet is the most important thing, you might be OK with a traditional solution. However, if you are a content-heavy eCommerce site like an online clothing store, then headless is usually a better way to go. For instance, consider Asos.com. It's a clothing site that for every single item has product specs, a description, sizes, colours, size notes, multiple images, and usually a product video. Keeping that content straight is much easier when you can deploy it dynamically in a modular way (e.g. headless-ly).

- **Marketing-centric organizations.** Organizations that are either marketing led or are looking to leverage marketing as a core differentiator in a highly competitive space are also good candidates for headless because it gives marketing teams the ability to curate a total customer experience with fewer technical inhibitions.

Headless examples

This isn't just theoretical as well. Headless architectures, but eCommerce and CMS, are well on their way towards the mainstream. There are a number of successful deployments worth calling out.

Herschel

Herschel built a beautiful website based on a headless architecture. Like most eCommerce sites, Herschel's had to contend with a huge amount of content that's being constantly updated, while providing an easy, visual experience that the brand's become known for. A headless architecture let them do just that.

Harts of Stur

Harts of Stur offers a slightly different approach to a headless design. They built their front end as a progressive web application while using a headless architecture to seamlessly and effectively provide the content.

Kodak

Kodak had a complex eCommerce architecture combining lots of different software. It meant that their marketing and merchandising team struggled to innovate and test quickly enough to identify what worked and cut out what didn't.

So they adopted a headless design. Combining compliance and transaction software with a CMS content interface that marketing teams were already familiar with, they enormously increased the pace of innovation without putting customers at undue risk. Plus, it meant that they could integrate seamlessly with Amazon, Apple, and PayPal payment solutions without bogging down the development team with a lot of custom coding.

Elastic

Elastic is a SaaS company that lets businesses take control of their data by making it manipulatable, integrated, searchable, and visible.

They were working with a design and development agency for their CMS front-end, but that system wasn't working for them. As a result, they shifted towards a headless CMS system. By going headless, Elastic was able to push documentation out the door faster than ever, without spending big on agency design/development.

Chapter summary

- Traditional eCommerce is when the back-end of the application is tightly coupled with the front-end experience. The back-end data must conform to the parameters set by the solution provider. Organizations like Shopify and Magento both provide robust traditional eCommerce solutions.
- Headless eCommerce architecture is when the front-end is decoupled from the back-end, relying on APIs to call the data it needs when and where it needs it. This means that the front-end experience is completely defined by the front-end scripting like JavaScript.
- Headless solutions allow teams to future-proof their products because they can push data into whatever framework or bucket or format they need.
- Headless solutions enable rapid testing and new solution deployment/iteration, without diving into the back-end.

- Headless solutions streamline content management because content only needs to be created and maintained in one location in the back-end.

- Headless solutions improve the customer experience by delivering a truly seamless, personalized experience to consumers for the very first time.

Chapter 2: An Agile Approach to eCommerce

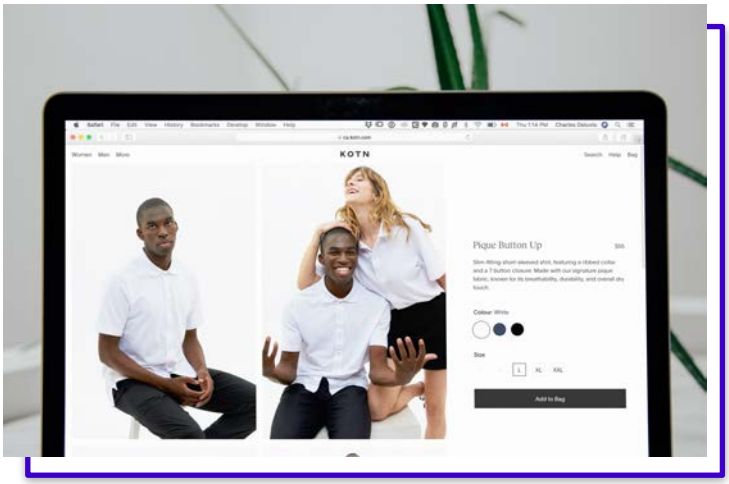
Last chapter we looked at one of the hottest topics in eCommerce right now — headless vs traditional eCommerce structures, and a few of the benefits of working with a headless implementation.

However, this is only one element of modern agile eCommerce that organizations should be looking to adapt.

An agile approach to eCommerce is when an organization approaches their eCommerce with an agile development framework.

Namely, iterative improvements that build a better overall product, with lots of revisions and reworks along the way. This is in contrast to traditional project development, where a single, large deliverable is handed over at the end, at which point testing and changing can be extremely time-consuming.

By approaching eCommerce and eCommerce architecture in a staged, incremental, and iterative way, businesses can be flexible, nimble enough to respond to changing requirements and demands.



In this chapter, we are going to explore the three stages of embracing an agile approach to eCommerce.

In stage one of agile eCommerce, businesses should be connecting their back-end systems using APIs to push and pull data wherever it needs to go.

In stage two, businesses should be closely integrating their eCommerce platform with their content management systems.

And in stage three, businesses should be looking to scale eCommerce in an agile way by embracing a microservices architecture.

By the end of the chapter, you should be able to identify what stage your organization is at, and have a general idea of what milestones to aim for within each stage.

Stage 1: APIs

APIs are incredibly important for modern IT infrastructure. Ever since Salesforce first introduced them in 2000, they've been an integral part of how we use technology today.

But before we continue, what is an API?

What is an API?

API stands for Application Programming Interface. It's essentially the data translator that turns the commands issued to a server by the end user's browser into commands that the server can understand, to pull the data it needs to.

An easy way to think about this is like a reference library. Library-goers come up to the window and request a specific book. The librarian goes and finds the book and brings it back to the window for the patron to read. An API works the same way.

The library-goer (the user) requests some information stored on a server or available through another app. This might be a request triggered by them submitting a form, loading a web page, reloading a live data table, refreshing a weather map, visiting a booking screen on a restaurant's website — it could be anything.

The request goes to an API, which translates it into a command for a server:

"Go get me this specific data."

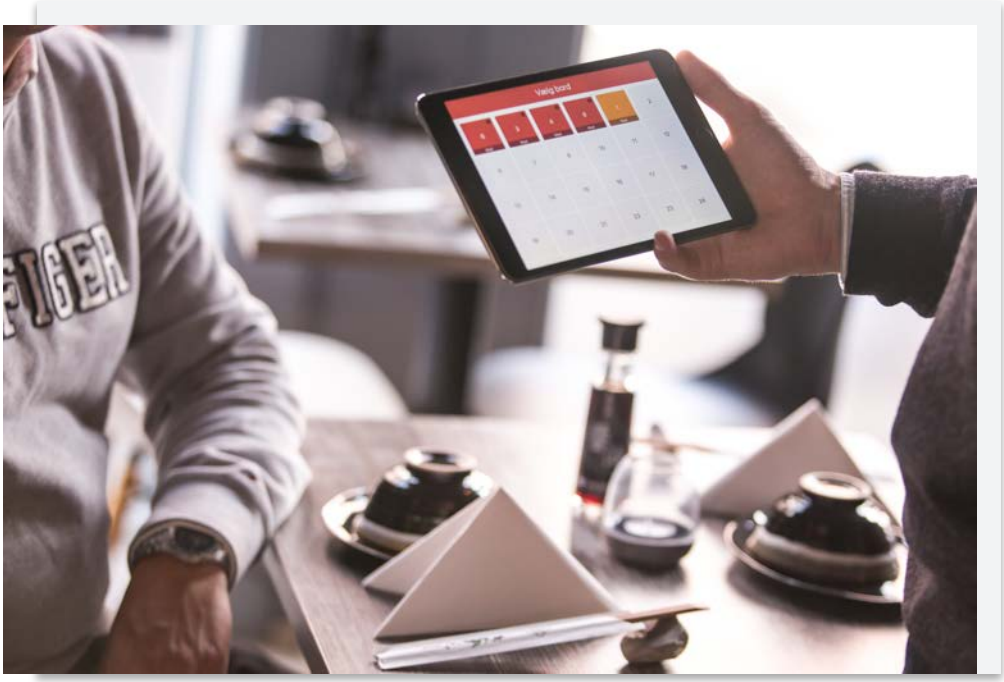
The server comes back, gives the data to the API, which then sends the data on to the end user. And that's (roughly) how APIs work.

APIs and integration

There are three benefits worth calling out here:

1. APIs allow applications to talk to each other in a secure way (e.g. Facebook app pulling data from YouTube)
2. APIs make integrations really, really simple
3. APIs are inherently modular

All of these fuel integration. Integration between different internal systems, integration between internal and external systems, and integrations across an omni-channel customer journey.



By enabling integration, APIs can help businesses take a big step towards providing the eCommerce experience that consumers expect today.

And this goes for any scale of business.

For instance, imagine you are a one-person store. You might use an API to connect your store to Amazon and sell your products on Amazon. You might also use an API to

connect your eCommerce platform to your back-end inventory management system so they automatically update as your stock changes.

For larger businesses, you might use an API integration to connect your:

- POS system
- Inventory
- eCommerce platform
- App
- Partner websites
- Franchises and shared warehousing facilities
- Suppliers and purchase order systems
- ERP system
- Warehouse management system

That's why stage one is connecting your tech stack with APIs. When you can push and pull data easily using APIs, then you can start to truly connect your business units and provide a single experience for your customers.

You should be aiming for Stage One if:

- You have multiple disconnected software silos
- You have multiple records of the same data
- You stop expanding your technology over integration concerns
- Tech upgrades and integrations are customized for every application

Stage 2: eCommerce & CMS connection

Businesses often see the eCommerce and the CMS as serving distinctly different functions and operate completely independently. In reality, though, there's enormous overlap in function, even if different teams regularly run the two software platforms.

eCommerce platforms are great … at one thing

Most businesses see their eCommerce platform as a tool to sell products online, manage their product catalogue / SKUs, and let customers rapidly filter search results and only see what they're looking for.

Which eCommerce platforms are undeniably great at.

But there are a lot of other pieces of work that needs to be completed to provide a total experience for the end user.

And eCommerce platforms are not up to the task. Things like workflows, behavioural nudges, modal windows, retargeting, and of course content management are all best to live in a CMS.

Examples of eCommerce solutions

- Shopify
- SnipCart
- Magento
- BigCommerce

Where a CMS wins

These are what we see clients often running into problems with when they are an eCommerce only tech stack:

- **Workflow management.** Workflows and permissions are much easier to manage in a CMS. As teams grow, it is natural to require more restrictions around promotions and web management, not only to prevent mistakes but also to keep the product, messaging, and brand tone of a business tightly bound.
- **Content management.** Content management is, of course, the strict purview of the CMS. And content drives a business forward. Helpful content for consumers, content to drive organic traffic via SEO, content like landing pages for paid ads, pop-ups on paid pages to drive engagement, content to help existing customers and build loyalty — all of these are best to live in a CMS rather than on an eCommerce platform. But make no mistake — they are part of the eCommerce journey.

Examples of CMS solutions

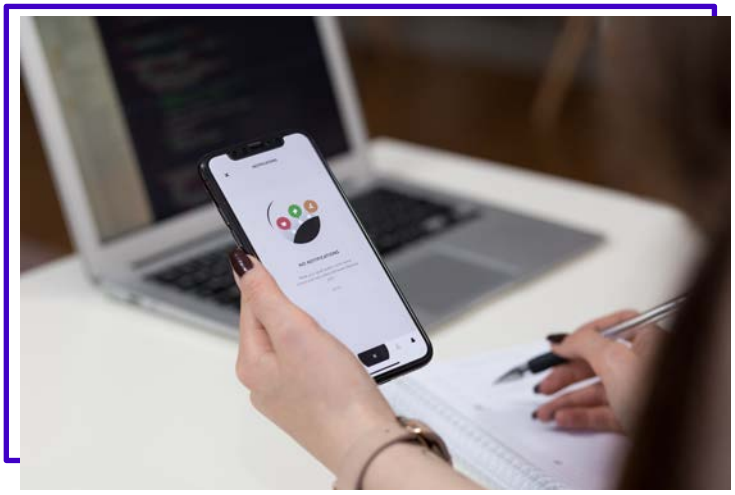
- Advantage CSP
- WordPress
- Drupal
- Umbraco

The magic of connection

By building two separate but linked entities, you can get the best functionality from both and build a positive brand and customer experience.

Organizations can have robust security, fast check-outs, rapid product loading, and database search and filter functions that are table stakes for an eCommerce platform.

And at the same time, they can use a CMS to build and manage workflows, create additional content (blogs, forums, landing pages, how to's, product pages, guides, portfolios, etc...) that's easy to manage and easy to consume.



But the real magic comes when both are combined.

By linking a CMS to an eCommerce platform, organizations can offer a total customer experience. This starts with fulfilling the basic product promise — to buy things online (the eCommerce platform). But it can extend that core functionality to other aspects of the customer's life with a CMS.

For instance, if you are an online, high-end men's clothing company, you can probably assume the guys who buy your product are interested in men's fashion. You can use a CMS to attract audiences in via content marketing and link them directly to an eCommerce platform to drive a better sales experience. On the other end, a CMS can help you serve relevant, timely content to the right person at the right time, nudging them to purchase even as they drop away from the online store.

It's only by owning and linking both that companies can meet customers' expectations.

You should be aiming for Stage Two if:

- You have an active eCommerce platform and a CMS
- You have a joint CMS/eCommerce platform (in which case, the first step is to split those into separate products)
- Your eCommerce and CMS are completely disconnected
- Your eCommerce platform is either fulfilling content and workflow requirements, or there's a major disconnect for the customer between their experience of your brand and their experience of your online store

Stage 3: Microservices

The final stage for an agile approach to eCommerce is microservices.

Microservices are a true embodiment of the modular, DevOps-inspired approach to architecture that we've been talking about through chapters 1 and 2, and is (in our opinion) the best design for an eCommerce solution.

So what are microservices?

According to Chris Richardson, the founder of microservices.io, microservices are:

"an architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities. The microservice architecture enables the continuous delivery/deployment of large, complex applications. It also enables an organization to evolve its technology stack."

What this means is that instead of building a single, monolithic IT tool or environment, you build dozens of small applications, each one doing a highly specific role. These are then linked together to deliver your online product or service.

And if you remember earlier our definition of agile eCommerce, a microservices architecture is a natural fit that encourages rapid innovation and testing.

Advantages of microservices

There are a couple of advantages to microservices architecture.

First, it is completely modular. If something breaks or you develop a better solution, you can rapidly replace that single module. This makes continuous improvement much easier.

Second, it is more resilient. A single module can fail while the architecture as a whole continues to work.

Finally, microservices are extremely scalable. Loads can be easily distributed to handle traffic spikes are high-use modules are shifted to different servers. This is much more difficult to do with a monolithic design.

These benefits of microservices — which are connected via the APIs from stage one — make them incredibly popular for organizations and help deliver an architecture that serves customers.

You should be aiming for Stage Three if:

- You have APIs that connect various elements and levels of your architecture
- You have a robust CMS platform and a robust eCommerce platform already
- You're struggling to keep up with traffic spikes and provide the customer experience your customers demand
- You're looking to scale your company and your eCommerce operations significantly
- You want to embrace a test-centric, continuous improvement approach to development

Chapter summary

- Agile eCommerce enables companies to connect a variety of systems using the technology that works for them to present a seamless product to the customer, regardless of how the customer chooses to engage. Agile eCommerce comes in three stages.

- Stage one is all about connecting existing systems with APIs to move data effectively from A to B. This also lays the groundwork for later stages by making various parts of the tech stack more adaptable and flexible.

- Stage two is about connecting your CMS to your eCommerce platform. eCommerce platforms are great at a very specific thing, and by connecting them to a CMS, you can leverage a much wider range of functionality that ultimately lets you serve your customers better.

- Stage three is about converting your monolithic architecture into microservices. This splits out all the functionality into much smaller units, each of which is connected via APIs. This allows each small service to be tested and iterated on extremely quickly, makes the system more reliable overall, and enhances scalability as high-user services can be allocated to servers with more space dynamically in response to requirements.

Chapter 3: What to Do Next?

So far we've talked about the different types of eCommerce architecture and the different stages that businesses progress through as they move towards truly agile online experiences. Now it is time to put this into practice.

In this final chapter, we are going to cover:

- How to transform your business strategy into an eCommerce one
- How to structure the customer journey for your eCommerce store
- How to develop a product roadmap.

By the end, you should be ready to embark on your own eCommerce transformation.

Setting eCommerce strategy

Building an eCommerce strategy that's agile, dynamic, and set to scale is hard. But the steps involved don't have to be. When you are deciding what and how you want to invest in eCommerce, you need to undertake the same 5-step exercise as when you are setting any business strategy:

1. Establish a vision.
2. Define your high-level goals.
3. Break your high-level goals achievable objectives.
4. Define objective KPIs — how are you going to measure success?
5. Set and deploy tactics to hit your KPIs.

When you think about this in terms of eCommerce strategy, you are usually starting at step 2 or 3.

Why? Because the business sets the vision, and usually the objectives.

How to uncover objectives for eCommerce sites

Once you have a vision and a goal for the organization, and you've decided that an eCommerce site is the best way to get you there, you need to establish clear objectives for the project. This will help define project scope and help you choose an architecture that will serve your needs in the long term.

First, understand the business requirements.
The first thing you need to do is understand the business requirements. How do the goals and vision relate to your specific eCommerce strategy, and what are the needs of the business? To do this you'll want to:

- Talk to stakeholders and capture their feedback
- Organize stakeholder feedback into themes
- Interpret those themes and turn them into objectives.

Second, understand the tech landscape.
Integration between different tools remains a core challenge for technology improvements and implementations. The problem, according to Ruben Mancha, an assistant professor of information systems at Babson College is that:

"[teams] see technology as individual pieces to implement, but don't have a value proposition that integrates all of them."

Basically, when it comes to eCommerce solutions, it is easy to look at the problem with blinders on. Instead, organizations need to consider the broader picture: what information do we need that exists elsewhere, and how are we going to get it to where it needs to go?

One example of this is our work with one of our clients in the publishing industry. The client had a huge number of products that they needed to sell through their eCommerce platform, across multiple sites and brands. We realized that the best way to offer these products was going to be via a connection to their ERP. Their ERP was their single source of truth for their 200,000+ SKUs, and by integrating with it, we were able to make their life a lot easier because inventory data was always in sync and up to date.

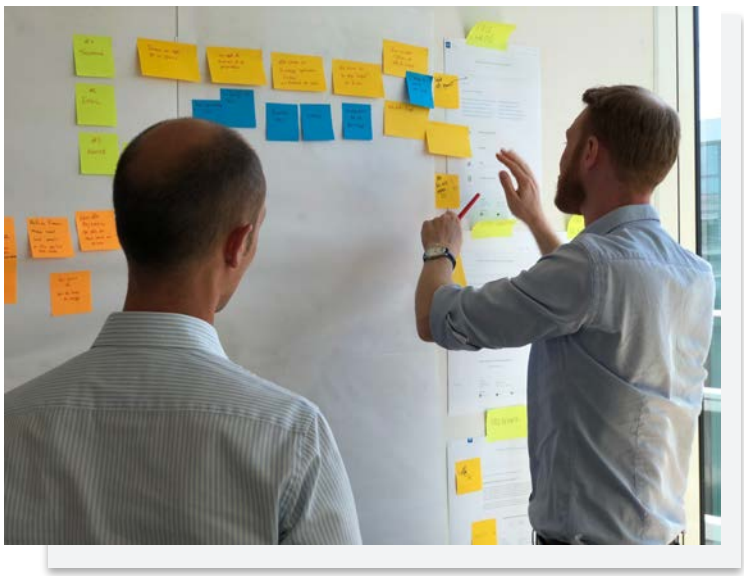
Third, identify any specific challenges.
Finally, when you are building objectives for an eCommerce site, you need to keep in mind any specific functionality a project requires.

For example, when we worked with Teknion, a major challenge was getting up to date content to their global sales and marketing teams. By building multiple global sites that could be managed centrally, we solved both the problem of global teams not having the right information and the problem of managing it from a single location.

Establishing a customer journey

Once you have your objectives and strategy in place, you need to build a good understanding of your customer journey.

A customer journey is what it sounds like: it is a map of your typical customer experience with your product. Generally speaking, the journey map is a map of touch points — where the customer engages with your business and what action they take at that engagement.



Customer journey mapping should be specific to an audience type. Particularly for B2B products that are purchased by committee, it is important to understand the journey different buyers will go on. This might not have to be as granular as a persona, but different maps for, say, different departments or buying centres.

The point of journey mapping in terms of eCommerce is to understand where your eCommerce is going to enter the customer journey and the role it plays when it is there.

By understanding these two things in the broader context of the total customer experience, you can build a site that is better at doing what it is supposed to do, and more tailored to your specific customers and their experience.

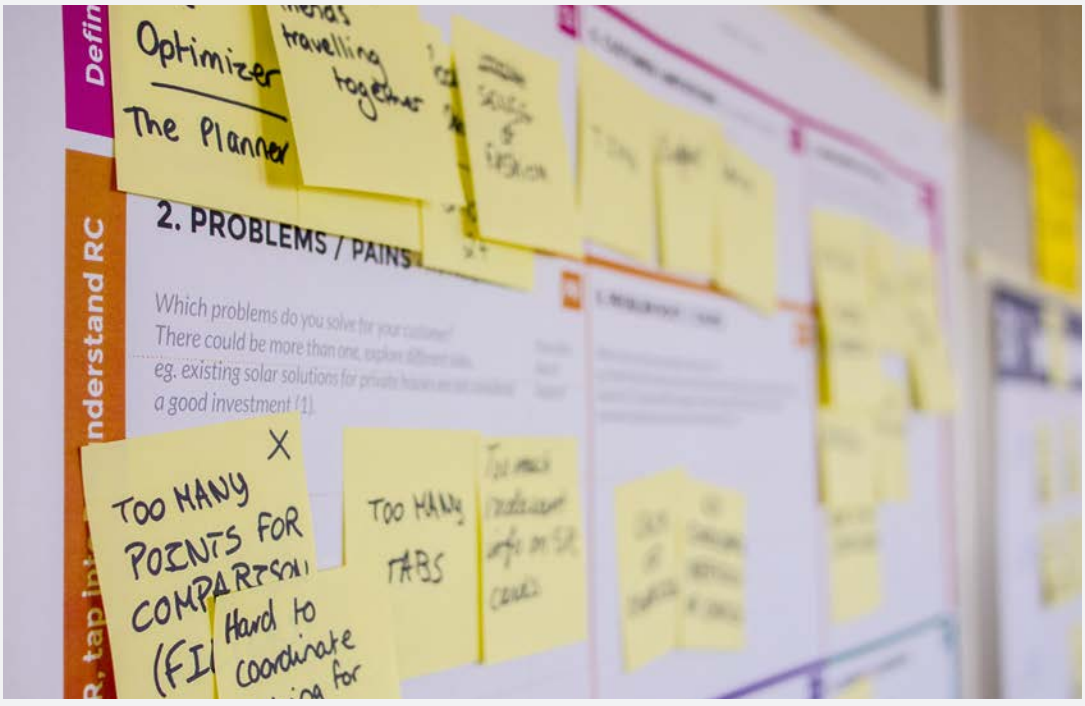
For instance, say you are a clothing company. You might find when you are undertaking this activity that most eCommerce visits come before a visit to a store, slotting into the discovery period rather than the purchasing period of the customer lifecycle.

That's a key insight that can drive your eCommerce project. For instance, you might make it easier for users to hold clothes in store from your eCommerce platform, or make it easier to return clothes purchased online that don't fit.

The point is that you are making decisions to improve the total customer journey, which might be different from the decisions you would make if you looked at things in isolation.

It's also important to consider the future of your eCommerce business. Where do you want it to go, and what do you want/envision the journey map looking like in 5-10 years?

Considering what you want and how you think customers will engage in the future can help you build the right technology now so you don't have to redo a bunch of work later.



eCommerce architecture

Once you have your objectives and strategy in place and have linked your eCommerce site to a broader business goal and vision, it is time to consider your architecture options.

As we've mentioned, we advocate a headless, agile architecture. Not only is it a more dynamic way to deliver eCommerce to customers, but it is by far the best way to prepare your digital properties to scale.

But there are some downsides. Out of the box, traditional eCommerce solutions are extremely well-developed. They have robust functionality that will suit the needs of some organizations perfectly, especially in the early stages of a company's growth.

So once you've established objectives, you need to map that to existing eCommerce architecture options.

Traditional eCommerce

Traditional eCommerce suits businesses who:

- Have limited requirements
- Need a solution extremely quickly
- Don't have a lot of integrations with a broader tech landscape
- Only need an eCommerce solution for a small part of their overall strategy
- Have a very stable user base who is unlikely to adopt new technology

Headless eCommerce

Headless eCommerce suits businesses who:

- Need a solution for a large national or global eCommerce platform
- Need to integrate into a complex tech environment
- Want to future-proof themselves against future content formats
- Want to future-proof themselves against future devices and engagement methods
- Want eCommerce to be a cornerstone of their go to market strategy

The idea isn't to select a specific architecture. It's about choosing what's right for you. And by understanding your business goals and vision, your customer journey, and the options available to you, you can fit the solution to your specific needs.

The only thing left to do is create a roadmap.

Product roadmap

The final step for what to do next is to create a roadmap.

If the customer journey is all about how the customer experiences your brand, then the product roadmap is all about how you deliver the experience your users need.

And this can be broken down into three main parts.

1. Transform requirements into technical specifications
The first step is to transform your requirements into technical specifications.

What technical capabilities do you need to deliver on the business objectives and goals and meet the customer journey requirements?

By understanding your requirements in terms of technical specs, you can begin to build an idea of what you need to accomplish. This can be done before or after you choose an architecture, but we recommend after. It's much more difficult to define a roadmap when you don't know what you are building on.

2. Prioritize your technical specifications
What's the most important thing? What's going to make the biggest difference for your customers? Those activities should be priority #1.

It's also worth thinking about how much work (roughly) each requirement is. A high-value, low-effort requirement should receive higher prioritization than a high-value, high-effort requirement. It will also help you identify where the quick wins are.

3. Turn your priority list into a timeframe
Finally, plot those requirements on a calendar. See when you can achieve specific milestones and when you deliver actual improvements to your audience.

By mapping this work and getting consensus, you can deliver a project on your terms, control scope creep and make it clear that the priority of a task isn't based on someone's ideas, but rather on what the customer needs most.

Basically, the product roadmap is a shared understanding of where an organization is, where it is going, how it is going to get there, and when.

By building a detailed, research-backed one, you can keep your eCommerce project on schedule with ease.

Examples of product roadmap software

Obviously, there's a lot of software that can help businesses walk through these stages. But here are few that we recommend.

Trello
Trello is fast and incredibly easy to use. It allows product roadmap owners to create Kanban boards, which you can then populate with discrete cards. As tasks get completed, the cards are moved from column to column.

The power of Trello is threefold. First, it is so simple and intuitive, it doesn't take a lot of management time. Second, each card has plenty of functionality, with checklists, the ability to upload information, tag people, and assign cards. And finally, Trello has a huge

range of add-ons, like calendar views, integrations, and more to make their customers' lives easier.

Aha!

Aha! is probably the best-known product roadmap tool. It is more powerful than Trello but has an understandably steep learning curve and is more software-specific. However, that extra power can be essential, especially as you get into the weeds.

Two functions, in particular, stand out on Aha!

First, they have a superb built-in roadmap calendar, letting you track development work, features, and releases easily over time.

And second, they have a simple way to gather, categorize, and view ideas. By inviting clients in or integrating with a feedback system, you can easily track feature requests and prioritize based on need.

Notion

While Trello is easy to use Kanban boards and Aha! is industry standard, Notion takes a different slant. Their idea is that product roadmapping happens in dozens of different software. So Notion is all of those platforms combined.

While it is not as powerful as any one thing, their ability to offer a beautiful interface for a huge number of business applications is a compelling value proposition. What's more, they recognize that for most roadmapping activities, you only need maybe 30% functionality from all your apps. Notion gives you that in a single, beautiful interface.

Chapter summary

- After you've reviewed your architecture options, the next thing to do is develop objectives based on business goals and a business vision, build a customer journey of touch-points, choose the right architecture, and prioritize requirements in a roadmap.

- Building objectives for an eCommerce site is a three-step process. First, understand the business requirements. Next, understand the tech landscape and any integrations you need. Finally, identify any specific requirements that your project might have.

- A customer journey is a series of touch-points and what the customer does at each one. By taking the time to map a customer journey, you can get clarity on what your site needs to do at each specific point.

- Next, choose your architecture. Based on the objectives, goals, vision, and customer journey, choose an architecture that works for you. We recommend headless eCommerce due to its flexibility but traditional eCommerce may suit some organizations.

- Finally, turn your requirements into a product roadmap. Transform your requirements into technical specifications, those, and then plot them on a calendar to get consensus around what's in scope and out of scope, and when the work can be expected to be completed.

Final Thoughts

It might seem hard to stay ahead in terms of where eCommerce is going. It might even be hard to get on board with the fact that your business has to change. Fortunately, the change doesn't have to be that hard. What we have learned over the years is that most change gets caught up not in the technology, but in the organization.

Hopefully, this ebook has helped you understand what the future of eCommerce is, and provided some tips on what you can do next, as it doesn't have to be an 18-month long project fraught with frustration and pain.

By approaching eCommerce is a staged, modular, and agile way, organizations can identify their challenges, roadmap a solution, and test and deploy that solution quickly and effectively. All without disrupting the overall business or the broader IT environment.

By embracing the modular design of headless eCommerce, hooking up disconnected technologies with APIs, embracing microservices, and building a data- and needs-driven roadmap, businesses can come out ahead.

eCommerce is changing. However, it is not a cause for alarm. By improving your solutions, even incrementally, you can start to advance your total customer experience, customer journey, and make your eCommerce solution work for you.

The future of eCommerce is bright and, in our opinion, there for the taking. So go, transform your business. And if you hit a snag, you know where to find us.

Thank you for the opportunity to share this book with you. We hope you have found it helpful on your path to make your organization a digital leader.

If you have any questions about your eCommerce or digital strategy, get in touch with us using the contact information below.

We'd love to hear from you.

info@enginess.io
416.901.6151
enginess.io