Monolithic architecture vs microservices

PIOTR KARWATKA, 14/01/2020

Microservices development for eCommerce is effectively pushing monolithic architecture out of the market. Industry leaders are now building their empires on microservice architecture. But why is this approach to architecture chasing out the traditional monolithic structure? We're going to take a look at monolithic architecture vs microservices to find out which is better for business and why.

What are the pros and cons of monolithic architecture?

It is a traditional model for software in which the structure is a single and indivisible unit. A monolith has one code base with multiple modules. This architecture has been the go-to functioning model for many years and countless applications have been successfully built as monoliths.

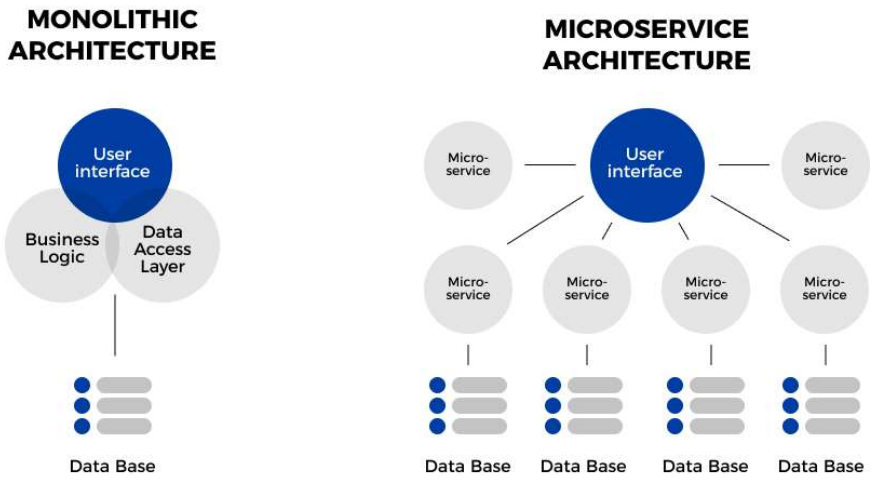
The advantages of monolithic architecture

- Starting a new project and developing it is simpler using monolithic architecture
- It is much easier to test a structure that is a monolith
- Components like frameworks, templates or scripts can be easily applied
- Deployment is very simple. All you have to do is paste the previously prepared application to the server

The disadvantages of monolithic architecture

Like any solution, monolithic architecture has its drawbacks. Here's a list of some disadvantages:

- A large code base can be significantly harder to understand
- The Integrated Development Environment can become overloaded, and size may also slow down startup time
- Every element is closely related and dependent on the others, so it is difficult to change to a new or advanced technology, language, or framework
- Updating can be a challenge as it redeployment of the application
- Problems with scalability, because each element has different resource requirements



The advantages of microservices architecture

One of the most popular arguments for getting into microservices architecture is that it can make the software easier to maintain by

decomposing and decoupling it into smaller units.

Considering monolithic architecture vs microservices, most monoliths are not well modularized and their features are not easy to test (unit testing, behavioral testing). It becomes difficult to deploy the changes because of the tight-coupling between the modules it's quite easy to break some other features.

The advantages of monolithic architecture vs microservices

It's worth noting that the problems with monoliths can be fixed without major re-architecting of your application. For example, Shopify started as a startup with a Ruby on Rails base and now they're perhaps the only force in the world to compete with Amazon.

Ruby on Rails is well known for its flexibility with a "convention over configuration" approach that has just a few strict rules on designing the application. You can get results fast and it is a fun and productive way for developers to work. In fact, startups often choose Rails because they need to be fast, validate, and pivot. In essence, if Shopify had started with microservices architecture, complex deployment schemas, and complicated CI/devops processes, they may have failed.

Monolithic architecture is the easiest to implement. If no architecture is enforced, the result will likely be a monolith. This is especially true in Ruby on Rails, which lends itself nicely to building monoliths due to the global availability of all code at an application level. Monolithic architecture can take an application very far since it's easy to build and allows teams to move very quickly in the beginning to get their product in front of customers earlier.

Maintaining the entire codebase in one place and deploying your application to a single place has many advantages. You'll only need to maintain one repository, and be able to easily search and find all functionality in one folder. It also means only having to maintain one test and deployment pipeline, which, depending on the complexity of your application, may avoid a lot of overhead. These pipelines can be expensive to create, customize, and maintain because it takes concerted effort to ensure consistency across them all. Since all of the code is deployed in one application, the data can all live in a single shared database. Whenever a piece of data is needed, it's a simple database query to retrieve it.

Source: Deconstructing the Monolith, Shopify blog

Why do microservices work well for large projects?

Unlike monolithic architecture, which consists of one element, microservices are created by cooperating services. Each of them is a small-scale service that performs functions from a single business area. Microservices were born out of SOA (Service-Oriented Architecture), as a solution to the challenges associated with monolithic applications. The division of large projects into small ones has gained more importance, especially among developers, and the impact of microservices on eCommerce is significant. One of the main reasons why the independent modules met with positive reception is its ability to manage the API and perform specific tasks. Microservice architecture divides services into small, autonomous elements, each with its own independent function.

Why has microservices gained such popularity among key players like Zalando, Netflix, and Amazon? Primarily, a departure from monolithic architecture has made it possible to relieve programmers. They no longer need to maintain large projects. Thanks to microservices, they can focus on small parts of independent projects. It is also easier to introduce new people to the project. It is not necessary to familiarize them with the entire monolith, but only a small part of it.

> *"When businesses come to us looking to replace their monolithic loyalty program architecture, their most common reason for wanting change is that their technology is 'limiting' their ambitions and capabilities.*
>
> *Monoliths don't necessarily limit businesses in terms of features. You can build any feature you like with enough time and resources. They limit the way that brands can interact with customers and offer more modern experiences. And the speed at which they can do it.*
>
> *A headless approach, using microservices exposed through an API, lets companies deliver those experiences faster and allows them to evolve as the relationship with customers develops over time."*
>
> *– Cezary Olejarczyk, CEO, Open Loyalty*

The keys to a successful microservices build

I'd risk the statement that a successful migration towards microservices architecture depends more on the proper organization structure than any other factor. In my recent interview with Kelly Goetsch, we concluded that one of the key factors for changing the architecture is to have a project sponsor with a budget and vision. On top of this, you then need the proper team structure. Only those factors – combined – will let you reach full speed.

We can also say that successful architecture stands on three legs: a proper domain model, people, and an operating model. The structure of the people, divided into the teams and their operating model differs from company to company, product to product, project to project. However, I really liked the observation made by Sam Newman in his book "Monolith to microservices" that the best structure is the one which gives the team full ownership. Usually, the ownership means the independent deployability of the services the teams are working on.

> *"Adding more people will only continue to improve how quickly you can deliver if the work itself can be partitioned into separate pieces of work with limited interactions between them"*
>
> *– Sam Newman, Monolith to Microservices – after the "Mythical Man Month"*

This means that, to perform well, the team has to have full competences (developers, design, and DevOps) to build and ship new features to production. They should not need to communicate with different structures. Of course, they need to fulfill the API service contracts and interfaces. Other than that, they should be able to experiment and ship new features solely on their own.

Monolithic architecture vs microservices comparison chart

What architecture should you choose for your business? We've broken the decision down into the different areas that you might consider and assessed each one in both scenarios.

Area	Monolith architecture	Microservices archit
Deployment	This architecture allows you to deploy once and then customize your solution based on current changes. But if something goes wrong, the whole project will break down.	Deployment is a con process in microserv Each microservice n implemented separa extends the impleme process. If somethin goes wrong, only on microservice will be and it will be easier
Maintenance	Maintenance in a monolithic architecture requires an IT team that specializes in multiple platforms, like Pascal, .NET, Java, or DB2. Finding errors and making changes takes a lot of time in the	Maintenance in micr is faster than in mor Smaller services are saving programmers Over time, it increas efficiency and saves

	monolith. However, testing itself is simple and will be done in one go.	
Reliability	If it comes to reliability, the monolith has no chance against microservices. If something goes wrong in monolithic architecture, it can stop the whole structure. Meanwhile, in microservices architecture, breaking one service will not cause critical problems in the overall application.	Microservices are st mostly reliable. Brea only affects that ele others remain intact allows for a fast pac and the introduction one function withou others.
Scalability	Scalability in monolithic architecture is difficult to achieve due to the size and scale of the structure. This option is hard to upgrade.	In the case of micro scalability is much e we can scale only th require more resour
Development	The development of monolithic architecture takes a little bit longer than microservices. This is due to the need for a parallel operation on the same code by all teams.	Microservices guara development. Teams have to work in para do in monolithic arcl because each applic be delivered indepen
Releases	A monolith is a one-piece structure which cannot be separated into smaller elements.That is why everything must be ready before release. Potential delays in the work of the team will impede the entire project.	Due to the structure allow for faster relea features.
Cost	Monolith architecture is more affordable and faster to develop, but each case must be considered individually. For businesses, monoliths involve one huge investment, which is a higher risk and a bigger strain on the budget.	Microservices are us expensive, and the w development takes r than in monolith. Bu run, they can cost le developers' work tim the long-run will be than with a monolith

Monolithic architecture vs microservices: Which to choose for your company?

Like everything, these two technologies both have their advantages and disadvantages. Monolithic architecture is a solid solution for simple eCommerce applications or blog platforms, especially when ongoing changes and development are not predicted. Microservices have grown to become a better fit for complex applications and are the modern solution in a time

when constant improvement and development of sites and services is the norm.

*This article is an updated and augmented version of a piece originally written for Divante by Ola Mazur and published in 2019.