Are Microservices the real game changer to product-driven businesses?

Practice Head

Aju Mathew
VP software engineering

Author

Jothi Rengarajan
Principal Architect

aspire SYSTEMS
attention. always.

TABLE OF CONTENTS

If a product based organization has built a software and if the software is successful, they would face the challenges of monolithic architecture and would contemplate to the migration of microservices. This paper discusses on why microservices migration is essential for growing product based organizations and how embracing it would prove successful.

Vision, goals and action:

Before jumping into microservices migration directly, it is important to know how the benefits of microservices migration would map back to a product driven business vision.

The best way to achieve this is to clearly analyze the vision, map the goals for the same and analyze how microservices fits into the same.

Following table provides a sample structure of vision, goals and action

Vision	Goals	Action
○ Be a market leader in the space in next 4 years	○ Improve customer satisfaction index by 10% ○ Decrease operational cost by 30% ○ Grow by 25% every year	○ Move to an architecture that supports continuous deployment easily ○ Move to an architecture that reduces maintenance cost significantly ○ Move to an architecture that can scale exponentially and offers economy of scale

The above action points regarding architecture can be further refined once we understand the benefits of microservices architecture.

Direct and indirect benefits of microservices

Direct Benefits:

Change is inexpensive: Each microservices deals with single responsibility and hence changes can be easily made without affecting any other parts of the system.

High Scalability: As each microservices is an independent scaling unit, microservices can be independently scaled.

Faster release time: Since any changes is done faster, release time is also reduced significantly.

Increased Resilience: Failure in any one of the services does not affect the other.

Decreased Maintenance Cost: Microservices architecture enforces stability and hence it is very easy to maintain a properly designed microservice almost cutting the maintenance cost to half.

Indirect Benefits:

Easier Team Expansion: We all know that it is easier to find developers and application teams to work on new applications rather than an existing application. With microservices, every microservice is almost like a new application and hence it is easier to add new teams much faster to work on them. Also a microservice is easy enough for anyone to understand making it much easier to induct teams to work on it.

Opex instead of capex: Once you have the base, it is very easy to add more and more microservices at scale and hence instead of looking at the cost per application, the costing changes to per microservice. This allows to incur on-demand development cost with ease rather than a huge development cost upfront.

Easier innovation and experimentation: Since it is very easy to add as well as discard microservices, it is possible to quickly and cost effectively build a new feature using microservices to test.

Competitive Edge: This is an extension of the above point. As innovation and experimentation is easier, it provides the product based companies high competitive edge.

Easier Course Correction: Many applications contain several features that are never used. This exist in the code because of a fear that it would break something else. With microservices, it is very easy to continuously assess business values and retire features which do not add significant value. This leads to significant saving in maintenance cost.

Now that we have clearly established both tactical and strategical benefits of microservices, the above table of mission, goal and action can be transformed to

Vision

- Be a market leader in the space in next 4 years

Goals

- Improve customer satisfaction index by 10%
- Decrease operational cost by 30%
- Grow by 25% every year

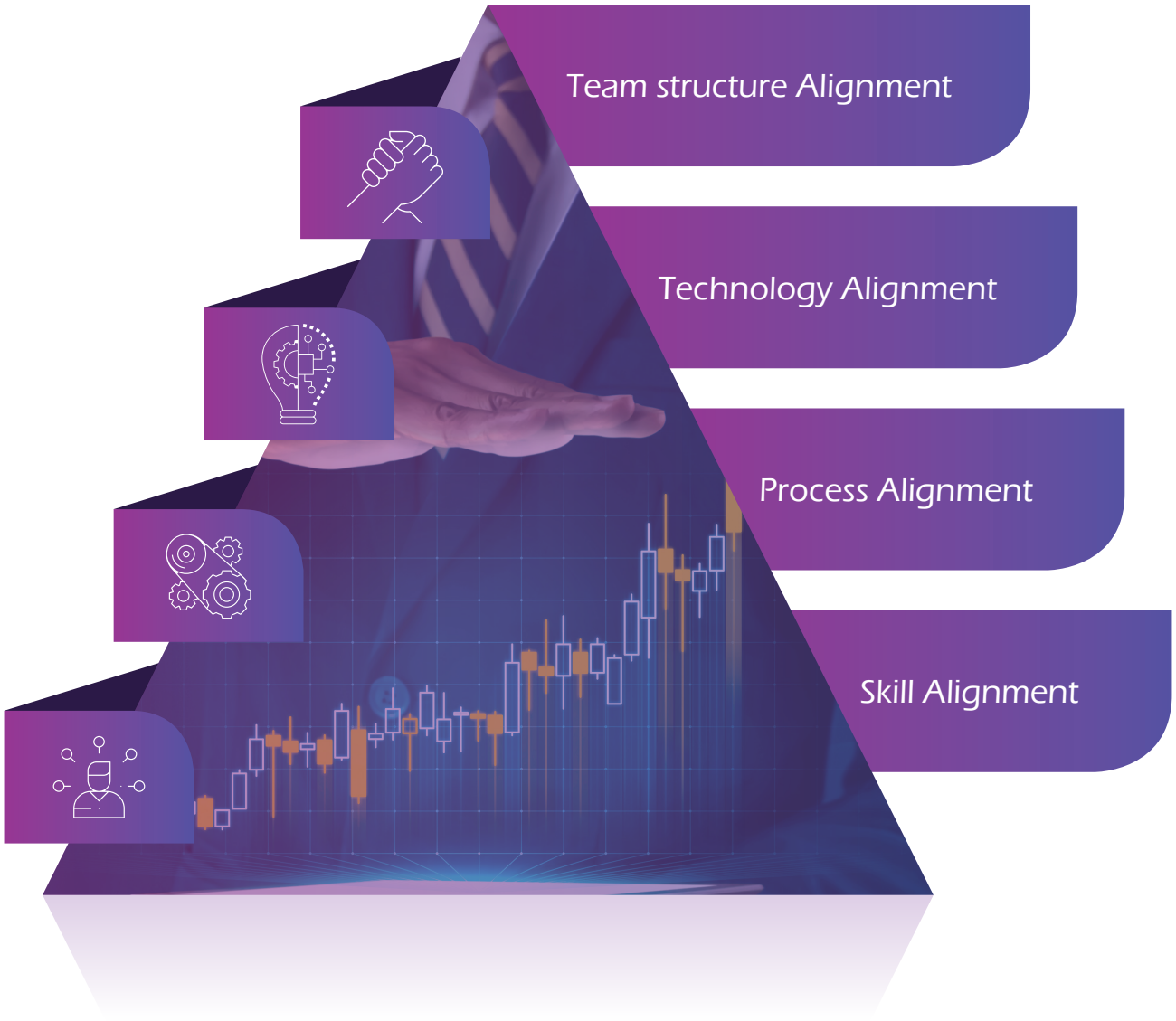
Action

- Move to microservices architecture that supports continuous deployment easily
- Move to microservices architecture that reduces maintenance cost significantly
- Move to microservices architecture that can scale exponentially and offers economy of scale

Achieving Microservices Migration:

Once a product driven organization decides that microservices is the right way forward, the next step is to consider end to end factors in this movement. In my experience, I have noticed many software vendors either crawl or fail because they do not account for all the important paradigms from the start.

Many think that microservices migration is only about dealing with technical changes. Though technical alignment is very important, there are other factors that are accountable for when an product based enterprises moves towards microservices architecture.



Team structure alignment:

According to Conway's law, product driven businesses that design systems "are constrained to produce designs which are copies of the communication structures of these products." This means that the structure of your software should reflect the structure of your software development software vendor. Teams that are aligned to the services can be organized in such a way that allows them to own what they're responsible for from end to end. Amazon calls this as "you build it, you own it," or "build and run" teams, responsible for development and production throughout the entire lifecycle for a chunk of software. The team size for one build and run team should not be more than 10. Since ISVs are going to end up with multiple smaller teams at any point, they should also build overall governance team which spans across all the teams for providing governance and also align individual goals of the team to an overall goal.

Process Alignment:

While the overall agile methodology holds good for microservices development, there are many things that we need to bring into the process that might not be relevant for monolithic applications. Following points cover the most important ones

1. At the beginning of the microservices development, atleast 3-4 sprints are dedicated to define the software architecture, infrastructure architecture, DevOps architecture, standard guidelines and practices for the microservices. This is very important to avoid massive reworks and failures later. A general naming we follow for these sprints are "Foundation sprints".
2. Once the development has started, multiple parallel scrum teams need to run for the microservice that are in development respectively. A **scrum of scrums** needs to run aligning the different delivery goals of the independent microservices to an overall goal of the product release.
3. **Testing strategy** needs to be different for microservices application. Microservices should not rely on manual testing. Since microservices are smaller in scope, the time required for automation testing is generally small and simple. Automated testing is the one that helps majorly in the faster release to market. Microservices without automated testing is definitely a road to failure. The types of automated tests that each microservice should develop are

    i. Unit Tests

    ii. API Tests

    iii. Event driven Integration tests

    iv. Saga automation tests

    v. Contract tests

    vi. End to end tests

Again, like any other application, these tests should follow a pyramid with highest number of unit test cases to lowest end-to-end tests.

4. A ***standardized DevOps process*** defines all the microservices. Adherence to IaC and pipeline as a code is crucial. IaC and Pipeline code library needs to be available from which each of the microservices can be chosen from. If necessary, microservices can alter it for their needs. This can be specifically monitored as task during scrum planning.

5. **Dependency management** can be highlighted during planning and developers need the habit of tracking where their microservices are used and ensure that their versioning strategy does not affect the dependents. It is highly recommended to use a specialized microservice dependency tracking system to do this.

6. **Release process** needs to be end to end automated for microservice. The stages for the release pipeline and the approval authorities need to be set in place from the beginning.

7. **Periodic audits** are conducted to ensure that microservices adhere to the right guidelines.

Technology Alignment:

Designing, delivering and operating microservices application needs to follow different mindset as opposed to monolithic. We will discuss the most technical principles below:

1. **API, Events, Bounded context as first class citizen**: People who are working on monolithic application tend to jump directly to database structures and classes when designing the solution. While working with microservices, the team should first think about APIs, events and the bounded context it represents. Only these things matters the most and the underlying implementation should follow only after these details iron out.

2. **Authentication and Access Control:** Authentication and access control in microservices have different needs as opposed to a monolithic application. As there are several microservices that are widespread, the best possible solution is to have a single gateway for all the external endpoints for the microservices. Granular access control is enforced by an individual microservice and an overall authentication check is enforced at the gateway. Another authentication need in microservices is for interservice microservices call. In this case, either the microservices can adopt a policy of full trust between microservices which does not need an authentication or adopt a policy to carry forward the authentication for interservice communication. These policies depend on the nature and sensitivity of the endpoint.

3. **Data Consistency:** Eventual consistency is the norm in microservices. Because of the widespread nature of microservices, a single business process and transactions can span multiple microservices. There are patterns to achieve this in microservices all relying on eventual consistency. This is an important shift in the mindset of the team designing the microservices

4. **Dependency Tracking:** At any point of time, multiple teams are going to be working on microservices. It is very important to minimize the dependencies by design. Though we can achieve it to a greater degree by properly compartmentalizing the microservices, there are going to be dependencies between them. A product company need to have a different mindset and depend on proper tools to visualize, track and validate the dependencies. Failing this will lead to microservice dependency chaos in a short while.

5. **Infrastructure Management:** Instead of deploying and managing a single microservice, there are numerous microservices that needs to be provisioned and maintained. Hence automation of the deployments and using a single pane where DevOps can provision and manage the infrastructure using the best practices. Entire release needs to be automated via DevOps. Monitoring should be enabled to monitor all the microservices via a single pane.

6. **Observability:** User calls can span multiple microservices and hence end to end tracing should be enabled for troubleshooting. This should be built right into the core design and should be enforced architecturally.

Skill Alignment:

Microservices are very different in the way of developing compared to monolithic and hence training engineering teams should be crucial and their thought process needs to realigned. There is a lot to learn and unlearn for every role in the engineering team right from the product owners to architects, testers, developers and operational teams.

Optimizing Microservices Implementation:

Once the microservice migration has commenced, it is important to track the goals and where an ISV stands post the migration effort. If there is significant difference between expectation vs achievement, there are possibilities that a product company has missed out certain important principles. Product based businesses have to optimize and take right corrective actions.

Following are some of the sample observations that leads to several issues:

Operational Cost

- Improper infra management
- Gap in skill alignment of Infra and Devops team

Maintenance Cost

- Improper testing Strategy
- Improper release process
- Improperdependency tracking
- Insufficient Observability across microservices
- Data consistency handing is insufficient

Release Time

- Improper devops process
- Scrum of Scrum team is not setup efficiently
- Gap in skill alignment of developers

Scaling Issues

- Improper infra management
- Improper boundary of microservices
- Gap in skill alignment of Infra and Devops team

Conclusion:

We have seen how a product driven businesses can achieve its goal of growth using microservices and how to adopt the right approach for migration. Microservices enablement just does not stop with the first step of migration. Product companies need to also measure the benefits and optimize the implementations to eradicate the gaps.

ATTENTION. ALWAYS.

aspire SYSTEMS

attention. always.

Aspire Systems is a global technology services firm serving as a trusted technology partner for our customers. We work with some of the world's most innovative enterprises and independent software vendors, helping them leverage technology and outsourcing in our specific areas of expertise. Aspire Systems' services include Product Engineering, Enterprise Solutions, Independent Testing Services, Oracle Application Services, Digital Services and IT infrastructure & Application Support Services.

we are currently over 2750+ employees and work with 200+ customers globally. We are headquartered in Singapore and have a growing presence in the US, UK, Middle East and Europe. For the ninth time in a row, Aspire has been selected as one of India's "Best Companies to Work For" by the Great Place to Work® Institute, in partnership with The Economic Times.

SINGAPORE +65 3163 3050 | NORTH AMERICA +1 630 368 0970 | EUROPE +44 203 170 6115 | INDIA +91 44 6740 4000 | MIDDLE EAST +971 50 658 8831

For more info contact
info@aspiresys.com or visit www.aspiresys.com