**How to implement microservices project**

**Evolving an existing application**

Now we go deeper into how to chart a course for a specific microservices application development journeys of different scale and with a view to a possible progression from smaller to larger scale transformations of existing applications.

While building microservice projects without a prior application are relevant or important, we agree on a “monolith first” approach when building microservices.  In short, this means you build your application in whatever way you can to validate your idea first. Then, you apply the principles in this blog series to scale and evolve your initial monolith into a microservices project. There is no value in the creating architecturally pure microservices that do not offer value back to the business.

There three areas that you need to understand to implement a successful microservices project—your business, your culture and skillset, and your technology.

**Understand and define your business needs**

Why are you thinking about moving to microservices? For many businesses, more efficient software development and operation practices are required to deliver value to the business faster and deliver better user experiences.

Before you can understand the impact of a microservices project on your existing applications and infrastructure, you must understand which parts of your business are moving too slowly to produce satisfactory results. In many cases, the organization’s systems of engagement (SOE) are causing the slow down. These systems are available through many channels – web, mobile, APIs, and the like. Lack of speed is the primary reason to move to a microservices-based architecture.

Before you can adopt a microservices-oriented approach, you and your business stake-holders must know what is not getting to market fast enough. Which pieces of the application need improvements and modifications to make them faster? Answering that question involves which parts of the existing monolith should be targeted for microservice evolution.

**Understand your culture and skill set**

While not specific to microservices-based architectures, a thorough understanding of an organization’s teams, culture, and skillset is critical to a successful digital transformation.

Typically, in engineering monoliths, most organizations are built into siloes, with teams participating as needed along the software development lifecycle. This often creates well-defined boundaries, with restrictive roles and responsibilities along those boundaries.

Microservices architectures can only succeed when teams have the power to own the complete software development and operations lifecycle.  Building cross-functional teams representing all roles and responsibilities is fundamental in implementing microservices-based architectures.  Everyone—from design to development to operations to business owner—works closely together and is often collocated.

Since every stakeholder is represented in the team by design, development, and operations, work can move forward more quickly, efficiently, and with clear focus on improving user experience to achieve business goals.

A cross-functional team also supports the rapid growth of individual skills across the team.  When a team owns everything the microservice is responsible for, everything from design to operations to runtime data, no single team member performs a single task. Often, front-end engineers develop database administration skills, while operations-oriented team members learn more about differences in user-interface frameworks. Expanding the skill set this way helps the entire IT organization succeed with microservices, as it is much easier to build new teams of well-rounded team members, instead of constantly looking for specialists to fill very specific roles.

**Understand the technology**

Unless you address the business problem and your team’s culture and skill set, you won’t be able to effectively implement the microservices technology and you will keep the same processes and structures in place.

Proper analysis of existing technology stacks vary widely from organization to organization, but the simplified approach we describe helps ensure both the initial and sustained success of your microservices projects. Starting small and defining iterative, progressive successes is a much more achievable and fruitful approach than a flashy, transform-everything-at-once approach.

The first phase of understanding your technology is to identify the coarse-grained services that are in the existing monolith.  Identifying these course-grained services helps you understand the complexity of the data structures, the level of coupling between current components, the teams who are responsible for these new coarse-grained services, and so on. A successful coarse-grained services review gives you a clear understanding of data boundaries, both inside of a given service and across services.

Once you identify the coarse-grained services, you must create a plan for how to evolve these coarse-grained services into fine-grained microservices. These microservices, based on your previous work, should all be working with similar data, owning their “own” data, and understanding what data they need to read from where and write to other services. From here, you can identify and implement resiliency, scalability, and agility of the individual fine-grained microservices.

APIs and microservices are two parts of the larger whole.  Once you have a better understanding of your fine-grained microservices, you will also have a better understanding of your interfaces, which interfaces are on the critical path, which interfaces are optional, and which interfaces you no longer need.  If you cannot map an existing interface or API to one of your coarse-grained or fine-grained microservices, most likely you can do away with it.

**Sizing the microservices effort**

The heavy lifting of understanding the business, understanding the team structure, and understanding the technology ensures your teams and larger organization are prepared to understand the entirety of the microservices evolution of any given monolith – whether it is in a proof-of-concept scope, pilot scope, or large-scale evolution scope.

With all the analysis and planning work complete, the next step is to define timelines, delivery velocities, and expected results.

In the next post ,we will consider development and operations patterns that you can apply in undertaking a microservices transformation