Executive Briefing: Microservices

Jeff Allison: Executive Briefing on Microservices

Productivity Implications and Testing Challenges

August 19 @ 7pm

meetup.com/SF-Bay-ACM

SKMurphy

Jeff Allison has 30 years of experience in the high technology computing and networking industries. He has held various roles in Hardware Engineering, Marketing and Engineering management. He has a proven track record of developing high power cross functional teams to solve complex engineering issues and drive methodology changes throughout the organization.

Jeff graduated from the University of Wales in '84 with a degree in Engineering. His first position was working in the Engineering Design Automation (EDA) industry. He spent many years on-site with customers helping them transition to a new paradigm for product development. During this time, Jeff helped many customers navigate and the rapid growth and later consolidation in the EDA industry.

In 1992, Jeff joined Cisco as an engineering manager at a time of dynamic growth in the company and the networking industry. Jeff was instrumental in helping Cisco satisfy customer requirements and differentiate from the competition by automating and optimizing Cisco's hardware development methodology.

He spent the next 20 years focused on customer satisfaction, product quality, development methodology. He led several initiatives that effected significant improvements to design productivity and quality. As a VP of engineering, Jeff developed trusted business relationships with many major customers.

Outline

I have spent last 2 years working with a MicroServices startup and their customers.

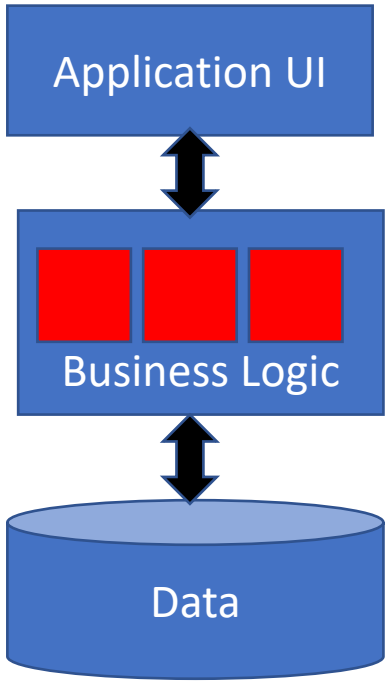
Last year, based on current and past experiences the technical leadership wrote a series of technical papers outlining strategies to migrate/start using microServices.

https://www.infoq.com/articles/twelve-testing-techniques-microservices-intro/
https://www.infoq.com/articles/twelve-testing-techniques-microservices-tradeoffs/
https://www.infoq.com/articles/testing-techniques-microservices-use-cases/

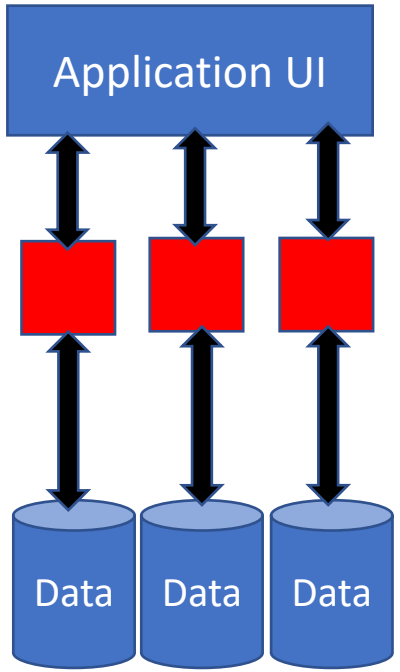
This presentation provides an “Executive View” of those strategies and a Framework/Approach to help Executives get started on this journey.

Definitions: Monolithic and MicroServices

A Monolithic application is built as a single unit.



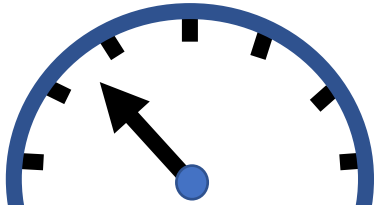
A MicroServices application is built using small self-contained services that are independently releasable.



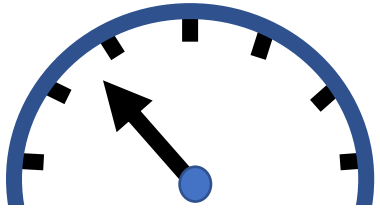
Business Logic : automation of business rules – ex how to validate a user, calculating tax on a purchase, shipping purchases etc..

Not all applications need a MicroServices architecture:

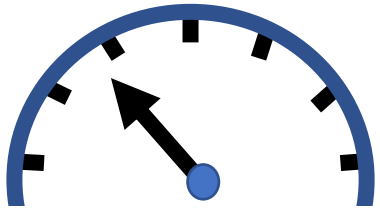
Application Complexity ?



Size of Team < 30 ?



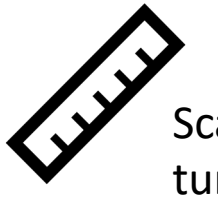
Business Requirements ?



Take Away: MicroServices only make sense complexity of the problem justifies it, otherwise, they can be more costly than a monolith.

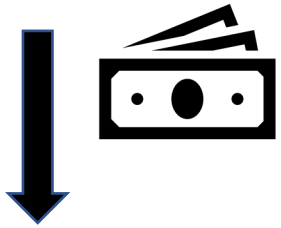
What is Promised : MicroServices Over Monolithics

Speed – Fast time to quality and performance



Scale – Fast turnaround Time

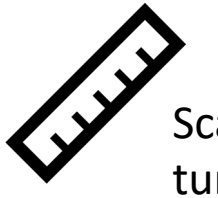
Responsiveness – Continuous Release Model



Lower Cost – smaller teams, scalable tools/test footprint.

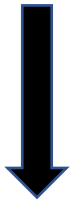
Executive Translation – Value proposition

Speed – Fast time to quality and performance

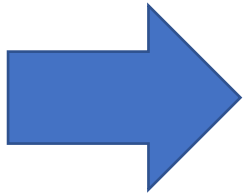


Scale – Fast turnaround Time

Responsiveness – Continuous Release Model



Lower Cost – smaller teams, scalable tools/test footprint.



Corporate Objectives:

Development Methodology and Product Architecture that will **scale to meet customer/market needs.**

Predictable Time to Market, Time to Customer Satisfaction and Time to Revenue.

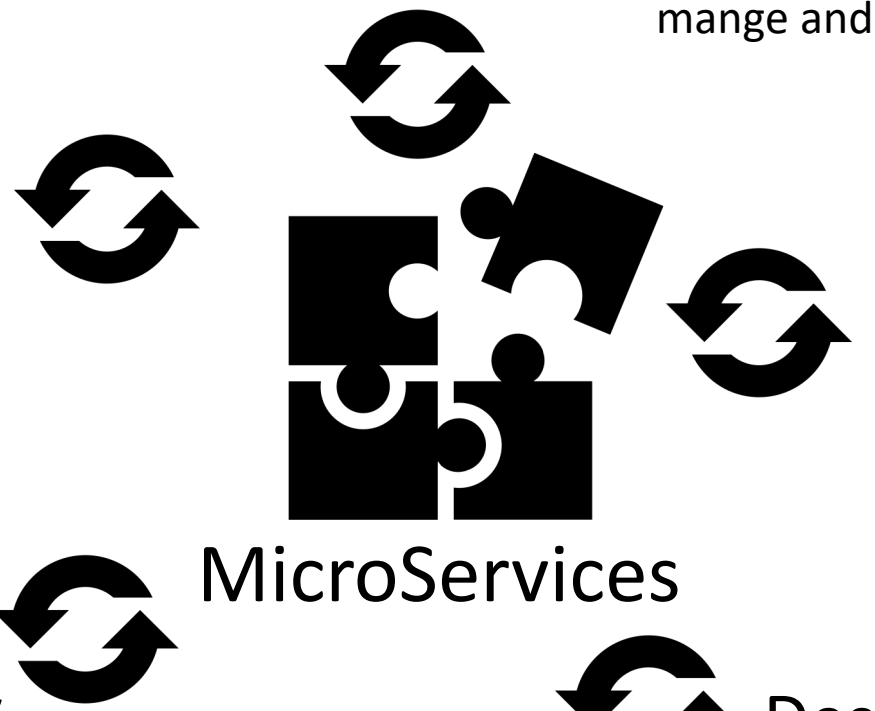
Why is it Difficult : Complex Moving Parts

Product Architecture

Map Business Logic to Services.
Assign Architects to mange and control

Organizational Structure

Restructure organization.
Conway 1967 :
Organizations design systems that mirror their own communication structure.



Development Methodology

Train Engineers on new design and development processes.

MicroServices

Test/Release Strategy

Develop processes and automation to manage services and enable continuous release.

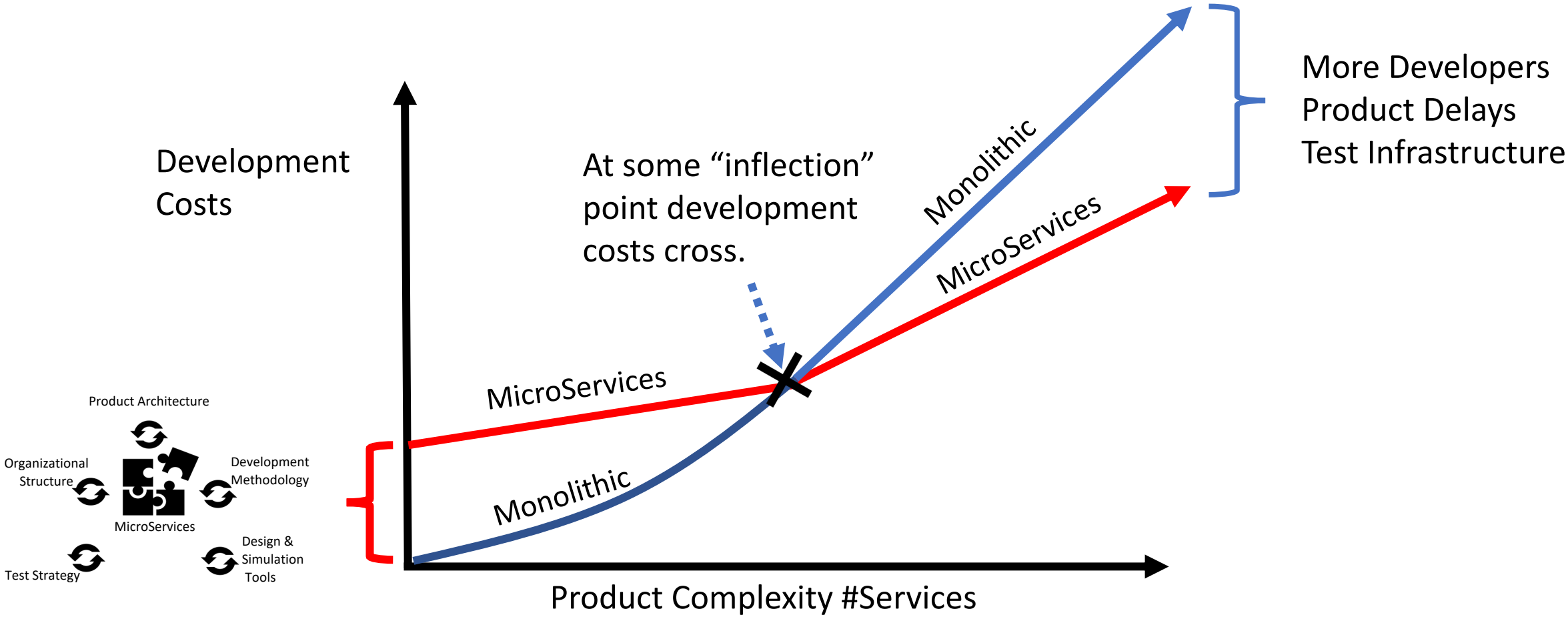
Design & Simulation Tools

Learn and onboard new design and simulation tools

Key Risks with MicroServices:

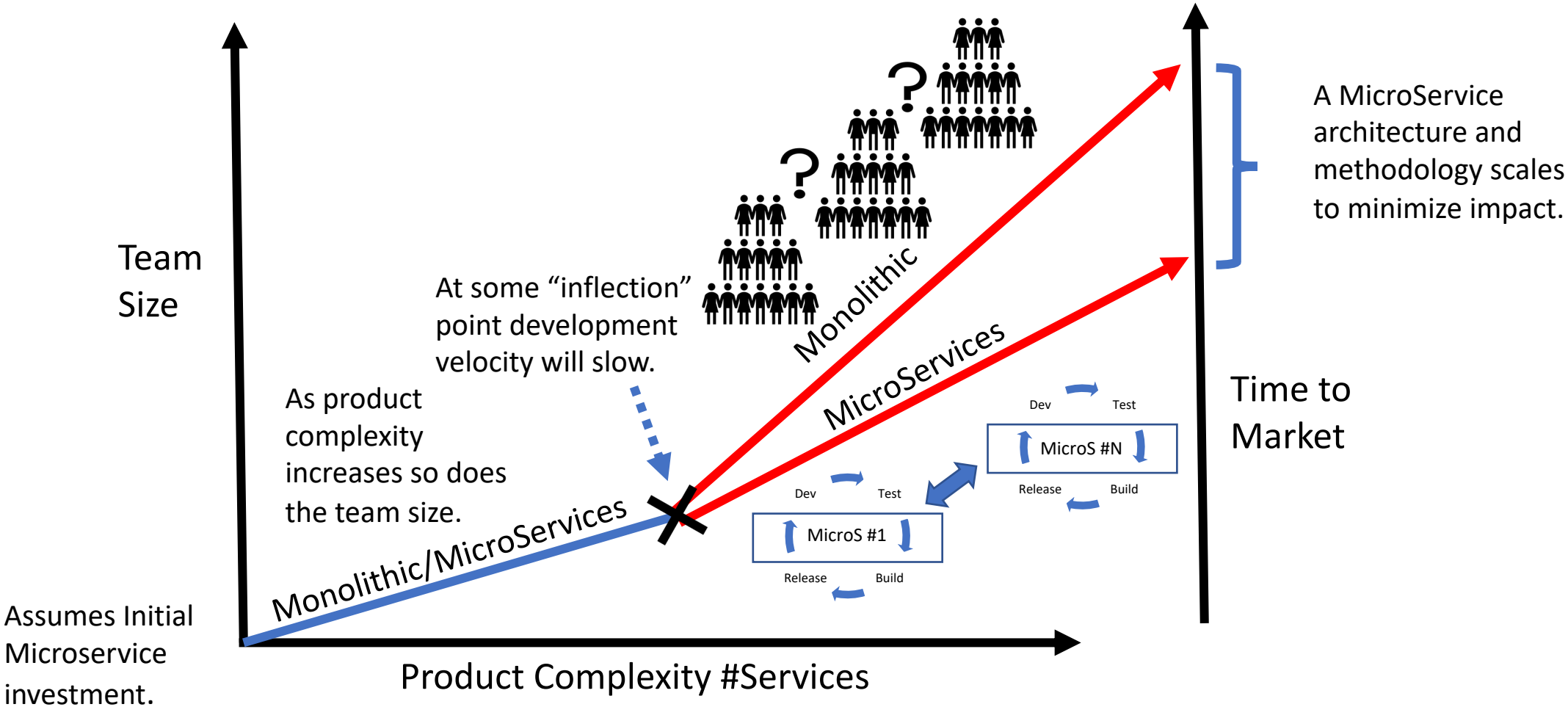
- Focus on security and governance
- Make sure you have enough telemetry data to diagnose issues in a distributed system
- Lack of infrastructure and test automation will impact any MicroServices efforts at scale
- Conway's law - communication patterns between teams

Cost Monolithic .vs. MicroServices: Need to invest up front in People, Process and Tools to Benefit later.

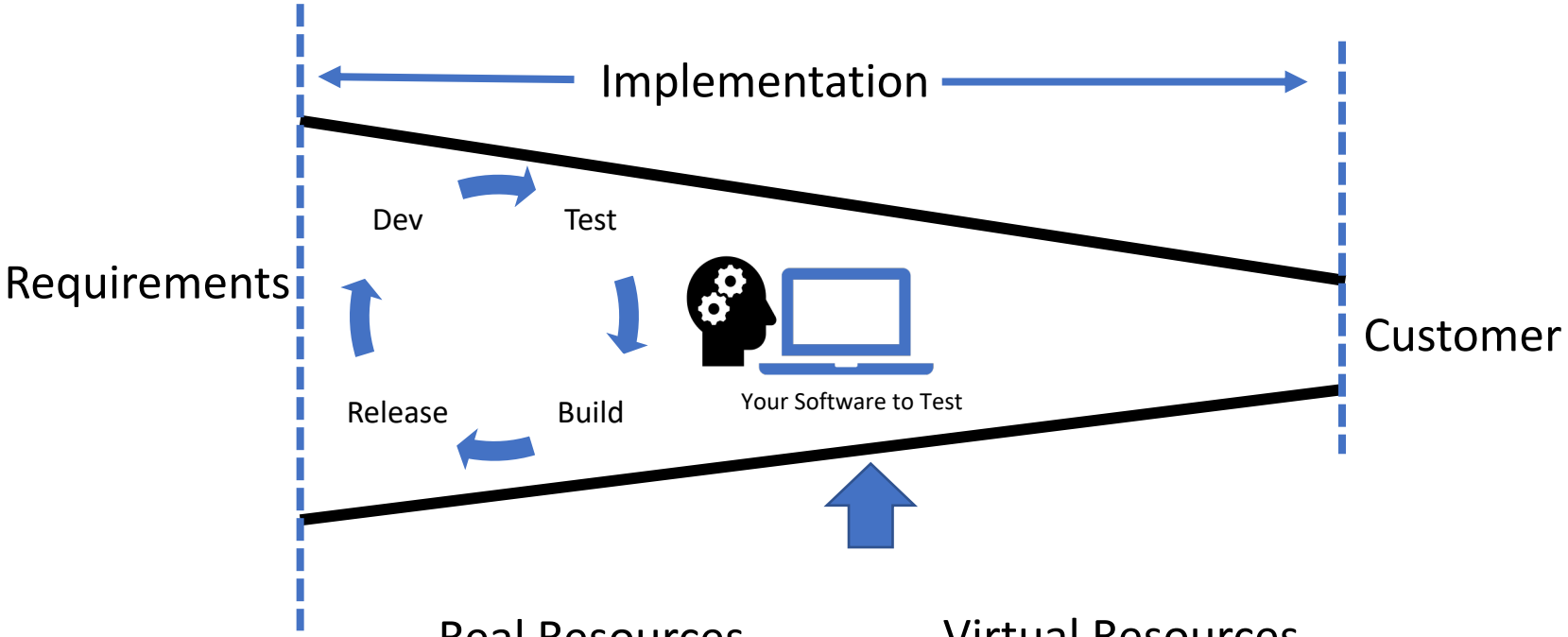


Example: Invest in automation and methodology to allow many small teams to release/test continuously.

Scaling Monolithic .vs. MicroServices Architecture, Methodology & Organizational Structure.



Testing MicroServices Techniques Using Real and Virtual Resources.



Test Strategy uses a combination of Real/Virtual Resources.

Real Resources

Software and/or other 3rd Party dependencies.

Virtual Resources

Emulates the behavior of a "Real Resource"

Virtual Resources are good to test Monolithic applications as well.

Testing MicroServices Techniques Using Real and Virtual Resources.

Real Resources


Another Software Module Test and/or Production


Production/Shipping Software


Hardware Dependencies Servers etc.


3rd Party Software (AWS)

Virtual Resources


Software Simulators


Service Virtualization


Test Doubles (Stubs and Mocking)


In-Memory Databases

Difficult to scale Real resources as system complexity increases.

Virtual resources provide more flexibility to divide and conquer test approach.

Virtual resources provide test capability not possible using Real resources.

GreenField: Start New Product with MicroServices.

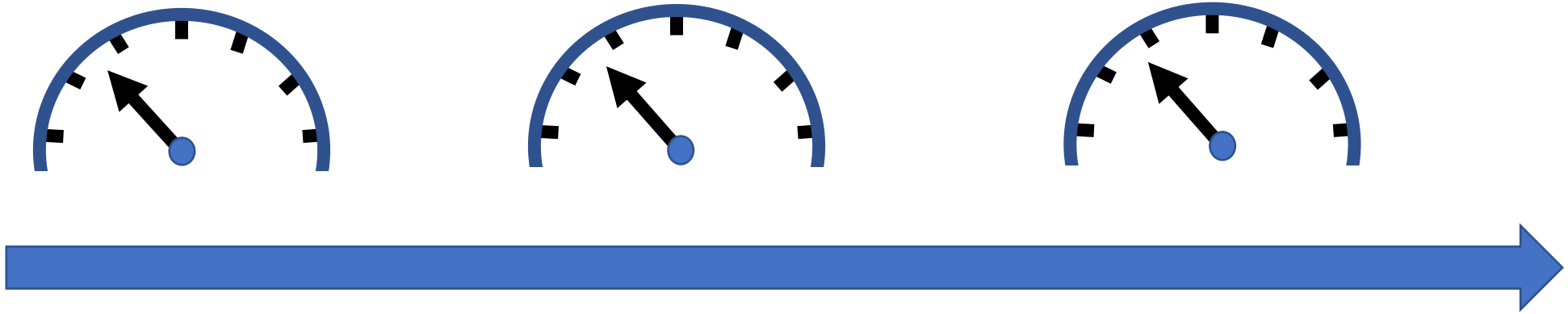
Makes sense if you are up the MicroServices curve.

If not start with a Monolithic and plan to go the MicroServices route later.

Application Complexity ?

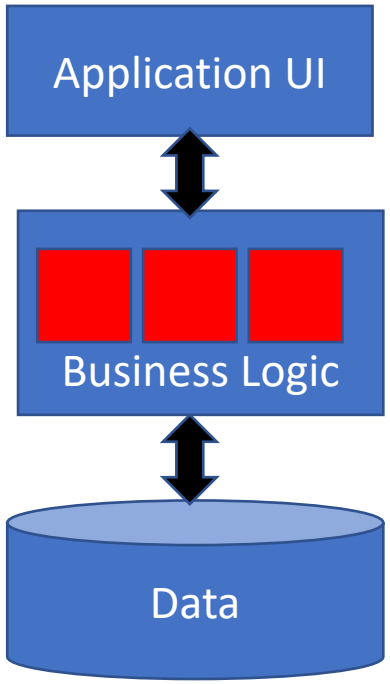
Size of Team ?

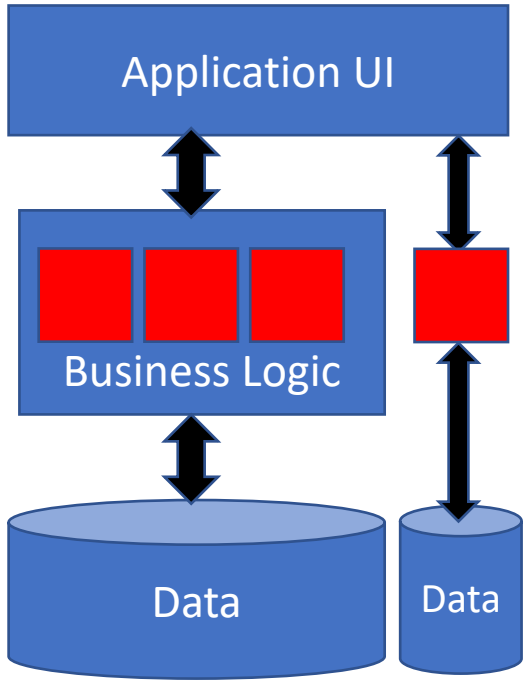
Business Requirements ?



Time

Wrap Strategy: Continue to Ship current Monolithic add new features using MicroServices. Incremental model.



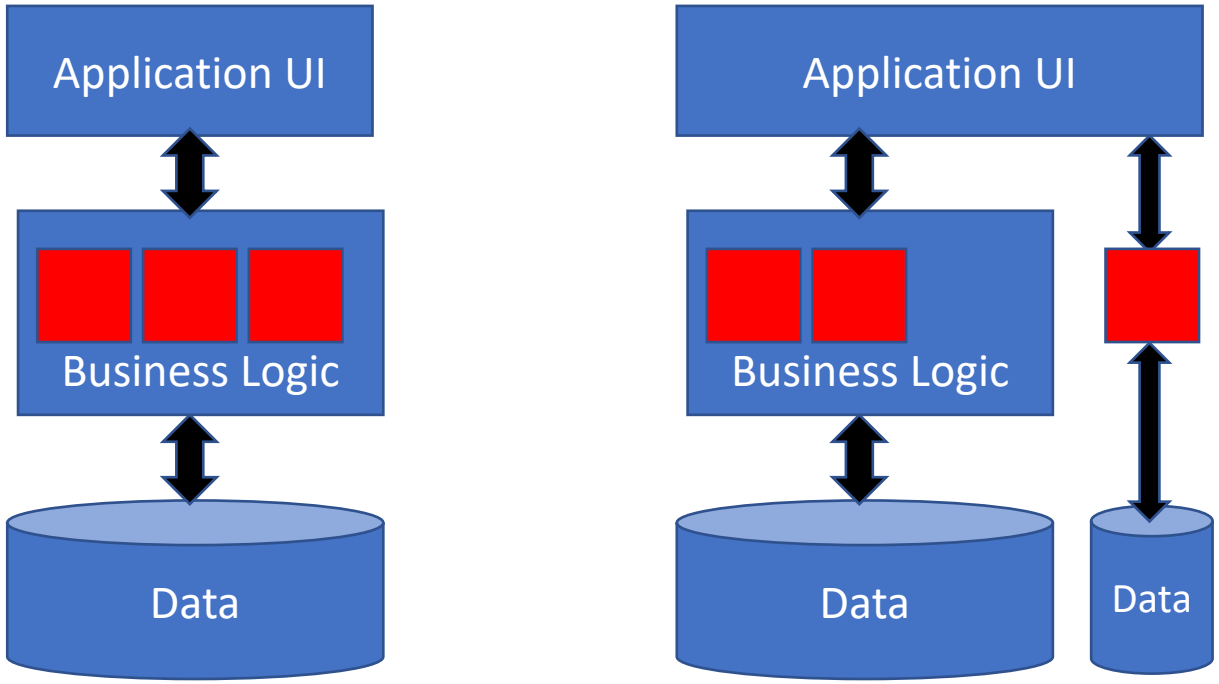


Monolithic and New MicroServices can be on different release cycles.

May be more difficult for some features.

Great way to get up-to-speed and experienced designing MicroServices.

Product Migration Strategy: Replace Monolithic Product with MicroServices or extract Monolithic components into MicroServices.



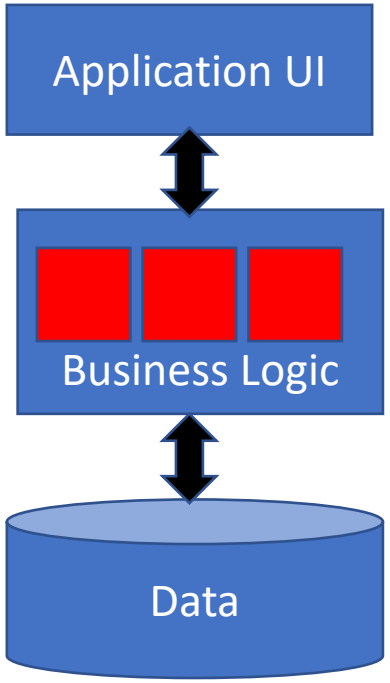
Extract that component into a separate microservice and have the deployment schedules decoupled from the monolith

The monolith could have a 3-6 month deployment cycle

MicroService can have daily releases

Allowing for faster time to market for changes to that critical component

Test Migration Strategy: Replace/augment Monolithic Test Infrastructure with Virtual Services



Virtual resources (emulate real resources) also work for Monolithic applications.

Used when real resource is not ready or available.

More difficult to isolate individual components because of centralized architecture.

Ideal for back-end or third-party services.

Great way to get started if MicroServices is in your future.

Key Points and Takeaways

- Not all applications need to be MicroService.

- Many ways to start :
  - Monolithic first then MicroService
  - Add new Features as MicroService to Monolithic
  - Replace components of a Monolithic with a MicroService

- Need to invest in people, tools and process
  - Hire expertise
  - Training
  - New tools and infrastructure
  - Structure of Organization

- Need to plan carefully and set expectations internally
  - Budget
  - ROI

Questions

Next Steps:

1. If you like what you read in Traffic Parrot articles you are welcome to contact them for a demo.

2. Briefing and evaluation on technical capability development. I have many years of experience driving and influencing change – development methodology, program management, organizational development, customer satisfaction, product quality, product risk management, team/individual professional development. It is not easy, not always quick but I have developed and fined tuned skills that have brought success to many people, many teams and organizations.

3. Change Agent Mastermind : useful for senior technical people and first and second managers who are developing new capabilities that will enable new products.