SUPERIOR MISSION SYSTEMS

Faster, Resilient, Secure & More Affordable

Dave Manley, Chief Mission Systems Architect

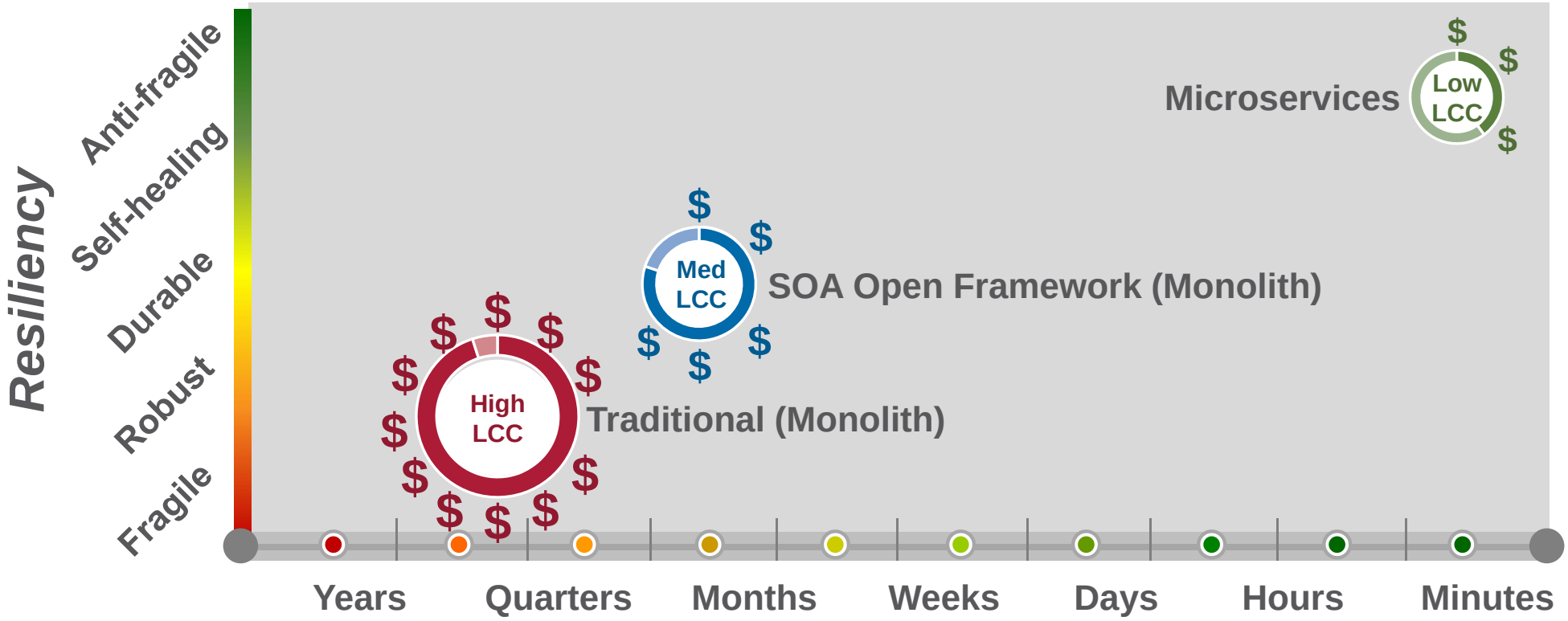
February 27, 2018

Emerging Requirements Are More Demanding

- Faster change velocities (sustainable)
- Better resiliency with higher availability
- Cybersecurity must outpace threats
- **All** within limited or reduced budgets

The challenge is monotonically increasing

The Critical Decision



The architecture decision determines a mission system's obtainable performance

Achieving Higher Availability

Downtime Allowed

Availability %	Downtime per day	Downtime per month	Downtime per year	Downtime per 3 years	Downtime per 5 years	Downtime per 10 years	Downtime per 25 years
90% ("one nine")	2.4 hours	72 hours	36.5 days	109.5 days	182.5 days	1 year	2.5 years
99% ("two nines")	14.4 minutes	7.20 hours	3.65 days	10.95 days	18.25 days	36.5 days	91.25 days
99.9% ("three nines")	1.44 minutes	43.8 minutes	8.76 hours	1.095 days	1.825 days	3.65 days	9.125 days
99.99% ("four nines")	8.64 seconds	4.38 minutes	52.56 minutes	2.628 hours	4.38 hours	8.76 hours	21.9 hours
99.999% ("five nines")	864.3 milliseconds	25.9 seconds	5.26 minutes	15.78 minutes	26.3 minutes	52.6 minutes	2.19 hours
99.9999% ("six nines")	86.4 milliseconds	2.59 seconds	31.5 seconds	1.575 minutes	2.625 minutes	5.25 minutes	13.125 minutes
99.99999% ("seven nines")	8.64 milliseconds	262.97 milliseconds	3.15 seconds	9.45 seconds	15.75 seconds	31.5 seconds	1.3125 minutes
99.999999% ("eight nines")	0.864 milliseconds	26.297 milliseconds	315.7 milliseconds	947.0 milliseconds	1.5785 seconds	3.157 seconds	7.8925 seconds
99.9999999% ("nine nines")	0.0864 milliseconds	2.6297 milliseconds	31.5569 milliseconds	94.6707 milliseconds	157.7845 milliseconds	315.569 milliseconds	788.9225 milliseconds

Always constrained by available funding, technology, and technique

The Resiliency Triad

Fragile things don't like volatility

- They easily **break** under stress
- Over time, they fail, erode, deteriorate
- e.g., wine glasses, pottery, eggs

Robust things appear immune to volatility

- They **resist** stress
- But when they fail, they do so spectacularly
- e.g., castles, Lehman Brothers, Maginot Line

Anti-fragile things enjoy volatility

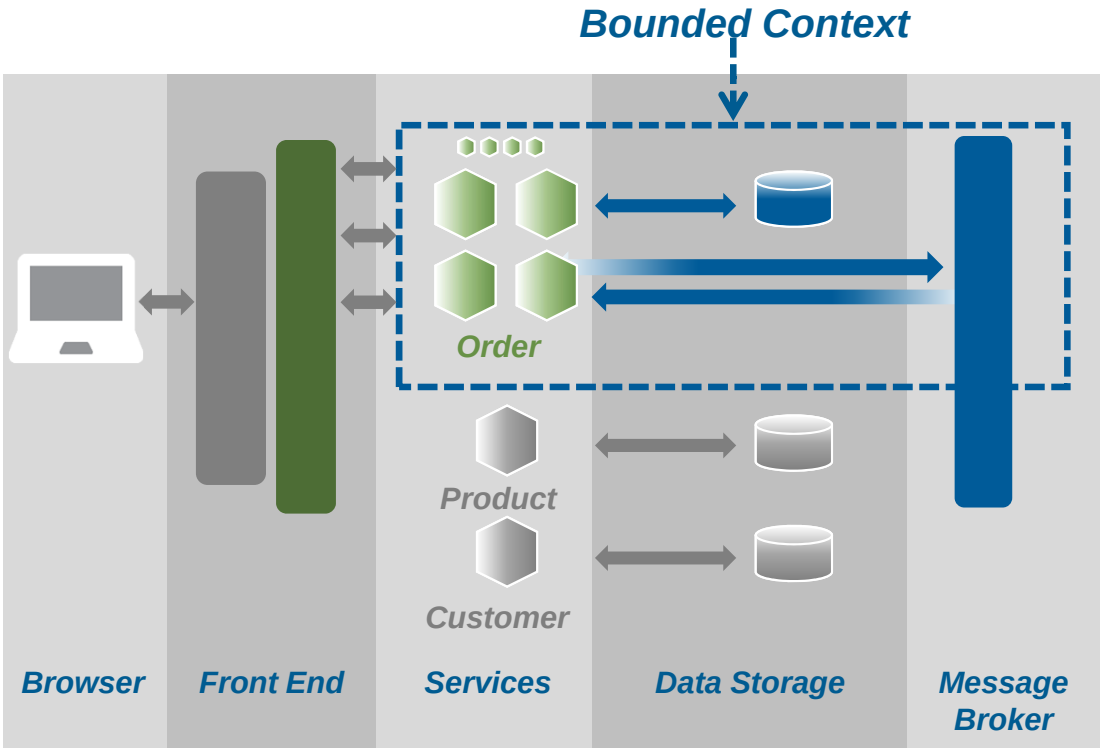
- They **benefit** from stress
- Over time, they evolve, improve, get better
- e.g., vaccines, athletes, silicon valley

The Microservices Architecture "Secret Sauce"

Modularity

Decompose application into modular set of services with **Bounded Contexts**

- Refer to Domain-Driven Design (DDD): Eric Evans (2003) & Vaughn Vernon (2013)
- Eliminate dependencies
  - Enforce implementation **guidelines** (e.g., event sourcing, aggregates, responsibility segregation, etc.)
  - Foster **independent** development, deployment, scaling, and technology stack choices
  - Pursue **simplicity** to ease learning, debugging, and enhancement



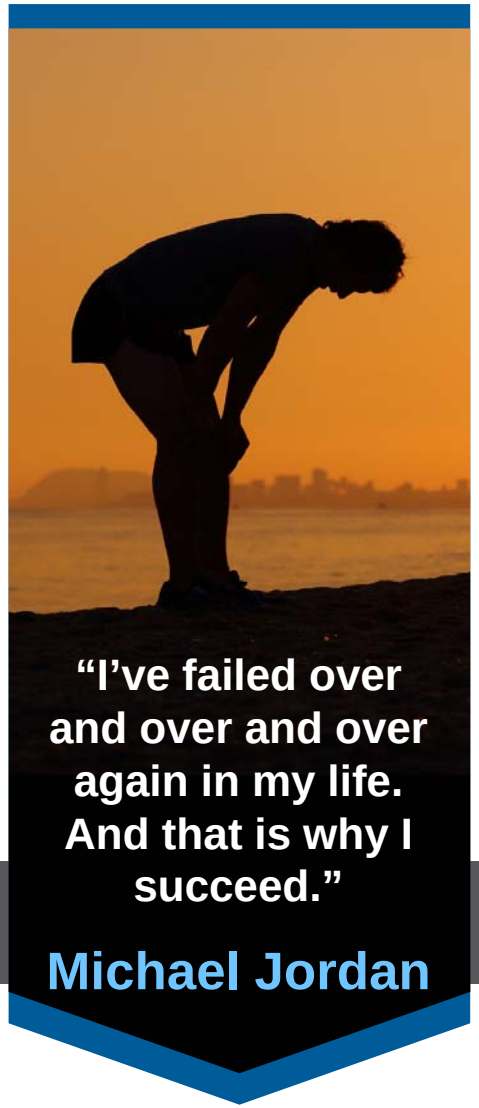
Modularity facilitates disaggregation, redundancy, and geographic separation

Additional Keys to Achieving Tougher Requirements

Disaggregation, Redundancy, Separation, Isolation, Awareness, Automation, and Testing

- Establish resiliency via disaggregation, redundancy, separation and isolation of services
- Realize continual awareness of entire stack
  - Characterize historic normal conditions, analyze current conditions, and merge results constantly
- Create robust continuous integration / continuous delivery (CI-CD) automation pipelines
- Establish and continuously improve self-healing and robust Defensive Cyberspace Operations (DCO) near real-time remediation capabilities
- Implement chaos testing and continuous security assurance

Continuously prove your implementation is anti-fragile and secure



Cyber Chaos Testing

- Assume everything will fail
- Force failure to validate resiliency
- Don’t wait for random failure, remove its uncertainty by forcing it periodically
  - Seek confidence that single failure points don’t exist
  - Seek confidence that cascading failure vulnerabilities do not exist
  - Seek confidence that self-healing automation works
  - Seek confidence that DCO capabilities are robust
- Getting stronger through failure is the basis of anti-fragility

Avoiding failure at all costs makes you brittle, vulnerable, slow, and expensive

Rapid Response Essential

Netflix Chief Architect:

*"The Chaos Monkey's job is to **randomly kill instances and services** within our architecture. If we aren't constantly testing our ability to succeed despite failure, then it isn't likely to work when it matters most – in the event of an unexpected outage."*

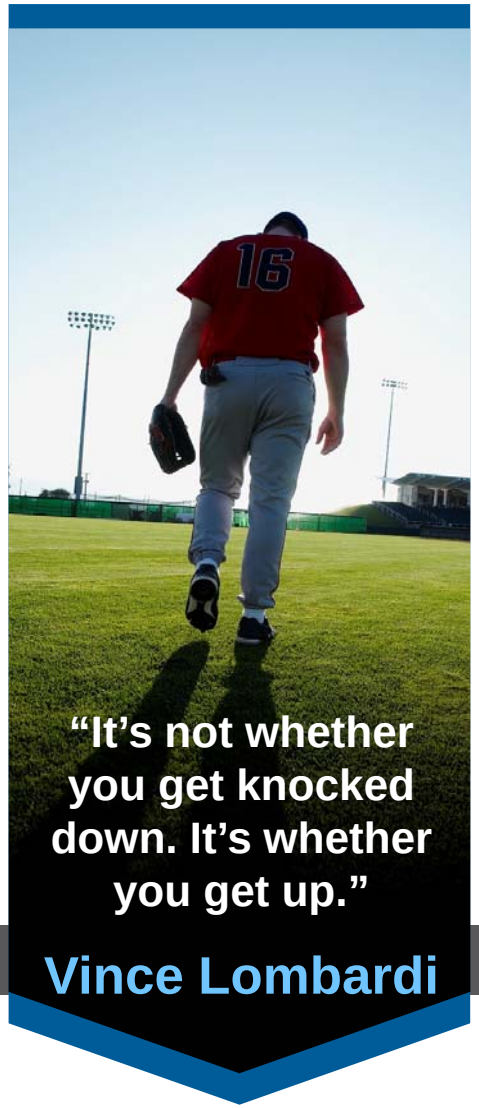
Unlike ships, with software you can automatically replace a "flooded bulkhead" in **milliseconds** and the rest of the system is none-the-wiser

If one component fails, but does not cascade, the problem can be isolated and healed/remediated while the rest of the system keeps working

Economic Benefit Sources

- **Operational efficiency** due to reductions in resources allocated to development operations
- Increased **developer productivity** resulting from automation and elimination of unexpected dependencies
- Decreased downtime due to **higher quality software that is secure by default**, which allows for better mission support and higher mission satisfaction
- Ability to support **mission growth** through shortened release cycles and faster response times to market dynamics

Significantly lower life-cycle costs are typically achieved



Summary

- Enable faster change velocity
- Improve resiliency with higher availability
- Enable cybersecurity to outpace threats
- All within limited or reduced budgets

Creates faster, resilient, secure & more affordable mission systems

ENGILITY
Engineered to Make a Difference