

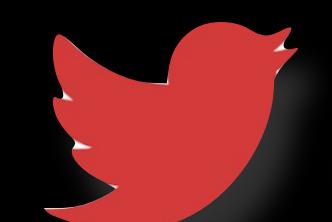


software architecture hour

ThoughtWorks®

NEAL FORD

Director / Software Architect / Meme Wrangler



@neal4d

<http://nealford.com>



software architecture hour

Distributed architectures



software
architecture
hour

Vanya Seth

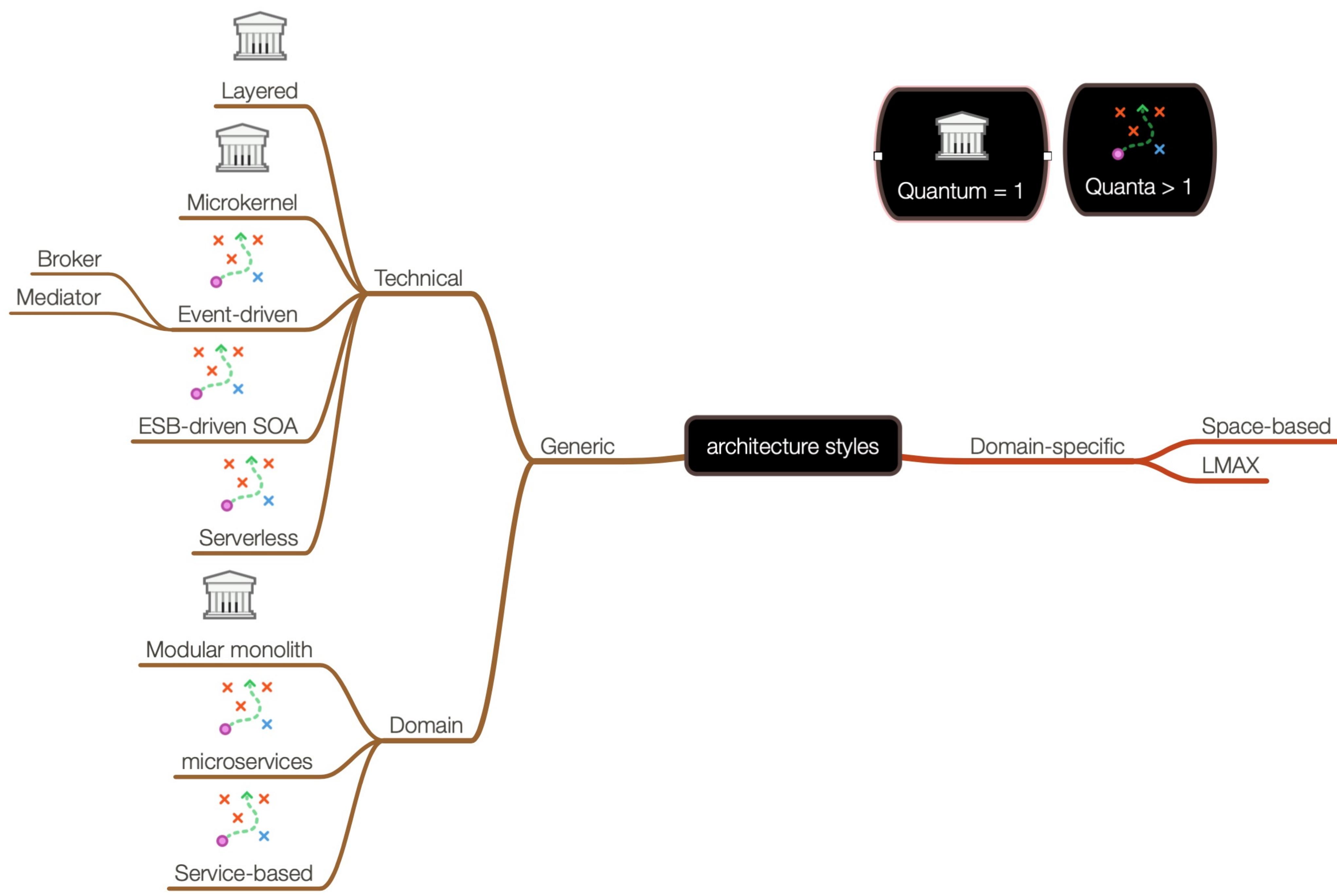




Feel free to ask us
questions in the Q&A
widget

Vanya Seth





Fallacies of Distributed Computing



The screenshot shows a web browser displaying the English Wikipedia article titled "Fallacies of distributed computing". The page is a standard Wikipedia layout with a sidebar on the left containing links to the main page, contents, and various sections like "The fallacies", "The effects of the fallacies", and "History". The main content area contains text describing the fallacies and their effects, along with a list of the eight fallacies. The URL in the browser's address bar is https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing.

The fallacies of distributed computing are a set of assertions made by L Peter Deutsch and others at Sun Microsystems describing false assumptions that programmers new to distributed applications invariably make.

Contents [hide]

- 1 The fallacies
- 2 The effects of the fallacies
- 3 History
- 4 See also
- 5 References
- 6 External links

The fallacies [edit]

The fallacies are^[1]

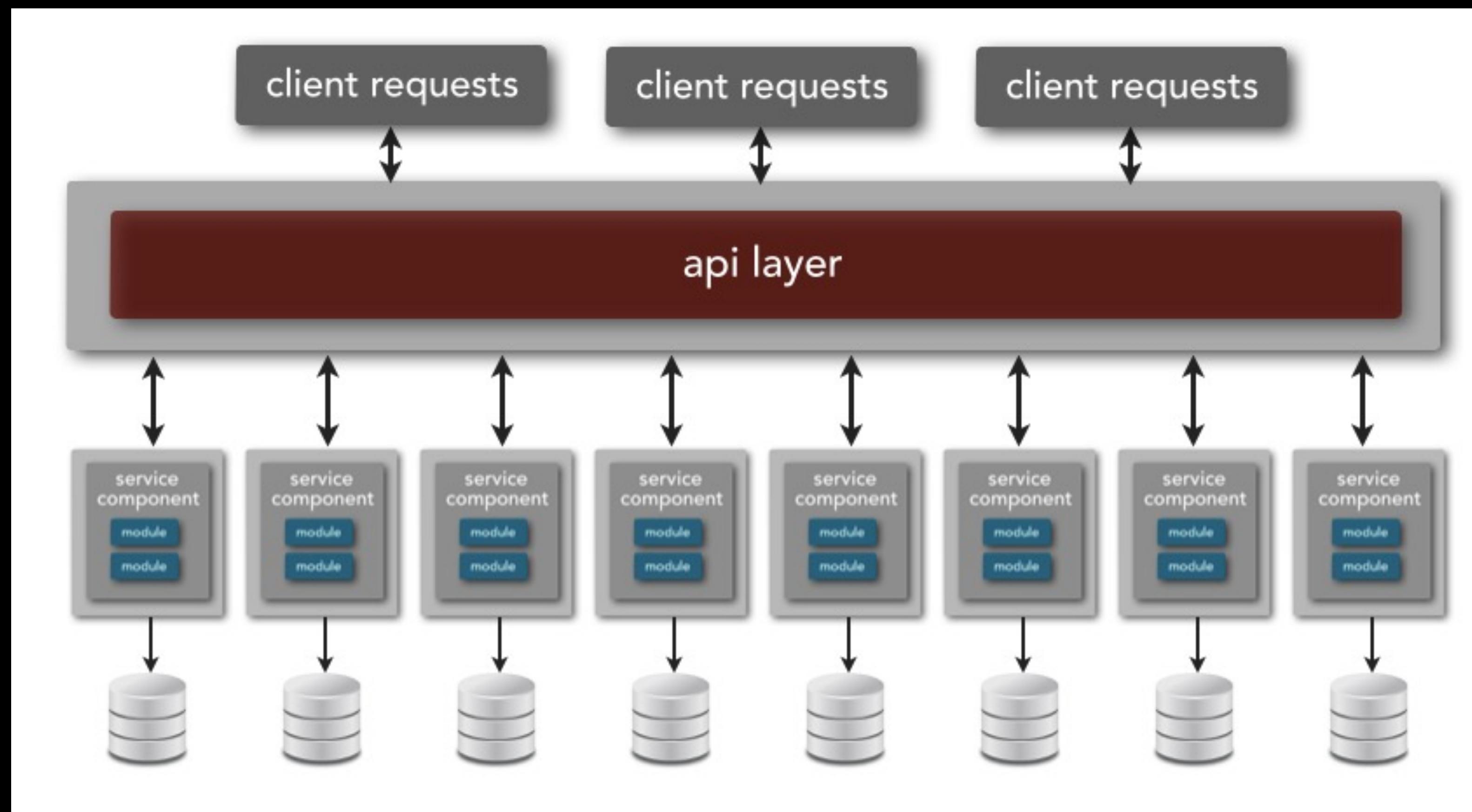
1. The network is reliable;
2. Latency is zero;
3. Bandwidth is infinite;
4. The network is secure;
5. Topology doesn't change;
6. There is one administrator;
7. Transport cost is zero;
8. The network is homogeneous.

The effects of the fallacies [edit]

- Software applications are written with little error-handling on networking errors. During a network outage, such applications may stall or infinitely wait for an answer packet, permanently consuming memory or other resources. When the failed network becomes available, those applications may also fail to retry any stalled operations or require a (manual) restart.
- Ignorance of network latency, and of the packet loss it can cause, induces application- and transport-layer developers to allow unbounded traffic, greatly increasing dropped packets and wasting bandwidth.
- Ignorance of bandwidth limits on the part of traffic senders can result in bottlenecks.
- Complacency regarding network security results in being blindsided by malicious users and programs that continually adapt to security measures.^[2]
- Changes in network topology can have effects on both bandwidth and latency issues, and therefore can have similar problems.
- Multiple administrators, as with subnets for rival companies, may institute conflicting policies of which senders of network traffic must be aware in order to complete their desired paths.
- The "hidden" costs of building and maintaining a network or subnet are non-negligible and must consequently be noted in budgets to avoid vast shortfalls.
- If a system assumes a homogeneous network, then it can lead to the same problems that result from the first three fallacies.

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

Microservices Architecture



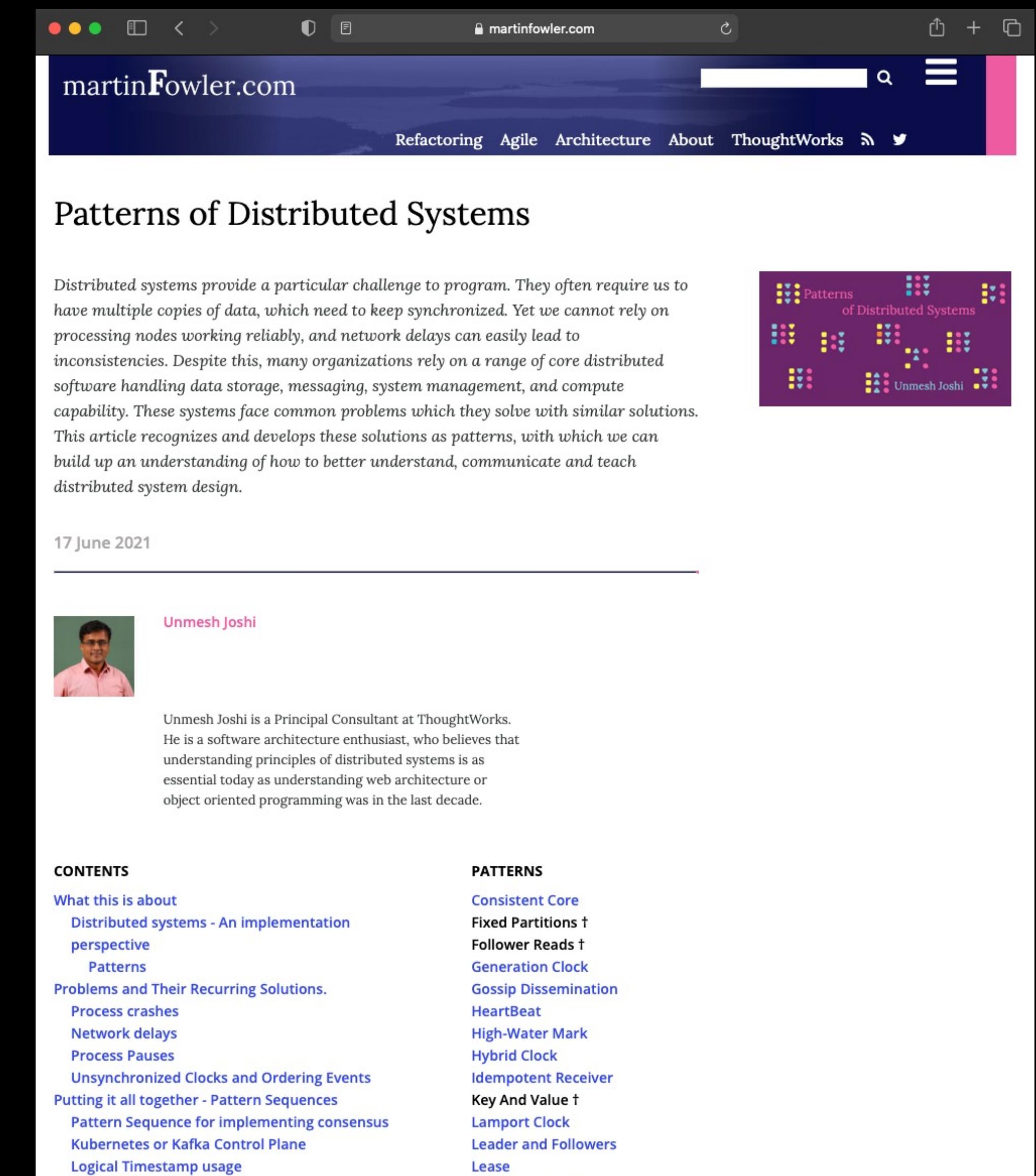


Feel free to ask us
questions in the Q&A
widget

Vanya Seth



Patterns of Distributed Systems



The screenshot shows a web browser displaying the martinFowler.com website. The page title is "Patterns of Distributed Systems". The main content area contains a paragraph about distributed systems, a author bio for Unmesh Joshi, and two columns of links: "CONTENTS" and "PATTERNS". The "CONTENTS" column includes links like "What this is about", "Distributed systems - An implementation perspective", "Patterns", "Problems and Their Recurring Solutions.", and "Putting it all together - Pattern Sequences". The "PATTERNS" column includes links like "Consistent Core", "Fixed Partitions", "Follower Reads", "Generation Clock", "Gossip Dissemination", "HeartBeat", "High-Water Mark", "Hybrid Clock", "Idempotent Receiver", "Key And Value", "Lamport Clock", "Leader and Followers", and "Lease". The right sidebar features a purple decorative graphic with the text "Patterns of Distributed Systems" and "Unmesh Joshi".

martinFowler.com

Refactoring Agile Architecture About ThoughtWorks

Patterns of Distributed Systems

Distributed systems provide a particular challenge to program. They often require us to have multiple copies of data, which need to keep synchronized. Yet we cannot rely on processing nodes working reliably, and network delays can easily lead to inconsistencies. Despite this, many organizations rely on a range of core distributed software handling data storage, messaging, system management, and compute capability. These systems face common problems which they solve with similar solutions. This article recognizes and develops these solutions as patterns, with which we can build up an understanding of how to better understand, communicate and teach distributed system design.

17 June 2021

 **Unmesh Joshi**

Unmesh Joshi is a Principal Consultant at ThoughtWorks. He is a software architecture enthusiast, who believes that understanding principles of distributed systems is as essential today as understanding web architecture or object oriented programming was in the last decade.

CONTENTS

- [What this is about](#)
- [Distributed systems - An implementation perspective](#)
- [Patterns](#)
- [Problems and Their Recurring Solutions.](#)
- [Putting it all together - Pattern Sequences](#)
- [Pattern Sequence for implementing consensus](#)
- [Kubernetes or Kafka Control Plane](#)
- [Logical Timestamp usage](#)

PATTERNS

- [Consistent Core](#)
- [Fixed Partitions](#)
- [Follower Reads](#)
- [Generation Clock](#)
- [Gossip Dissemination](#)
- [HeartBeat](#)
- [High-Water Mark](#)
- [Hybrid Clock](#)
- [Idempotent Receiver](#)
- [Key And Value](#)
- [Lamport Clock](#)
- [Leader and Followers](#)
- [Lease](#)

<https://martinfowler.com/articles/patterns-of-distributed-systems/>

Mark Richard's Software Architecture Monday

<https://developertoarchitect.com/lessons/>



Software Architecture Monday

Software Architecture Monday with Mark Richards is a free bi-weekly software architecture lesson containing a short video about some aspect of software architecture. These lessons contain tips, techniques, and advice to help you in your journey from developer to architect. New lessons will be posted every other Monday.

[All Lessons](#)

[Microservices Lessons](#)

[General Architecture Lessons](#)

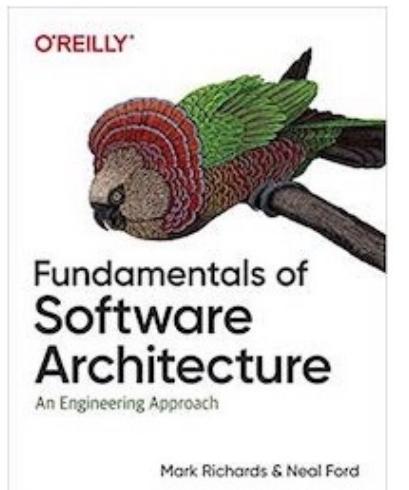
[Event-Driven Architecture Lessons](#)

[Soft Skills Lessons](#)

[Integration Architecture Lessons](#)

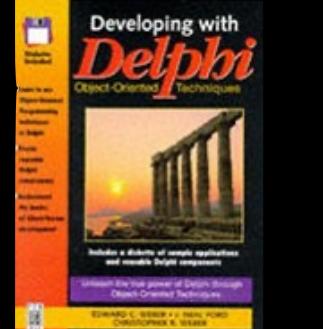
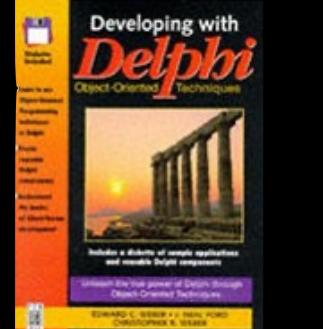
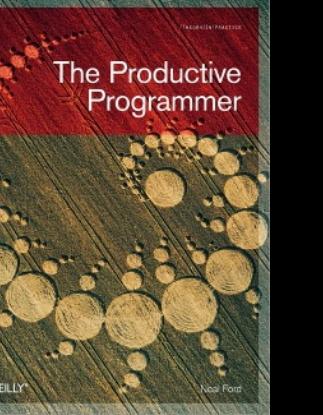
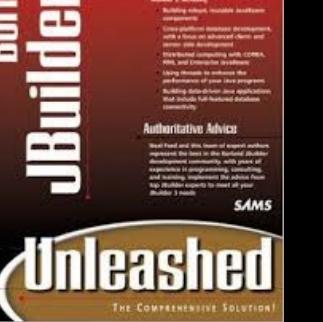
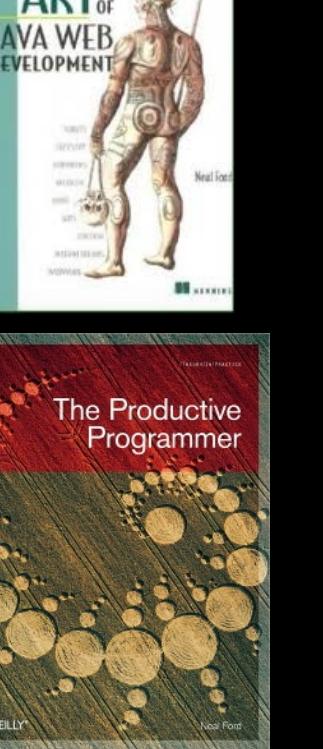
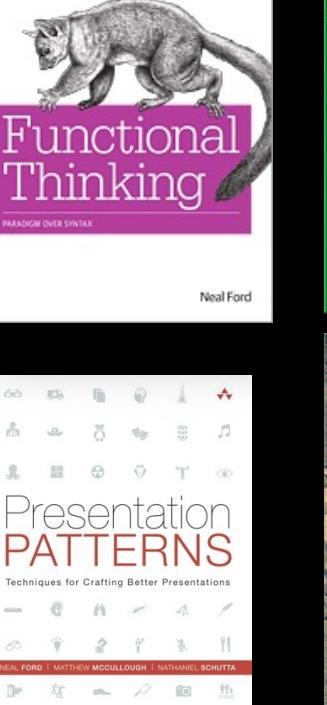
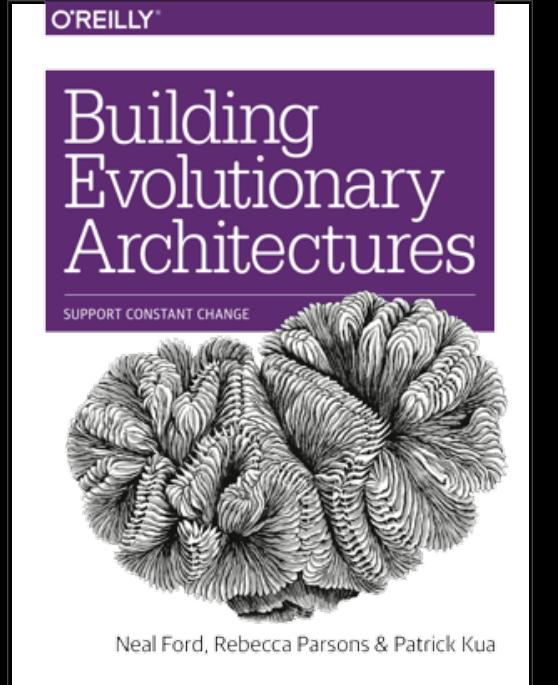
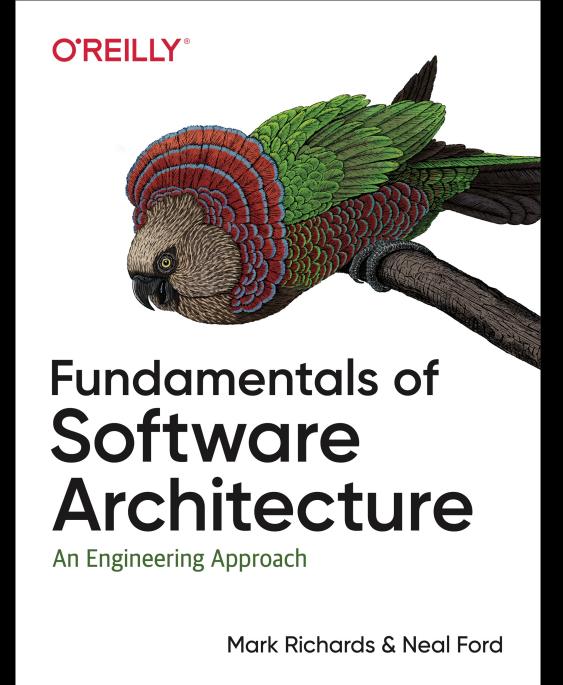
[Enterprise Architecture Lessons](#)

- [Lesson 116 - Negotiation Tips for Software Architects \(posted June 21, 2021\)](#)
- [Lesson 115 - Email-Driven Architecture \(posted June 7, 2021\)](#)
- [Lesson 114 - Microservices vs Service-Based Architecture \(posted May 24, 2021\)](#)
- [Lesson 113 - Cart Before The Horse Anti-Pattern \(posted May 10, 2021\)](#)
- [Lesson 112 - Architecture Characteristics Worksheet \(posted April 26, 2021\)](#)
- [Lesson 111 - CAP Theorem Illustrated \(posted April 12, 2021\)](#)



Fundamentals of Software Architecture

Be sure and check out my latest book with Neal Ford [Fundamentals of Software Architecture](#) (O'Reilly).



ThoughtWorks®

Clients Services Products Insights About us Careers

PODCASTS

Tech Talks that matter

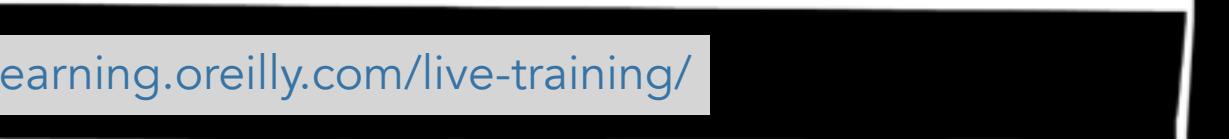
The ThoughtWorks podcast plunges deep into the latest tech topics that have captured our imagination. Join our panel of senior technologists to explore the most important trends in tech today, get frontline insights into our work developing cutting-edge tech and hear more about how today's tech megatrends will impact you.

www.thoughtworks.com/podcasts

Building Evolutionary Architectures

Home Resources Talks Fitness Function Katas The Book

Building Evolutionary Architectures: Support Constant Change [Learn More](#)



O'REILLY®

Live Online Training

Real-time. Real experts. Real learning.

O'REILLY®

Learning Path Software Architecture Fundamentals—Evolutionary Architecture

O'REILLY®

Learning Path Software Architecture Fundamentals—Architecture Styles

O'REILLY®

Learning Path Software Architecture Fundamentals—Diagramming and Documenting Architecture

O'REILLY®

Learning Path Software Architecture Fundamentals—Architectural Thinking

O'REILLY®

Learning Path Software Architecture Fundamentals—Soft Skills

O'REILLY®

Learning Path Software Architecture Fundamentals—Architecture Techniques

O'REILLY®

Software Architecture Fundamentals—Architecture Techniques



fundamentalsofsoftwarearchitecture.com

About Architectural Katas Fundamentals of Software Architecture List of Architecture Katas

Fundamentals of Software Architecture

Mark Richards & Neal Ford

Fundamentals of Software Architecture

An Engineering Approach

Mark Richards & Neal Ford

About the Book

Salary surveys worldwide regularly place software architect in the top 10 best jobs, yet no real guide exists to help developers become architects. Until now. This book provides the first comprehensive overview of software architect's many responsibilities, including architecture design, component determination, and many other topics. It software architecture classes professionally for years—focus on architecture principles that apply across all technology innovations of the past decade.

This book examines:

fundamentalsofsoftwarearchitecture.com

Incremental

Fitness Functions

Multiple Dimensions

Evolutionary architectures are built one part at a time, with many different increments. Speed to the next increment is key.

Every system at different points of their life need to optimise to be "fit" for its environment. Evolutionary architectures make it explicit what "fit" means with as much automation as possible.

ThoughtWorks®

NEAL FORD

Director / Software Architect / Meme Wrangler



<http://nealford.com>