Thinking Architecturally

Nathaniel T. Schutta
@ntschutta
ntschutta.io

Architecting is hard…

Many competing agendas.

Technology changes.

Constantly.

Feature not a bug.

Keeps things interesting…

We want to avoid legacy platforms.

But we can’t change
things every few months.

“Our app has 4 different UI frameworks…”

Developers kept chasing the new hotness.

How do we avoid that?

How do we evaluate new technology?

I have no idea what language/ framework/platform is “next”.

No one does.

But I can guarantee you this much:

It will be different than
what we use today.

Five years from now we will be using something that isn’t invented yet.

Chasing the new thing.

Technology changes.

Constantly.

Tempting to always chase the “new hotness”.

Bleeding edge.

It’s fun!

Part of being in this industry.

Our understanding constantly evolves.

Twitter, Inc.

Thread

Benedict Evans
@BenedictEvans

There are things you understand well enough to have strong opinions on, things you know just well enough to know what others' opinions are, and things where you can have no opinion. The complication in tech is that things move between these categories all the time.

1:42 PM · Dec 30, 2017

70 Retweets **218** Likes

Benedict Evans @BenedictEvans · Dec 30
Replying to @BenedictEvans
In tech, there are:
Things you're starting to learn about
Things you know well
Things you're no longer following
Things you were an expert on but moved on from years ago and half-forgot

In tech, the point that you really understand something is the point it's becoming irrelevant

5 79 193

Benedict Evans @BenedictEvans · Dec 30
For example, I used to know all about radio networks, mobile networks and the Japanese mobile market iMode! IAppli! PHS!). Now, none of that matters. Most questions about smartphones are now moving rapidly down the list as well. On to new questions.

13 2 50

@FemiAbodunde · Dec 30
Replying to @BenedictEvans
🔥🔥🔥🔥The speed of the feed.

Let’s be honest…

Developers have opinions!

Often ⁺very⁺ strong opinions.

Maybe we fear old things?

Twitter, Inc.

Thread

Jeff Johnson (Mac/iOS developer for hire) @lapcats... · Dec 21

It's said that "senior" developers tend to be afraid of learning new things. This has always struck me as absurd, because who hasn't learned a ton of new things over the course of 10+ years of programming? Almost impossible not to.

4 5 18

Jeff Johnson (Mac/iOS developer for hire) @lapcats... · Dec 21

It almost feels as though less experienced developers are projecting their own fears onto others with more experience.

1 1 8

Jeff Johnson (Mac/iOS developer for hire) @lapcats... · Dec 21

I get the sense that many programmers are afraid to learn *old* things. As if the older things are impossibly arcane and complex, compared to the supposed simplicity and elegance of new tech.

1 6 18

Jeff Johnson (Mac/iOS developer for hire) @lapcats... · Dec 21

They have a strong preference for the new. They'd rather use unfinished, buggy, unproven technology than older, refined, stable, tested tech. Even if it reinvents or indeed breaks the wheel.

2 7 15

Jeff Johnson (Mac/iOS developer for hire) @lapcats... · Dec 21

So now there's a of excitement about a "UnicornKit" that combined UIKit and AppKit into one. But do people realize that unless UnicornKit is simply UIKit, you're going to have to learn something else anyway?

1 3 9

Jeff Johnson (Mac/iOS developer for hire) @lapcats... · Dec 21

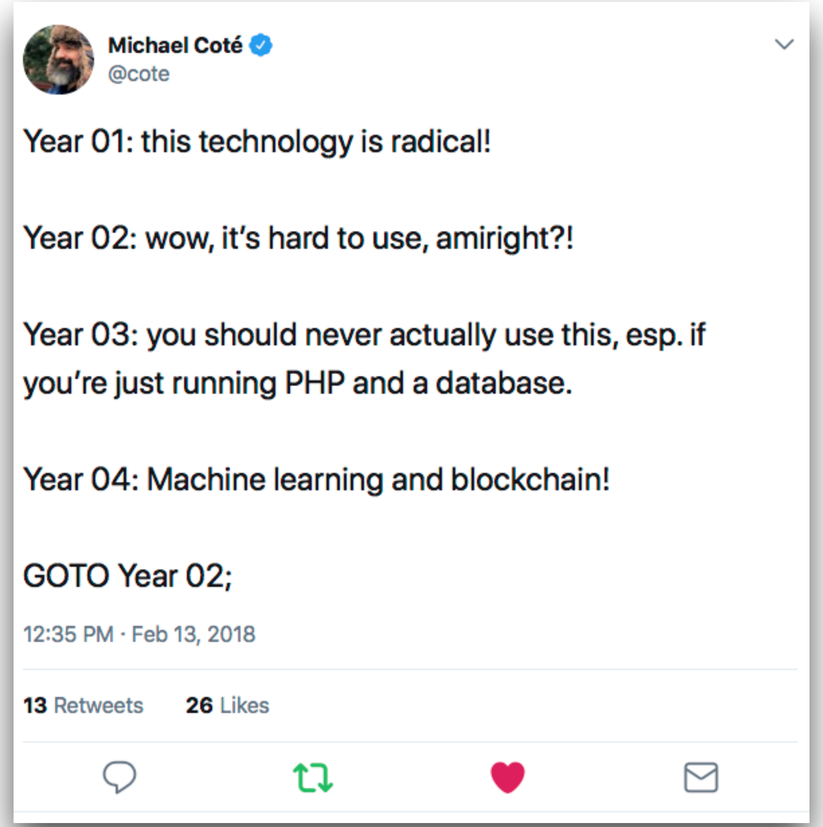
Why is learning UnicornKit harder than learning AppKit? Many developers have successfully written both Mac and iOS apps. It's not as hard as you may think if you have no experience with AppKit.

2 3 8

Jeff Johnson (Mac/iOS developer for hire) @lapcats... · Dec 21

Also, any new technology is going to be chock full of bugs. AppKit has its bugs, yes, but it's assuredly vastly more reliable and stable than UnicornKit

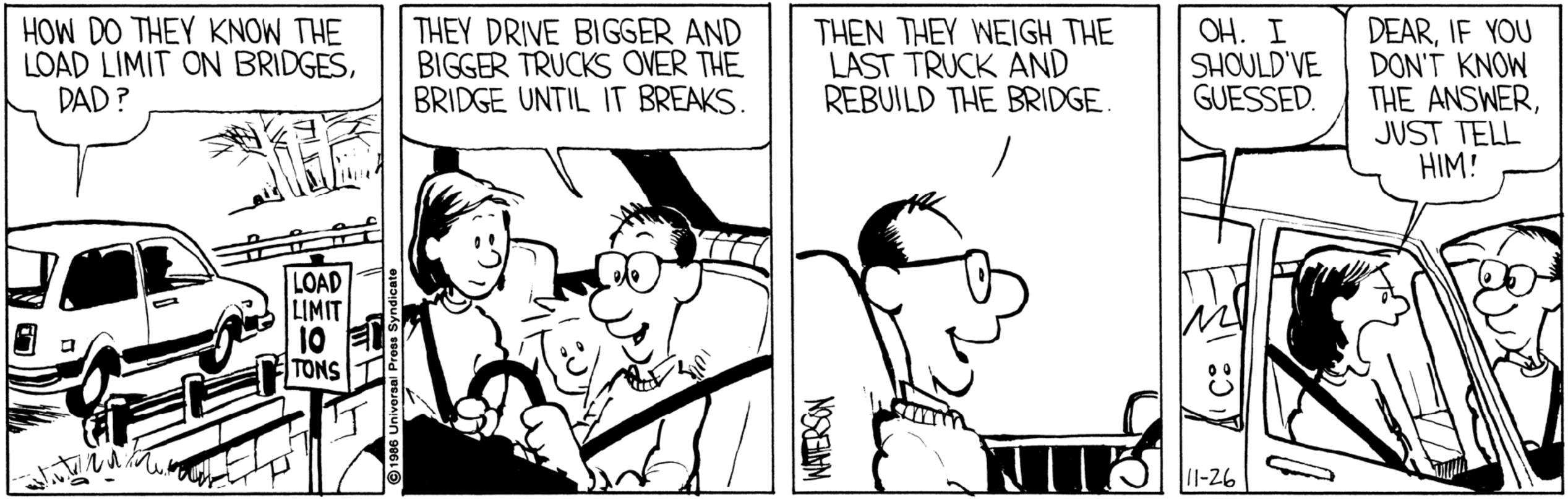
Predictable hype cycle.



https://mobile.twitter.com/cote/status/963481741171265537

How do we know where **not** to use a technology?

Trial and error.



CALVIN AND HOBBES © 1986 Watterson. Reprinted with permission of ANDREWS MCMEEL SYNDICATION. All rights reserved.

Developers tend to get bored quickly.

Learning keeps it fresh.

But we have to deliver business value.

Can’t do that if we’re always experimenting.

Have to commit at some point.

Develop some expertise.

Bleeding edge…means you will bleed!

Joe Armstrong
@joeerl

Dear <redacted>

Thank you for the latest version of your wonderful program with many new features.

Breaking my old code that worked in your previous version is a great idea and will provide me with many hours of fun searching for a work around.

Tearing out my hair
@joeerl

6:28 AM · Nov 15, 2017

73 Retweets **221** Likes

https://mobile.twitter.com/joeerl/status/930774515512201216

Pioneers…the ones with arrows in their backs.

What is your strategy?

How do we avoid dead platforms?

Without constantly changing direction?

Strategy.

Hope is not a strategy!

But it is what rebellions are built on.

We need to be deliberate.

There are a lot of bits out there…

New languages,
techniques, approaches.

How do you keep up?

Blogs? Books? Twitter?
Podcasts? Conferences?

Develop a routine.

Block out Friday afternoon.
Tuesday over lunch. Whatever fits.

Consider “morning coffee”.

Take 15-30 minutes in the morning to peruse the tech news.

Before the day gets away from you…

Attention is precious.

“Attention is a bit like real estate, in that they're not making any more of it. Unlike real estate, though, it keeps going up in value.”

— Seth Godin

http://sethgodin.typepad.com/seths_blog/2011/07/paying-attention-to-the-attention-economy.html

Don’t waste it.

Be selective.

Can’t read it all.

In fact, you'll miss almost everything.

http://www.npr.org/blogs/monkeysee/2011/04/21/135508305/the-sad-beautiful-fact-that-were-all-going-to-miss-almost-everything

We cannot adopt every new thing.

How do we know where to invest our time?

Hacker's Radar?

http://www.paulgraham.com/javacover.html

“I have a hunch that [Java] won't be a very successful language.”

Never written a line of Java,
glanced at some books.

Need more than just a hunch.

“Judging Covers” can be a useful filter.

But beware bias.

Where is the community?

Are you skating to
where the puck *was* ?

Technology Radar.

https://www.thoughtworks.com/radar

thoughtworks.com

ESPAÑOL PORTUGUÊS DEUTSCH 中文 Search thoughtworks.com NEWS CONTACT US

ThoughtWorks

Clients Services Products Insights Events About us Careers

TECHNOLOGY RADAR *VOL.17*

Insights into the technology and trends shaping the future

Select an area to explore

Search About the Radar Build your Radar

Downloads (EN, CA, ES, IT, PT, TR, 中文)

Subscribe to Technology Radar

THEMES FOR THIS EDITION

Open Source on the Rise in China

Because of changes in both attitude and policy, many large Chinese companies such as Alibaba and Baidu are rapidly releasing open source

Kubernetes, the Choice for Container Orchestrator

The software development ecosystem is settling on Kubernetes and related

Cloud as the New Normal

As cloud providers become more capable and reach parity of features, the public cloud model is becoming the new default for many organizations

Trust in Blockchain More Evenly Distributed

Many of our clients are finding ways to leverage blockchain solutions for distributed ledgers and

Remember Google's 20% time?

Fallen out of favor in some circles…

Innovation Fridays.

Could you carve out Friday afternoons?

How about Tuesday Tech Talks?

Architectural Briefings.

https://github.com/stuarthalloway/presentations/wiki/Architectural-Briefings

One person does some research, presents to the team.

And no, you don’t need to be an architect to present!

Why should we use X?

What do you need to know to answer the “why”?

What do you need to know in order to use X?

Keep it short - 45 minutes.

Not a how to.

Beyond the initial documentation.

These are participatory events!

Attendees should be taking notes.

Asking questions.

Using their own experiences.

Do you agree? Why or why not?

By the way, you are up next week…

Pass the briefing filter?

Hands on time.

Workshop it.

Couple of hours.

A few exercises.

Focus on how to, simple setup.

Pass the hands on filter?

Time to trial it in the organization.

Real project work that is a good fit.

Probably not a “bet the company” project though!

The new hotness is not our only concern though.

Need to stay current on the things we are using day in day out.

heartbleed.com

The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

What leaks in practice?

We have tested some of our own services from attacker's perspective. We attacked ourselves from outside, without leaving a trace. Without using any privileged information or credentials we were able steal from ourselves the secret keys used for our X.509 certificates, user names and passwords, instant messages, emails and business critical documents and communication.

How to stop the leak?

As long as the vulnerable version of OpenSSL is in use it can be abused. Fixed OpenSSL has been released and now it has to be deployed. Operating system vendors and distribution, appliance vendors, independent software vendors have to adopt the fix and notify their users. Service providers and users have to install the fix as it becomes available for the operating systems, networked appliances and software they use.

BUSINESS
SEP 14 2017, 3:21 PM ET

Equifax Hackers Exploited Months-Old Flaw

by BEN POPKEN

SHARE
Share
Tweet
Email
Print

Equifax announced late Wednesday that the source of the hole in its defenses that enabled hackers to plunder its databases was a massive server bug first revealed in March.

For the rest of the IT world, fixing that flaw was a "hair on fire moment," a security expert said, as companies raced to install patches and secure their servers. But at Equifax, criminals were able to pilfer data from mid-May to July, when the credit bureau says it finally stopped the intrusion.

Equifax, Software Company Blame Each Other for Security Breach 1:52

"We know that criminals exploited a U.S. website application vulnerability," Equifax said in an update on its website Wednesday night. "The vulnerability was Apache Struts CVE-2017-5638." Equifax said it was working with a leading cybersecurity firm, reported to be Mandiant, to investigate the breach. Mandiant declined an NBC News request for comment.

Related: The One Move to Make After Equifax Breach

The Apache Software Foundation, which oversees the Apache Struts project, said in a press release Thursday that a software update to patch the flaw was

advertisement

FROM THE WEB
Sponsored Links

Find Out In One Minute If You Pre-Qualify For A Citi Card
Citi

Why These 10 SUVs are the Cream of the Crop
Kelley Blue Book

by Taboola

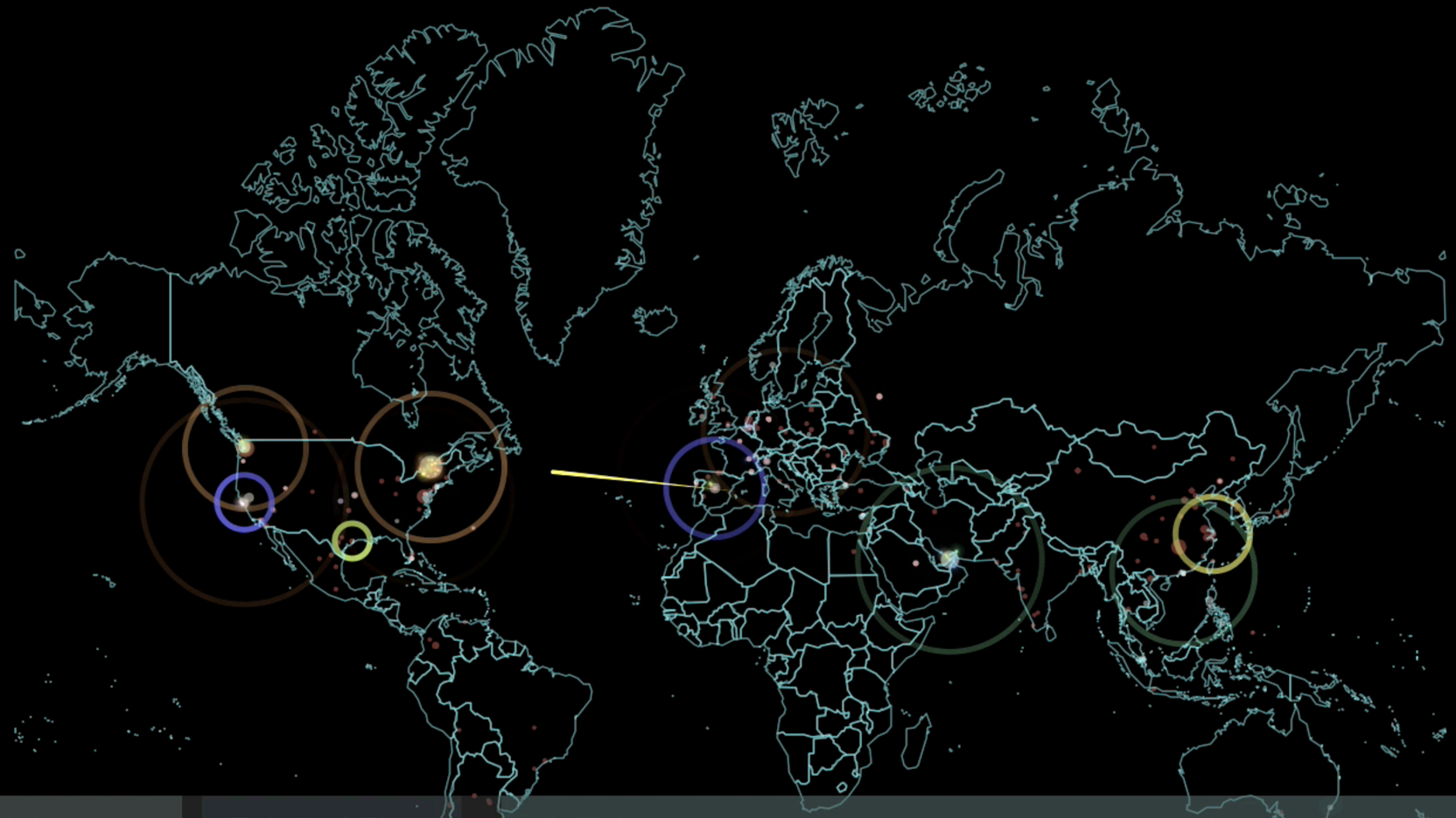
MORE FROM NBC NEWS

Oops.

Don’t think you’re a target?

map.norsecorp.com

NORSE



ATTACK ORIGINS

#	COUNTRY
525	United States
381	China
34	Ukraine
31	Netherlands
22	South Korea
21	Colombia
18	Switzerland
14	France
13	Vietnam
13	Turkey

ATTACK TYPES

#	PORT	SERVICE TYPE
418	25	smtp
205	23	telnet
192	8080	http-alt
59	3389	ms-wbt-server
41	5900	rfb
35	445	microsoft-ds
31	50864	xsan-filesystem
25	3306	mysql
21	53413	netis-router
18	123	ntp

ATTACK TARGETS

#	COUNTRY
746	United States
278	United Arab Emirates
62	Spain
26	Singapore
25	Italy
17	Philippines
16	France
11	Norway
10	Russia
9	Belgium

LIVE ATTACKS

TIMESTAMP	ATTACKER	ATTACKER IP	ATTACKER GEO	TARGET GEO	ATTACK TYPE	PORT
17:39:01.311	Chinanet Shanghai Province Network	180.168.28.12	Shanghai, CN	Montreal, CA	ctdb	4379
17:39:01.061	Telefonica De Espana	83.52.142.84	Madrid, ES	San Francisco, US	x11	6000
17:39:00.765	Microsoft Corporation	157.56.111.251	Redmond, US	De Kalb Junctio...	smtp	25
17:39:00.414	An Internet Service Provider In Hong Kong.	203.132.205.97	Kowloon, HK	Dubai, AE	ms-wbt-server	3389
17:38:59.932	Xdsl Network-Adsl	90.178.246.150	Louny, CZ	San Francisco, US	telnet	23
17:38:59.485	Microsoft Corporation	65.55.169.248	Washington, US	De Kalb Junctio...	smtp	25
17:38:59.346	Retecal Sociedad Operadora De Telecomunicacio...	81.172.32.177	Leon, ES	De Kalb Junctio...	telnet	23
17:38:58.928	Chinanet Guangxi Province Network	171.107.91.55	Nanning, CN	Madrid, ES	telnet	23
17:38:58.712	This Ip Network Is Used For Internet Security Res...	185.35.62.37	Geneve, CH	Madrid, ES	ntp	123

HOME

EXPLORE

WHY NORSE?

Justin Smith Follow
Identity and Security Geek
Apr 19, 2016 · 8 min read

The Three Rs of Enterprise Security: Rotate, Repave, and Repair

At high velocity, the three Rs starve attacks of the resources they need to grow. It's a complete 180-degree change from the traditional careful aversion to change to mitigate risk. Go fast to stay safer—in other words, speed reduces risk.

—Justin Smith

What is your patching strategy?

What version of X are you on?

Some organizations have a policy of N or N-1.

Do they measure it? Do they enforce it?

What needs to change in your culture to stay at N?

What hurts more? Changing your patching strategy?

Or being on the receiving end of the latest “largest hack ever”?

Pros and Cons.

Every technical choice involves tradeoffs.

When we find ourselves presented with technology that promises to offer us drastic improvements, we need to look for the trade-offs.

— Susan J. Fowler
Production-Ready Microservices

Essence of design.

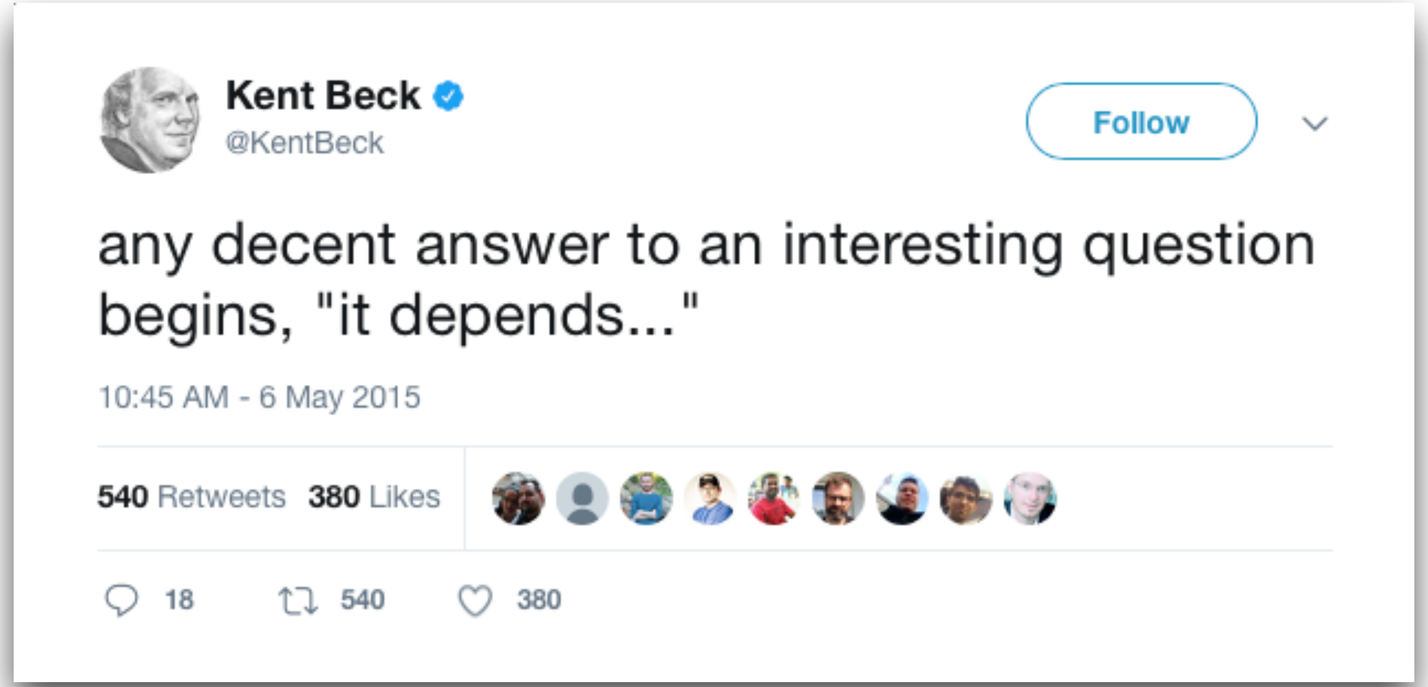
To paraphrase Harry Truman…

Give me a one handed technologist.

Should we use React or Angular?

Should we refactor to microservices?

Should we be on prem
or public cloud?



https://twitter.com/KentBeck/status/596007846887628801

In many cases?

卒卒 ! !!

Balancing those opposing forces is the art of architecture.

No tech is perfect, don’t pretend it is.

Acknowledge the negatives.

What do you like about it?

What don’t you like about it?

What would you add?

What would you remove?

King of Java for a day…

https://mobile.twitter.com/kelseyhightower/status/963428093292457984

How does it stack up to alternatives?

The spreadsheet approach.

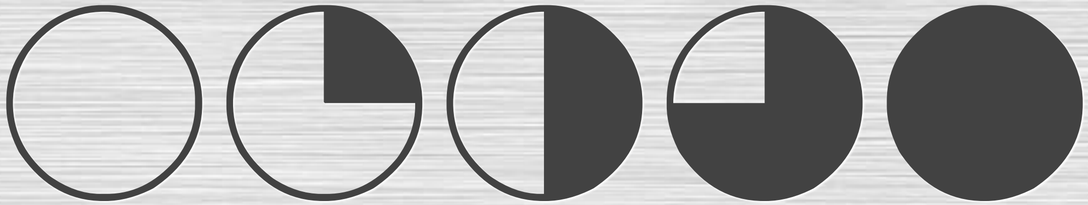
Options across the top.

Criteria down the left.

Criteria can be weighted.

Harvey balls.

http://en.wikipedia.org/wiki/Harvey_Balls



How closely does does it map to the criteria?

Very effective...

	Angular	React
Documentation	●	◕
Community	●	◕
Committer diversity	●	●
Codebase	●	●
Testability	●	●
Update history	◔	◔
Maturity	●	◕

	Angular	React
Stability	●	◕
Extensibility	●	●
Support	●	●
Training	●	●
Hiring	●	◕
Corporate fit	?	?
Usage	●	●

What criteria should you use?

How should they be weighted?

Up to you.

You can tip the scales…

Usually backfires.

Quality Attributes.

Sometimes called non functional requirements.

Or quality goals, constraints, quality of service goals…

Cross-functional requirements.

Architecturally significant requirements.

The “ilities”!

Customers usually focus on functionality.

It’s what they “see” after all.

Obviously important we meet their needs!

But we have to look beyond that.

Vital that we focus on quality attributes.

Service level objectives if you will.

What are some quality attributes?

Maintainability.

Scalability.

Reliability.

Security.

Deployability.

Simplicity.

Usability.

Compatibility.

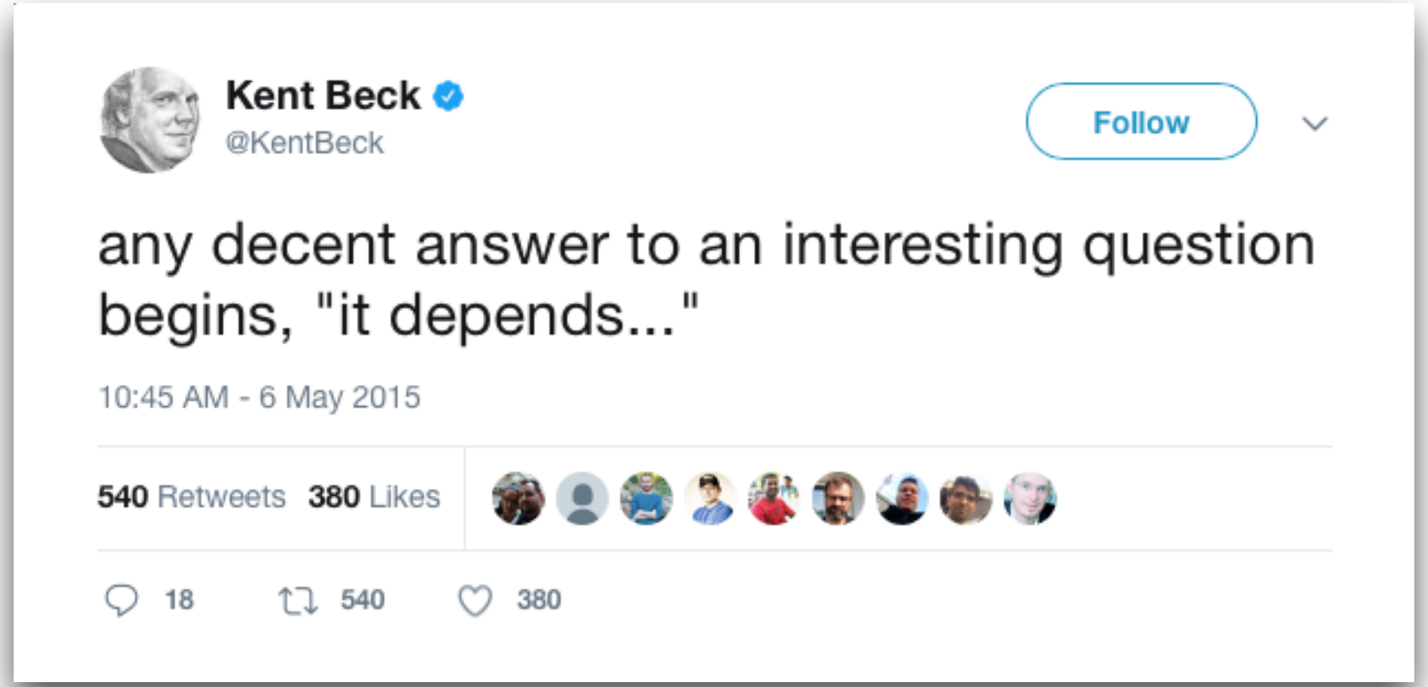
Fault tolerance.

Modularity.

The list goes on and on!

http://en.wikipedia.org/wiki/List_of_system_quality_attributes

What quality attributes do you focus on?



https://twitter.com/KentBeck/status/596007846887628801

Depends a lot on the type of software you build!

We don’t get to turn every knob to 11 do we?

Inverse relationships.

Maximizing one may minimize another.

Security and usability for instance.

It’s a balance.

Some are obvious to our customers.

If the systems won't support the user load…

heartbleed.com

The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

What leaks in practice?

We have tested some of our own services from attacker's perspective. We attacked ourselves from outside, without leaving a trace. Without using any privileged information or credentials we were able steal from ourselves the secret keys used for our X.509 certificates, user names and passwords, instant messages, emails and business critical documents and communication.

How to stop the leak?

As long as the vulnerable version of OpenSSL is in use it can be abused. Fixed OpenSSL has been released and now it has to be deployed. Operating system vendors and distribution, appliance vendors, independent software vendors have to adopt the fix and notify their users. Service providers and users have to install the fix as it becomes available for the operating systems, networked appliances and software they use.

BUSINESS
SEP 14 2017, 3:21 PM ET

Equifax Hackers Exploited Months-Old Flaw

by BEN POPKEN

SHARE
Share
Tweet
Email
Print

Equifax announced late Wednesday that the source of the hole in its defenses that enabled hackers to plunder its databases was a massive server bug first revealed in March.

For the rest of the IT world, fixing that flaw was a "hair on fire moment," a security expert said, as companies raced to install patches and secure their servers. But at Equifax, criminals were able to pilfer data from mid-May to July, when the credit bureau says it finally stopped the intrusion.

Equifax, Software Company Blame Each Other for Security Breach 1:52

"We know that criminals exploited a U.S. website application vulnerability," Equifax said in an update on its website Wednesday night. "The vulnerability was Apache Struts CVE-2017-5638." Equifax said it was working with a leading cybersecurity firm, reported to be Mandiant, to investigate the breach. Mandiant declined an NBC News request for comment.

Related: The One Move to Make After Equifax Breach

The Apache Software Foundation, which oversees the Apache Struts project, said in a press release Thursday that a software update to patch the flaw was

advertisement

FROM THE WEB
Sponsored Links

Find Out In One Minute If You Pre-Qualify For A Citi Card
Citi

Why These 10 SUVs are the Cream of the Crop
Kelley Blue Book

by Taboola

MORE FROM NBC NEWS

Fairly easy to convince people of the importance.

Others - are “invisible”.

Or at least harder to see.

Maintainability and simplicity for instance.

How do we get the decision makers to buy in?

What techniques can we use to influence them?

Outline the benefits.

Find common ground.

Avoid aggression.

Listen.

Have a conversation!

Can be hard to convince people!

Two approaches…

Find the influencers.

Influence the influencers.

Approach as equals.

Rely on the strength of your ideas and your reputation.

Your reputation speaks for you when you aren’t there.

Not sure what your rep is?

Ask.

May not like the answer…

But you can work to change it.

Find common ground.

Reciprocity rules…

Be helpful.

Be respectful.

Research your ideas.

Use trusted sources.

Recruit credible allies.

Nothing wrong with bringing help!

Speak their language.

Avoid techno babble.

You may be impressed by the jargon…

Most customers aren’t.

What resonates in your organization?

Cost savings?

Developer productivity?

Speed to market?

Shape your approach accordingly.

The iDon’tDrive System.

Self driving cars are all the rage these days.

Let’s pretend our company has the answer!

VCs can’t give us money fast enough.

Customers are flocking to our stylish website.

Our team has been tasked to build a backend system.

The cars generate a lot of data.

Battery level, health of the engine, maintenance.

Basics

- Car “phones home” on a regular basis.
- Sends a standard data payload including VIN.
- Demand is extremely high.
- Expect millions of cars on the road in the next 3 years.
- Marketing is full of great ideas…
- System must be available 24x7.

Customer Facing

- Web interface as well as mobile apps.
- Allow an owner to check the stats of their car.
- When does it need maintenance? How's the battery? Etc.
- Must be secure - only access *your* car.
- Allow owner to summon the car to their present location.
- Push notifications for maintenance, low battery etc.

Company Facing

- Fleet generates a lot of information that can be mined.
- Data must be anonymized.
- Ensure only authorized users have access.
- Audit access to customer data.
- Push software updates to the car.
- Push recall/update information to customers.

What quality attributes matter most here?

What words/phrases stand out to you?

Basics

- Car "phones home" on a regular basis.
- Sends a standard data payload including VIN.
- Demand is extremely high.
- Expect millions of cars on the road in the next 3 years.
- Marketing is full of great ideas…
- System must be available 24x7.

Customer Facing

- Web interface as well as mobile apps.
- Allow an owner to check the stats of their car.
- When does it need maintenance? How's the battery? Etc.
- Must be secure - only access *your* car.
- Allow owner to summon the car to their present location.
- Push notifications for maintenance, low battery etc.

Company Facing

- Fleet generates a lot of information that can be mined.
- Data must be anonymized.
- Ensure only authorized users have access.
- Audit access to customer data.
- Push software updates to the car.
- Push recall/update information to customers.

Auditability! Availability.
Security. Usability.

How would you rank them?

Depends on the
perspective of the system!

From a driver/owner…

Rank	Quality Attribute	Comments
1	Usability	Customers will not read a manual on how to use the iDon’tDrive system.
2	Availability	Owner must be able to summon a car 24x7.
3	Security	Owner must be confident that only then can access their car.
4	Reliability	System must work as expected when called upon to maintain confidence in the system.

From a service center…

Rank	Quality Attribute	Comments
1	Security	Only authorized users should have access to the system.
2	Auditability	System should audit access to determine appropriate usage of the system.
3	Efficiency	System should be minimize the time service representatives need to find information.
4	Usability	System should require minimal training and allow new service reps to be productive quickly.

What architectural decisions result from that?

Rank	Quality Attribute	Decision(s)
1	Usability	UX designers will be engaged and ensure the design requires no training to use.
2	Availability	System will be geographically dispersed across multiple data centers. Zero downtime deploys will be utilized.
3	Security	Standard three zone security will be employed. System will encrypt personally identified information and follow all security standards.
4	Reliability	System will be geographically dispersed across multiple data centers.

Questions?

Establish principles.

We can’t be everywhere…

We can’t be involved
with every decision.

We can establish principles.

Guard rails.

Guide posts.

North stars.

Create the environment within which our projects can thrive.

But how do we know if projects are following our principles?

Fitness functions.

We're all familiar with the second law of thermodynamics…

Otherwise known as a teenagers bedroom.

The universe really
wants to be disordered.

Software is not immune from this!

We go through the thoughtful effort to establish an architecture…

How do we maintain it?

We can’t spend every minute of every day on every project.

How do we ensure teams continue to make good decisions?

We cannot predict the future.

That’s not entirely true.

One constant - change.

Architecture is often defined as the decisions that are hard to change.

Or the decisions we wish we got right.

But we *know* things will change!

Isn’t this approach anti agile?

Contributing factor to the “we’re agile, we don’t have architects” theory.

You definitely have people making architectural decisions!

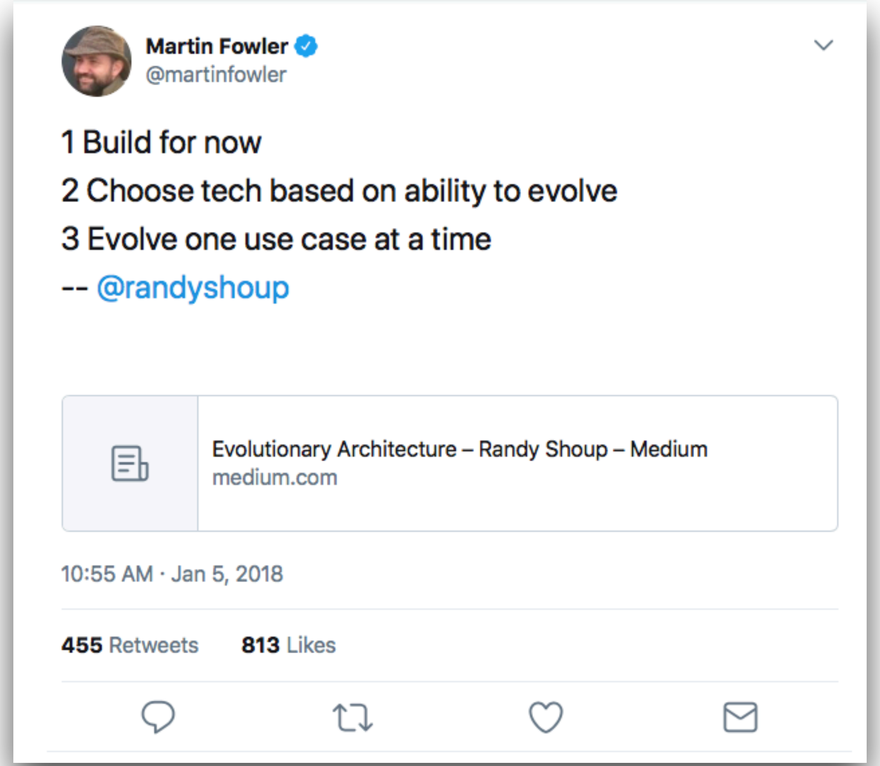
Sure hope they are making good ones…

You’ll know in a year or two.

“Our app has 4 different UI frameworks…”

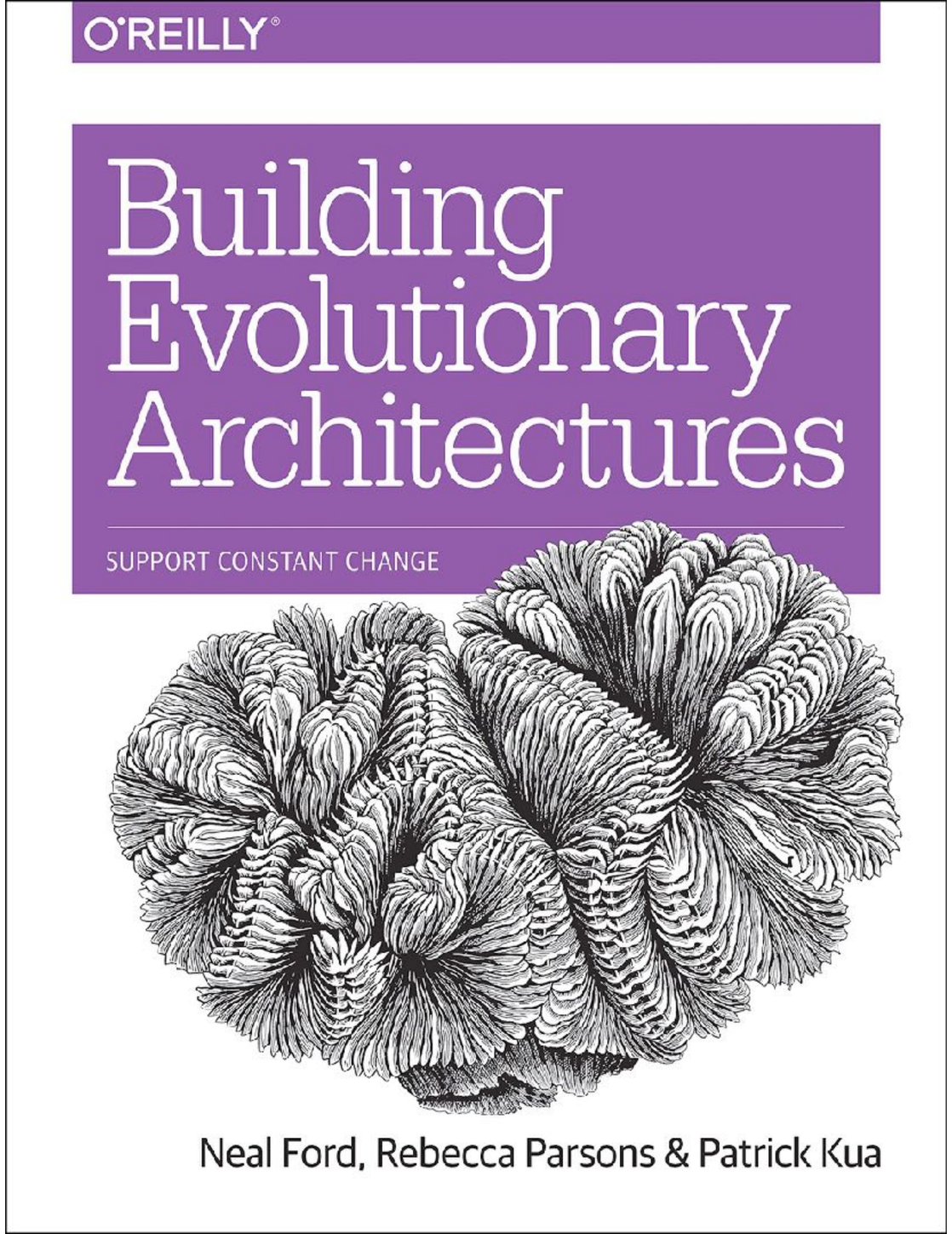
What do we do about that?

Maybe we should change our assumptions.



https://mobile.twitter.com/martinfowler/status/949323421619548161

What if our architectures expected to change?

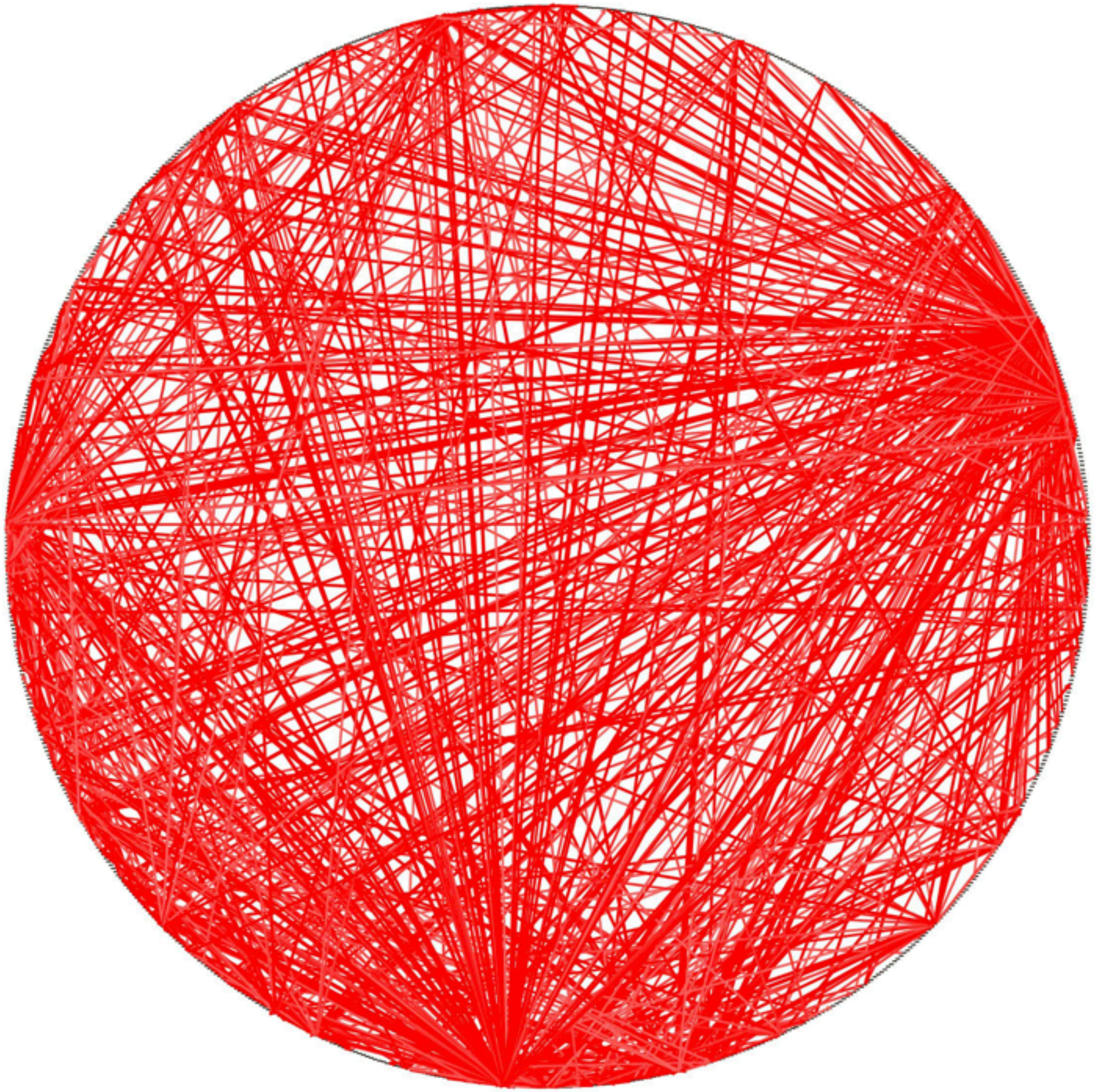


http://evolutionaryarchitecture.com

An evolutionary architecture supports guided, incremental change across multiple dimensions.

— Building Evolutionary Architectures

Some architectures are more evolvable than others…



http://evolutionaryarchitecture.com

Components are deployed,
features are enabled via toggles.

Allows us to change incrementally.

Also perform hypothesis driven development!

But how do we ensure the architecture still meets our needs?

How do we know if a solution violates part of the architecture?

Fitness functions!

Concept comes from evolutionary computing.

Is this mutation a success?

Are we closer to or further from our goal?

For architecture, it is all about protecting the ilities.

And balancing the tradeoffs.

We want to capture and preserve the key architectural characteristics.

First, we need to identify those key measures for project success.

Service Level Indicators if you will.

What can we measure?

Sometimes we let what we can measure dictate too much…

Just because we can measure it doesn't mean it matters!

Lines of code anyone?

Once we have our metrics,
we can set some goals.

Service Level Objectives.

SLO !== SLA!

Now we can create a fitness function!

Basically, a set of tests we execute to validate our architecture.

How close does this particular design get us to our objectives?

Ideally, all automated. But we may need some manual verifications.

For example…

All service calls must respond within 100 ms.

Cyclomatic complexity shall not exceed X.

Hard failure of an application will spin up a new instance.

NETFLIX

SIMIAN ARMY

https://github.com/Netflix/SimianArmy

Chaos Engineering.

https://medium.com/production-ready/chaos-monkey-for-fun-and-profit-87e2f343db31

Fitness functions remind us what is important in our architecture.

Informs our thinking about tradeoffs.

Different categories of fitness functions.

Atomic vs. Holistic.

Some characteristics must be tested in isolation…others cannot.

Holistic fitness functions test combined features.

We can’t test every possible combination!

Must be selective, driven by the value of the architectural characteristic.

Triggered vs. Continual.

Must consider frequency of execution.

Fitness functions can be triggered by something - checkin, QA pass…

Continual tests are just that.

Monitoring Driven Development!

http://benjiweber.co.uk/blog/2015/03/02/monitoring-check-smells/

Static vs. Dynamic.

Static tests have a fixed result - they either pass or they fail.

Nearly any test based on a metric.

Other fitness functions have a shifting definition of success.

Generally defined within a range of acceptable outcomes.

Automated vs. Manual.

Automation is good!

Ideally most of our fitness functions will live in our deployment pipeline.

Not everything is amenable to automation though…

Legal.

Existing projects.

Temporal fitness functions.

Essentially a reminder.

Check for an upgrade of library X.

Break upon upgrade tests.

Clearly we want to identify fitness functions as early as we can.

The discussion about the tradeoffs is invaluable to our understanding.

Help us prioritize features.

May lead us to break a system up to isolate certain features.

We can’t know everything up front.

Fitness functions will emerge as the system changes.

But we should strive to identify as many as we can up front.

We can also classify fitness functions.

Key - critical decisions.

Relevant - considered but unlikely to influence the architecture.

Not Relevant - won’t impact our decisions.

Can still be very useful to identify the non relevant dimensions!

Keep fitness functions visible!

Need to review the fitness functions.

Are they still relevant?

Are there new dimensions we need to track?

Are there better ways of measuring/ testing our current fitness functions?

Aim for at least an annual review.

Architectural Decisions.

They happen!

How do we document them?

Title/ID.

What is the problem?

List assumptions and constraints.

What are the options?

List the pros and cons of each alternative.

Which one did you chose?

Why?

Status - is this a proposal or has this been accepted? Is it now deprecated?

Consequences of the decision.

Consider a “time capsule”.

Screen cast or podcast of what you did and why.

Prevent Monday morning quarterbacking…

Or just “why did we do this?”

Should be stored in version control.

Keep old decisions around - just make sure they are marked appropriately!

Architecting is hard…

We have a lot to juggle!

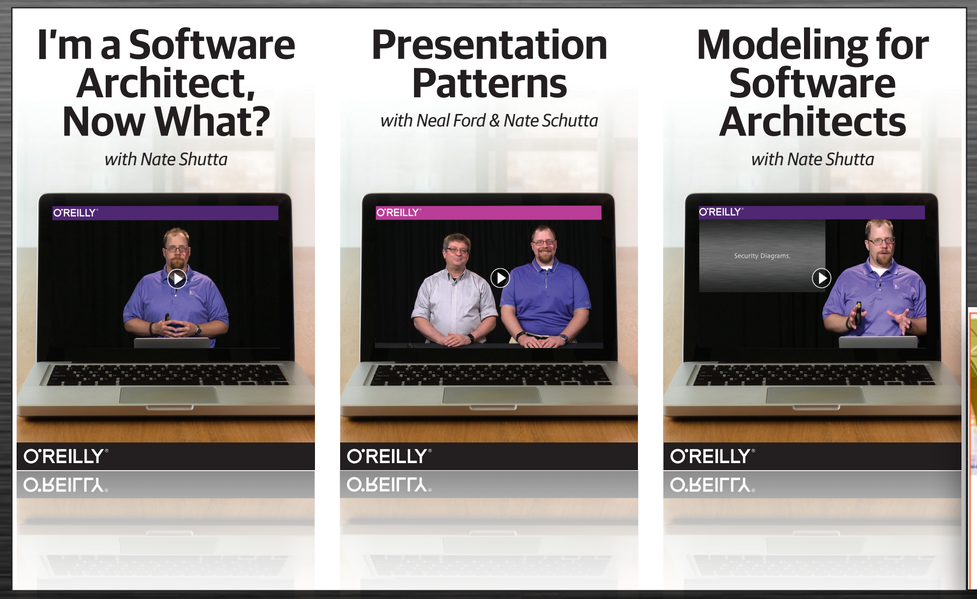
Important that we thing strategically.

We can’t afford
Resume Driven Design.

Good luck!

Questions?

Thanks!



Nathaniel T. Schutta
@ntschutta
ntschutta.io