

THE SEVEN (MORE) DEADLY SINS OF MICROSERVICES

DANIEL BRYANT

@DANIELBRYANTUK

OPENCREDO

PREVIOUSLY, AT QCON NYC 2015...

THE SEVEN DEADLY SINS (OF MICROSERVICES)

1. **LUST** - USING THE LATEST AND GREATEST TECH
2. **GLUTTONY** - EXCESSIVE COMMUNICATION PROTOCOLS
3. **GREED** - ALL YOUR SERVICE ARE BELONG TO US
4. **SLOTH** - CREATING A DISTRIBUTED MONOLITH
5. **WRATH** - BLOWING UP WHEN BAD THINGS HAPPEN
6. **ENVY** - THE SHARED SINGLE DOMAIN FALLACY
7. **PRIDE** - TESTING IN THE WORLD OF TRANSIENCE

12/08/15

@danielbryantuk

OpenCredo
delivering emerging technology today

<https://www.infoq.com/presentations/7-sins-microservices>

14/06/2016

@danielbryantuk

OpenCredo
delivering emerging technology today

THE SEVEN (MORE) DEADLY SINS OF MICROSERVICES

1. **LUST** - USING THE (UNEVALUATED) LATEST AND GREATEST TECH
2. **GLUTTONY** - COMMUNICATION LOCK-IN
3. **GREED** - WHAT'S MINE IS MINE (WITHIN THE ORGANISATION)
4. **SLOTH** - GETTING LAZY WITH NFRS
5. **WRATH** - BLOWING UP WHEN BAD THINGS HAPPEN
6. **ENVY** - THE SHARED SINGLE DOMAIN (AND DATA STORE) FALLACY
7. **PRIDE** - TESTING IN THE WORLD OF TRANSIENCE

@danielbryantuk



- CHIEF SCIENTIST AT **OPENCREDO**
 - ✓ TRANSFORMING ORGANISATIONS THROUGH **TECHNOLOGY AND TEAMS**
 - ✓ AGILE, LEAN, ARCHITECTURE, CI/CD, DEVOPS
 - ✓ MICROSERVICES, CLOUD, CONTAINERS, JAVA, GO, DOCKER, KUBERNETES
- LONDON JAVA COMMUNITY ASSOCIATE
- ADOPT OPENJDK AND JSR
- INFOQ EDITOR, DZONE MVB, VOXXED, O'REILLY





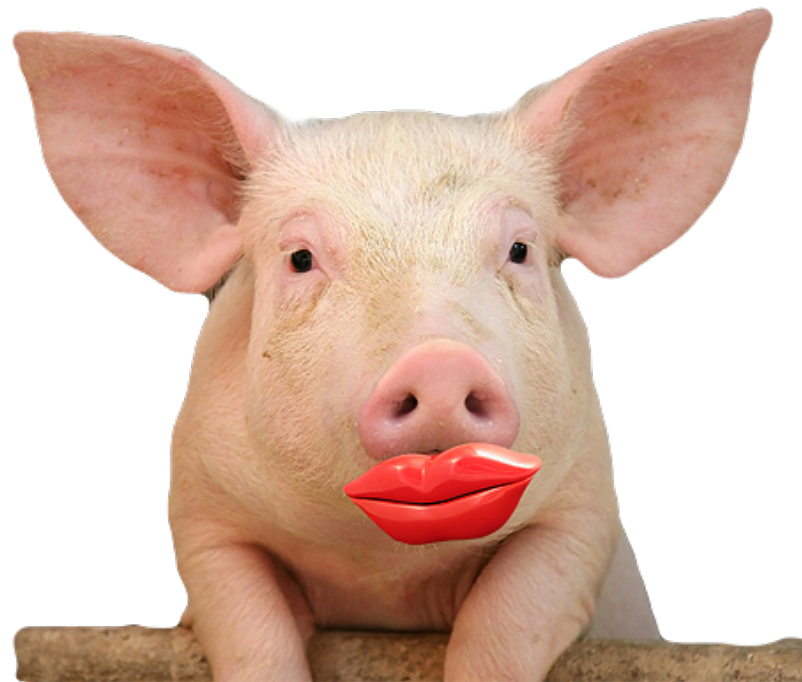
1. LUST - USING THE LATEST AND GREATEST TECH

PREVIOUSLY...

- MICROSERVICES ARE **NOT** ALWAYS A BEST FIT
 - ARCHITECTURAL SKILLS, STAGE OF BUSINESS, DEVOPS
- **EVALUATION** (AND DOCUMENTATION) ARE UNDER-USED SKILLS
 - LANGUAGE, FRAMEWORKS, MIDDLEWARE, DATA STORES

EVALUATION – ARE MICROSERVICES A GOOD FIT?

- NOT UNDERSTANDING **PRINCIPLES** (CARGO-CULTING)
 - NOT BUILT AROUND BUSINESS FUNCTIONALITY
 - MINI-MONOLITHS
- “OUR ‘MODE TWO’ APPS ARE MICROSERVICES”
 - NO TRANSFORMATION / MIGRATION **PLAN**
 - SOE EVOLUTION LIMITED BY SOR
 - LIPSTICK ON THE PIG
- NO WELL-DEFINED **DEVOPS / SRE / OPS**
 - DEPLOYMENT/OPS FREE-FOR-ALL



EVALUATION - SITUATIONAL AWARENESS

Speaker Deck

Published on Jun 9, 2016

Wardley Map

Value

Velocity

DevOps Culture

OODA Loop

Policy

Data Agility

App Ops

Microservices

Ops Tools

CI/CD

Registry

Orchestration

Container Runtime

IaaS

Genesis

Custom Built

Off the shelf

Commodity

Adrian Colyer

8 Presentations

★ Star this Talk 1 Star

Published in Technology

Stats 203 Views

Share

Twitter, Facebook

Embed

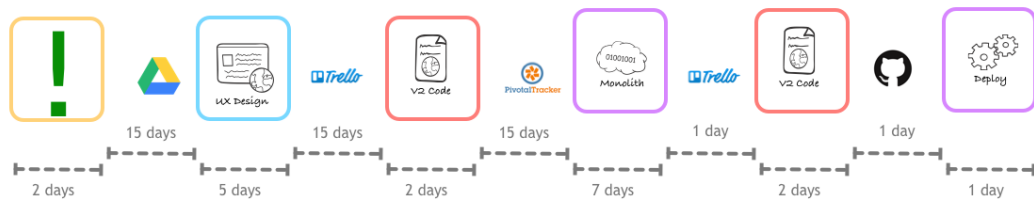
Direct Link

Download PDF

Making Sense of it All by Adrian Colyer

Published June 9, 2016 in Technology

speakerdeck.com/acolyer/making-sense-of-it-all



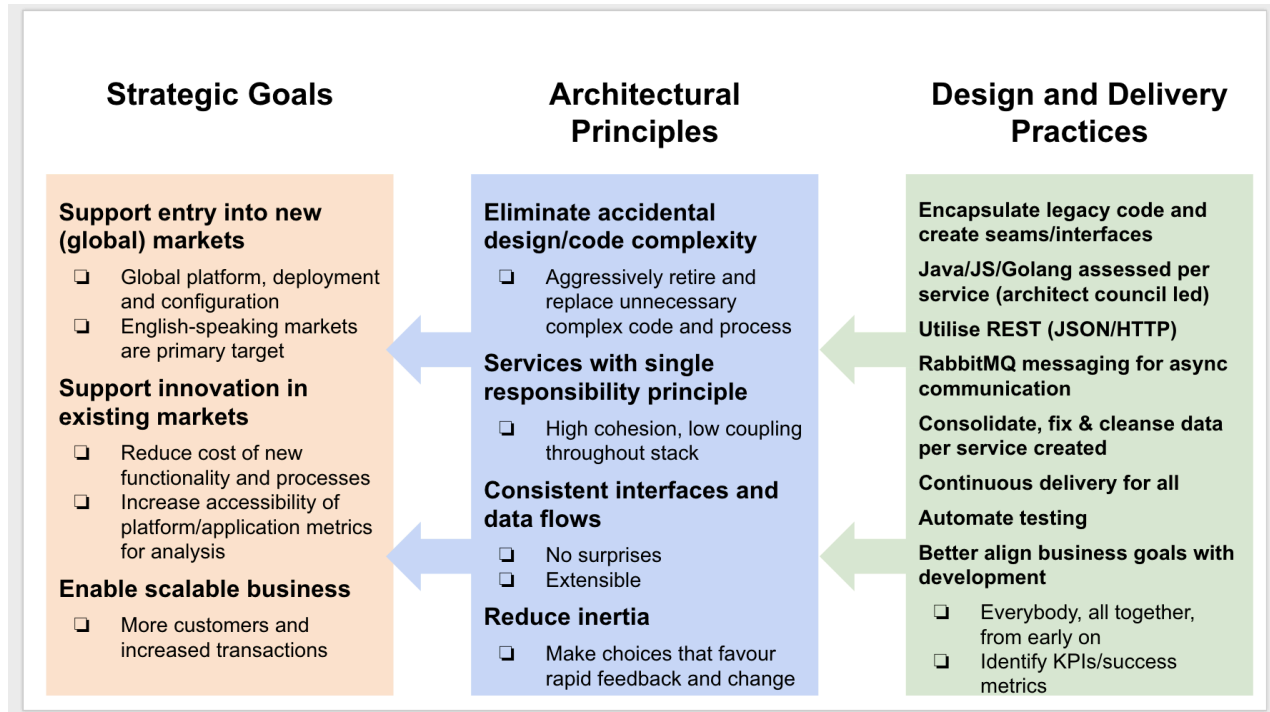
66 days

philcalcado.com/2015/09/08/how_we_ended_up_with_microservices.html

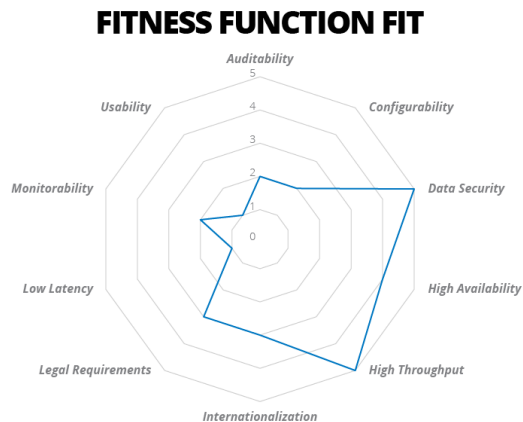
14/06/2016

@danielbryantuk

EVALUATION - START WITH WHY



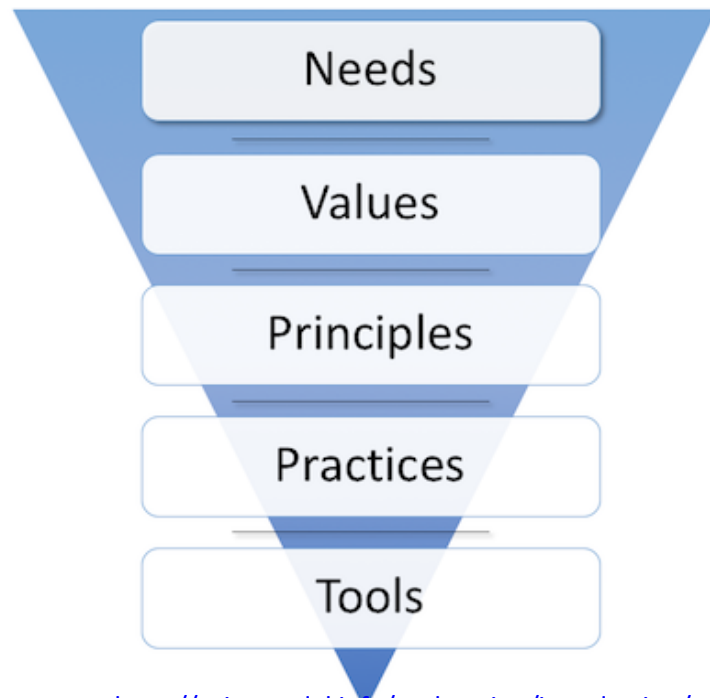
EVALUATION – FITNESS FUNCTIONS



- MICROSERVICES AS AN **EVOLUTIONARY ARCHITECTURE**
 - NEAL FORD AND REBECCA PARSONS
- GREAT FOR **EVALUATION AND DOCUMENTATION**
 - PLATFORMS / LANGUAGE
 - MIDDLEWARE
 - DATA STORES

EVALUATION – THE SPINE MODEL

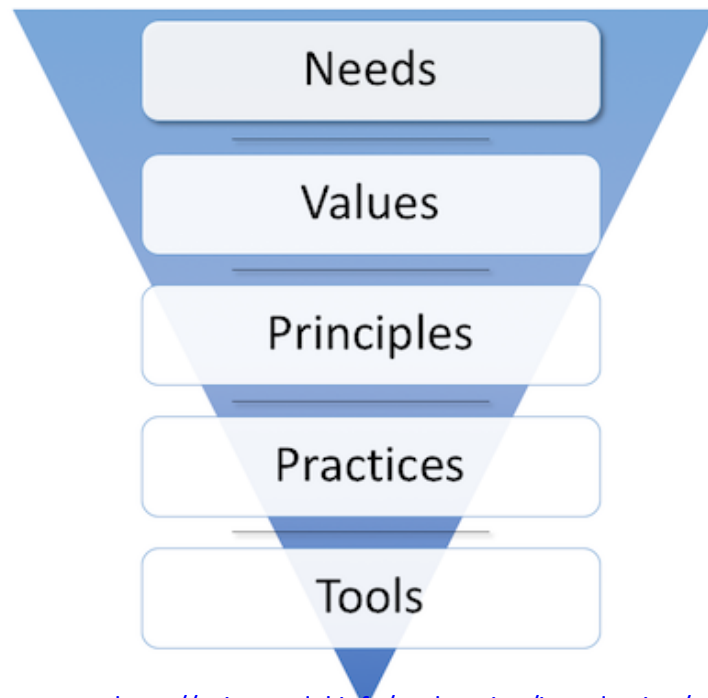
- EFFECTIVE **CONVERSATIONS** MAKE FOR EFFECTIVE **COLLABORATION**
- **IT'S A TOOL PROBLEM**
 - AS A SPECIES, WE HAVE ALWAYS BEEN TOOL USERS AND MAKERS.
 - WE USE _____ TO GET OUR WORK DONE
- PEOPLE GET STUCK IN A DILEMMA WHERE EQUALLY PLAUSIBLE OPTIONS ARE AVAILABLE
- “GOING UP THE SPINE” **BREAKS DEADLOCK**



<http://spinemodel.info/explanation/introduction/>

DETERMINE THE NEED FOR THE TOOL

- PRACTICES BEFORE TOOLS
 - DECIDE ON THE PRACTICES THAT THE TOOLS ARE THERE TO SUPPORT
 - WE DO _____ TO CREATE VALUE
- PRINCIPLES BEFORE PRACTICES
 - DECIDE ON THE PRINCIPLES TO MEASURE THOSE PRACTICES AGAINST.
 - WE LEVERAGE _____ TO CHANGE THE SYSTEM
- VALUES BEFORE PRINCIPLES
 - MAKE AS EXPLICIT AS POSSIBLE THE VALUES AT PLAY IN THE SYSTEM.
 - WE OPTIMISE FOR _____
- NEEDS BEFORE VALUES
 - IT ALL STARTS AT NEEDS. WHY DOES THIS SYSTEM EXIST IN THE FIRST PLACE?
 - WE ARE HERE TO SATISFY _____



<http://spinemodel.info/explanation/introduction/>

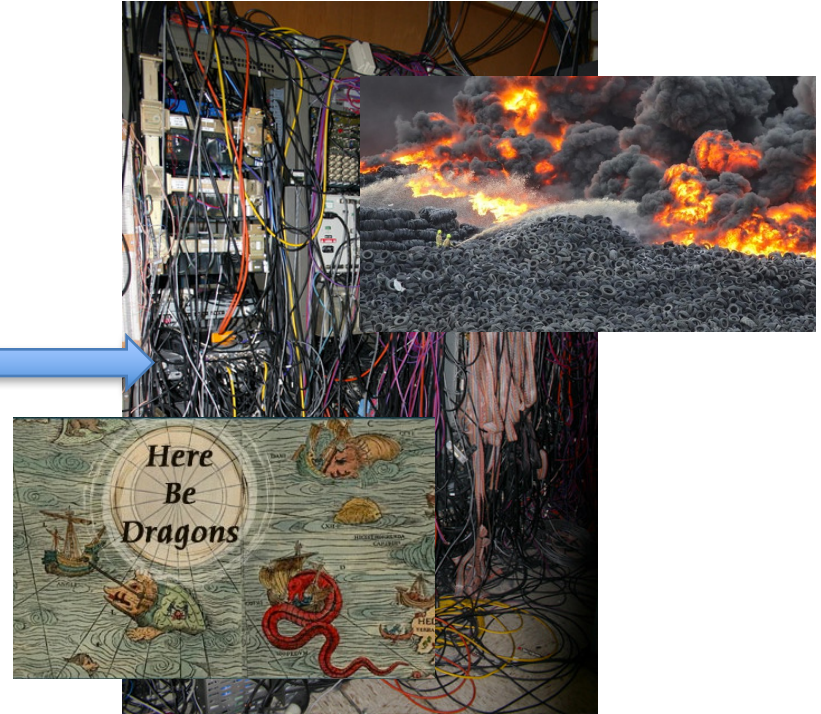
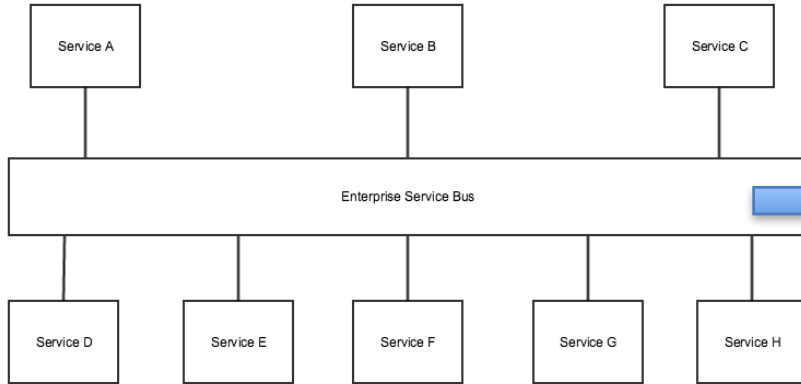


2. GLUTTONY - COMMUNICATION LOCK-IN

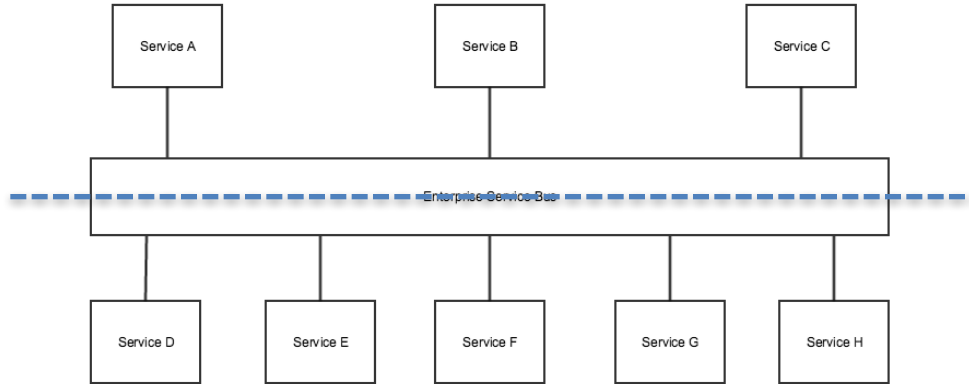
RPC – NOT THE DEVIL IN DISGUISE

- **DON'T RULE OUT RPC (E.G. GRPC)**
 - SOMETIMES THE **CONTRACT (AND SPEED)** ARE BENEFICIAL
 - HUMAN READABILITY OF JSON IS OVER-RATED
- **STICK TO REST (JSON OVER HTTPS) ON THE FRONT-END**
 - PRINCIPLE OF LEAST SURPRISE
 - BEST SUPPORT IN JAVASCRIPT/MOBILE

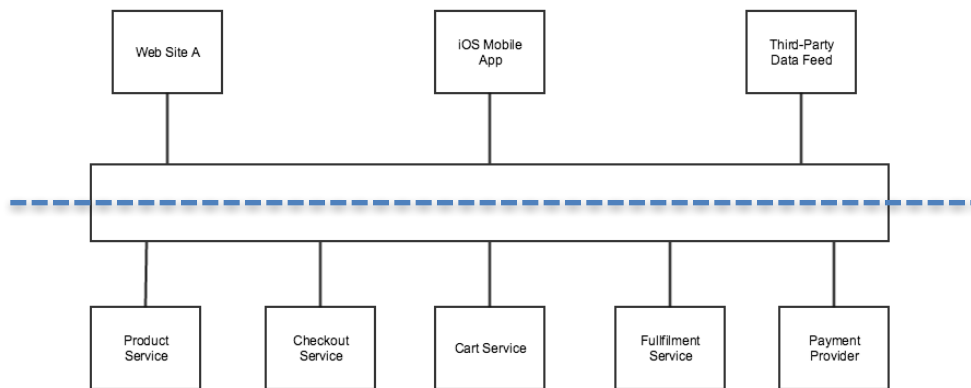
THE ESB IS DEAD - LONG LIVE THE ESB!



THE ESB IS DEAD - LONG LIVE THE ESB!

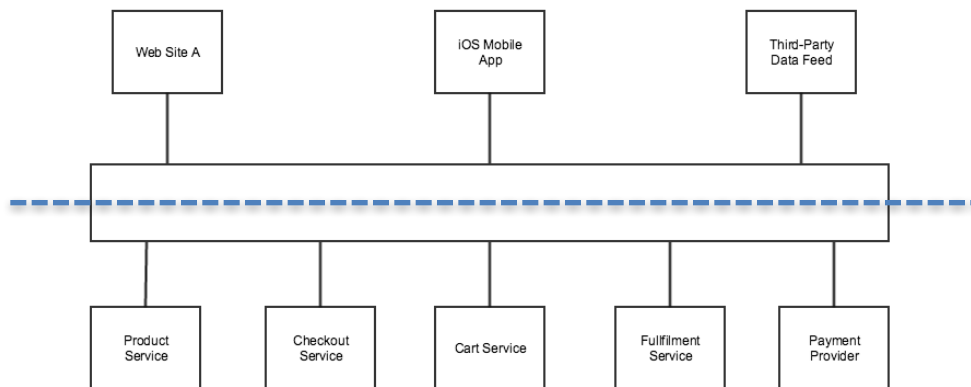


THE ESB IS DEAD – LONG LIVE THE ESB!



- IS THIS AN **ESB**?
- OR AN **API GATEWAY**?

THE ESB IS DEAD – LONG LIVE THE API GATEWAY!



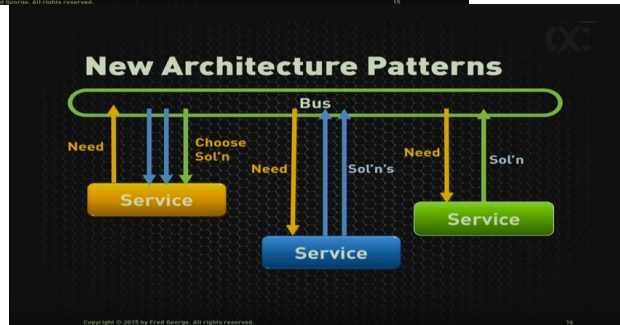
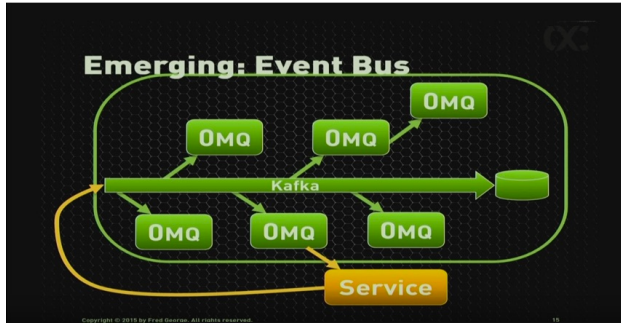
- WATCH FOR THE API GATEWAY MORPHING INTO AN ENTERPRISE SERVICE BUS

— LOOSE COUPLING IS VITAL

- BUT LET ME BE CLEAR...

- THE API GATEWAY PATTERN IS **AWESOME**
- CENTRALISE **CROSS-CUTTING** CONCERNS
- PREVENT WHEEL-REINVENTION (PLUGINS)
- CHECK OUT **KONG, APIGEE, AWS API GATEWAY, MULESOFT** ETC

ESB != EVENT BUS



Examples

A bare bones microservice

```
var seneca = require('seneca')()

seneca.add({ role: 'user', cmd: 'login' }, function (args, callback) {
  callback(null, { loggedIn: true })
})

seneca.listen()
```

A bare bones client

```
var seneca = require('seneca')()
var client = seneca.client()

client.act({ role: 'user', cmd: 'login' }, function (err, result) {
  console.log(result.loggedIn)
})
```

www.infoq.com/news/2016/02/not-just-microservices

www.youtube.com/watch?v=0pfghZxIFSg



3. GREED – WHAT'S MINE IS MINE... (WITHIN THE **ORGANISATION**)

PREVIOUSLY...

- CONWAY'S LAW
- MICROSERVICES ARE ABOUT **PEOPLE**, AS MUCH AS THEY ARE TECH
- GET YOUR BUSINESS READY

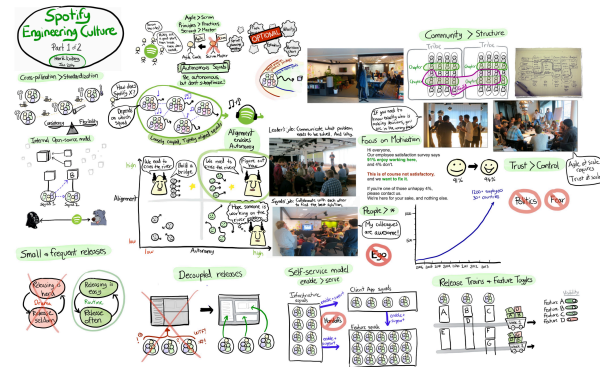
WE HEAR THIS A LOT...

“WE’VE DECIDED TO REFORM OUR TEAMS AROUND SQUADS, CHAPTERS AND GUILDS”

- BEWARE OF **CARGO-CULTING**

— REPEAT THREE TIMES “WE ARE NOT SPOTIFY”

- UNDERSTAND THE **PRACTICES, PRINCIPLES, VALUES** ETC



EMPATHY – THE HIDDEN INGREDIENT IN GOOD SOFTWARE DEVELOPMENT

Know others



05/06/2016

@danielbryantuk

Empathy
The hidden ingredient of good software development?

Daniel Bryant
@danielbryantuk



DevOps - it's not a department

- Pair with developers
- Treat operators as stakeholders
- Involve in standups
 - Communication face-to-face



05/06/2016

@danielbryantuk

OpenCredo

“Developer-on-call”



05/06/2016

@danielbryantuk

An occasional spike to the head
is a good thing...

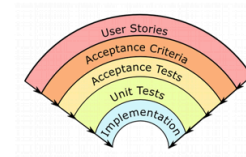
...metaphorically speaking

- **You build it, you run it**
 - Accountability
 - Shared responsibility
 - Communication

OpenCredo

Systems thinking – the user journey

- Understand the user journey
- “Shift left” QA
 - Three amigos
 - “Quality Advocates”
- BDD and TDD
 - Outside in



05/06/2016

@danielbryantuk

OpenCredo

<http://www.ustream.tv/recorded/86154111>

14/06/2016

@danielbryantuk

OpenCredo
delivering emerging technology today



4. SLOTH - GETTING LAZY WITH **NFRS**

14/06/2016

@danielbryantuk

GETTING LAZY WITH **NON-FUNCTIONAL REQUIREMENTS**

**“THE DRIVING TECHNICAL REQUIREMENTS FOR A SYSTEM SHOULD BE IDENTIFIED EARLY
TO ENSURE THEY ARE PROPERLY HANDLED IN SUBSEQUENT DESIGN”**

AIDAN CASEY

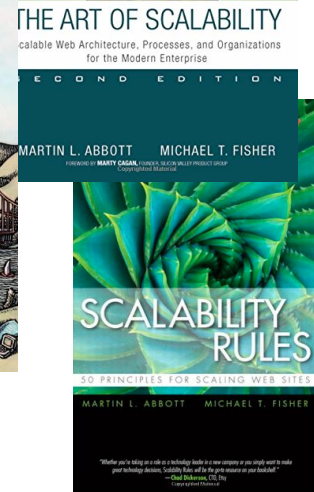
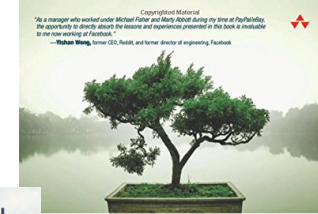
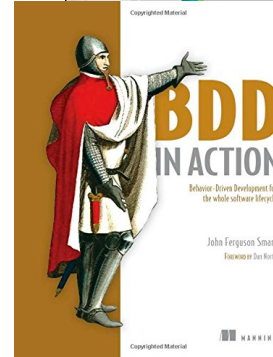
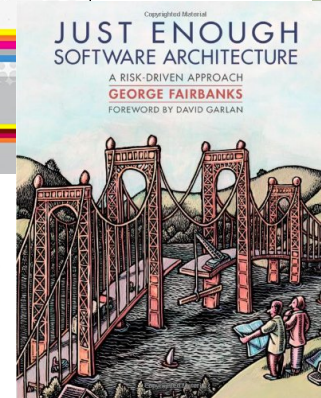
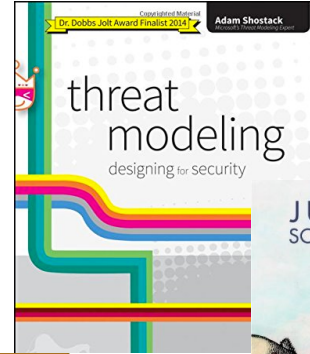
GUIDING PRINCIPLES FOR EVOLUTIONARY ARCHITECTURE

GETTING LAZY WITH NON-FUNCTIONAL REQUIREMENTS

- THE 'ILITIES' CAN BE (OFTEN) BE AN **AFTERTHOUGHT**
 - AVAILABILITY, SCALABILITY, AUDITABILITY, TESTABILITY ETC
- AGILE/LEAN: DELAY DECISIONS TO THE '**LAST RESPONSIBLE MOMENT**'
 - NEWSFLASH - **SOMETIMES THIS IS UP-FRONT**
- IT CAN BE COSTLY (OR PROHIBITIVE) TO ADAPT LATE IN THE PROJECT
 - **MICROSERVICES DON'T MAKE THIS EASIER** (SOMETIMES MORE DIFFICULT)

BEDTIME READING

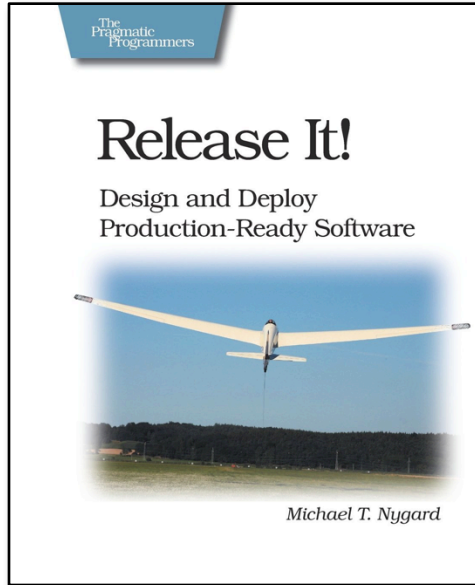
- PERFORMANCE AND LOAD TESTING
 - GATLING / JMETER
 - FLOOD.IO
- SECURITY TESTING
 - OWASP ZAP
 - BDD-SECURITY





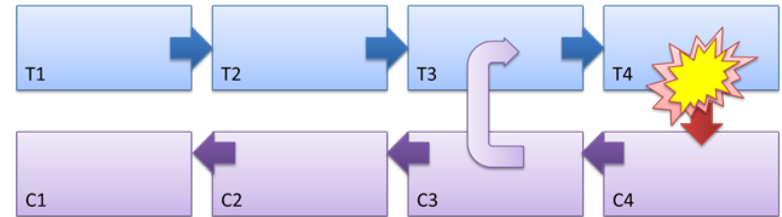
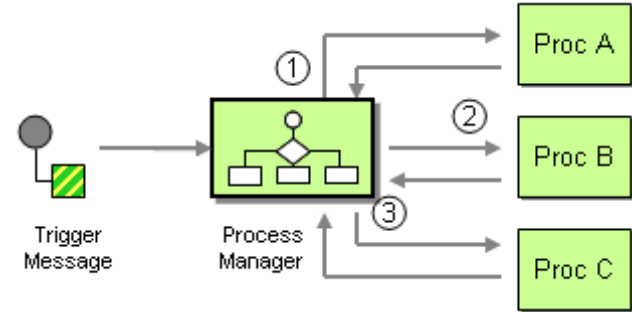
5. WRATH – BLOWING UP WHEN **BAD THINGS** HAPPEN

PREVIOUSLY – BRING IN **MICHAEL NYGARD** (OR SOME MONKEYS)



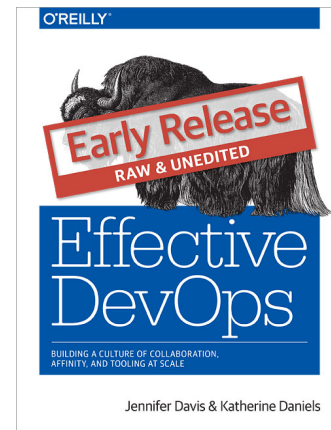
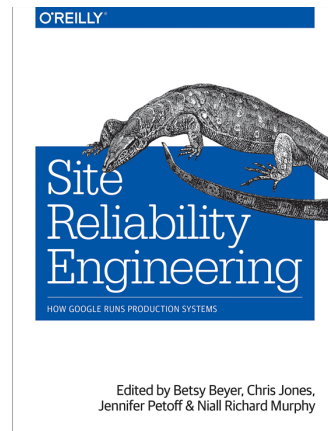
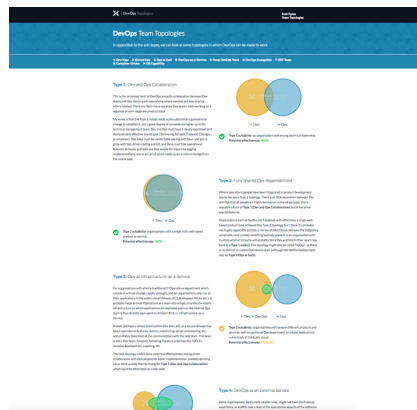
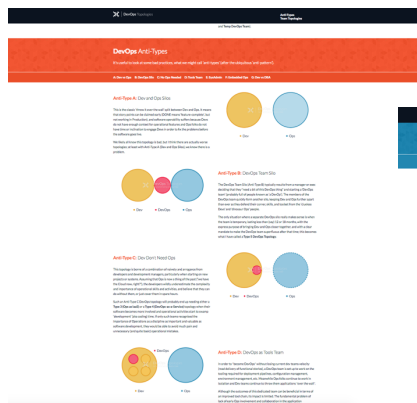
TECHNICAL PAIN POINT – DISTRIBUTED TRANSACTIONS

- **DON'T**
 - (WHERE POSSIBLE)
 - **PUSH TRANSACTIONAL SCOPE INTO SINGLE SERVICE**
- **SUPERVISOR/PROCESS MANAGER**
 - E.G. ERLANG OTP, AKKA, EIP
- **SAGA PATTERN**
 - WORKFLOWS PROVIDING A PATH (FORK) OF COMPENSATING ACTIONS



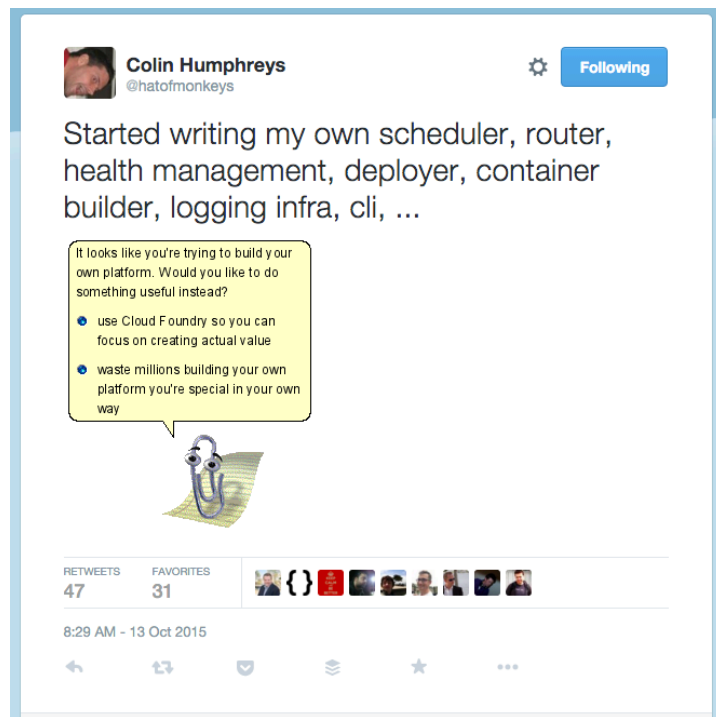
PEOPLE PAIN POINT – HOW DOES DEVOPS FIT INTO THIS?

- [HTTP://WEB.DEVOPSTOPOLOGIES.COM/](http://web.devopstopologies.com/)
- @MATTHEWPSKELTON
- @BEEROPS AND @SIGJE
- GOOGLE SRE



DEVOPS - DEFINE RESPONSIBILITIES

- DO YOU REALLY WANT TO BUILD AN **ENTIRE MICROSERVICES PLATFORM?**
- FOCUS ON **WHAT MATTERS**
 - CI/CD
 - MECHANICAL SYMPATHY
 - LOGGING
 - MONITORING



DEVOPS – THE 'FULLSTACK ENGINEER' MYTH

**“I’M SORRY, BUT IF YOU’RE NOT DESIGNING THE COMPUTER CHIPS AND
WRITING THE WEBSITE, THEN I DON’T WANNA HEAR FROM YOU”**

CHARITY MAJORS (@MIPSYTIPSY), CRAFTCONF 2016

[HTTP://WWW.USTREAM.TV/RECORDED/86181845](http://www.ustream.tv/recorded/86181845)

DEVOPS - RESPONSIBILITIES

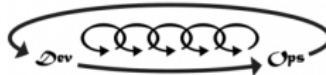
The First Way:
Systems Thinking



The Second Way:
Amplify Feedback Loops



The Third Way:
Culture Of Continual Experimentation And
Learning



We assert that the Three Ways describe the values and philosophies that frame the processes, procedures, practices of DevOps, as well as the prescriptive steps.

Gene Kim

“Outage Start Up” RACI Chart

Task/Positions	Maint. Technician	Maint. Supervisor	Prod Supervisor	Reliability Engineer	Safety	PdM Tech.
Verify Safety on all equipment	C	A	C	I	R	
Verify Equipment Reliability	C	A	I	R		C
Verify Equipment Functions	R	A	R	C	I	C
Clean Up	R	A	I			
Inspect and Return Tools	R	A				
Meeting on Lessons Learned From Outage	C	R	I	C	C	C

Responsibility
Accountable
Consulted
Informed

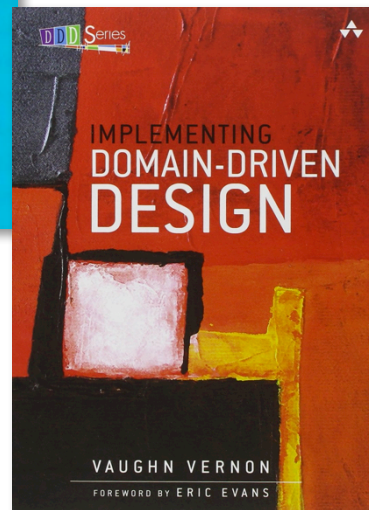
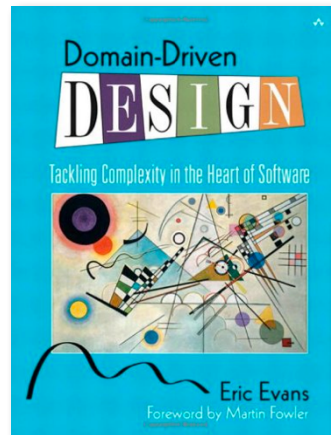
“the Doer”
“the Buck stops here”
“in the Loop”
“kept in the picture”



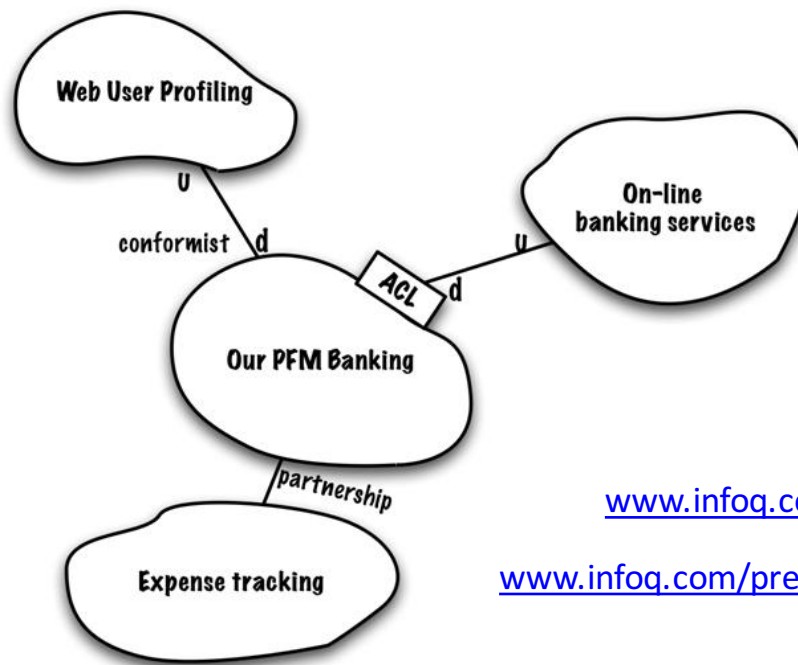
6. ENVY – THE SHARED SINGLE DOMAIN (AND DATA STORE) FALLACY

PREVIOUSLY – ONE MODEL TO RULE THEM ALL...

- ONE MODEL
 - BREAKS ENCAPSULATION
 - INTRODUCES COUPLING
- KNOW YOUR DDD
 - ENTITIES
 - VALUE OBJECTS
 - AGGREGATES AND ROOTS



CONTEXT MAPPING

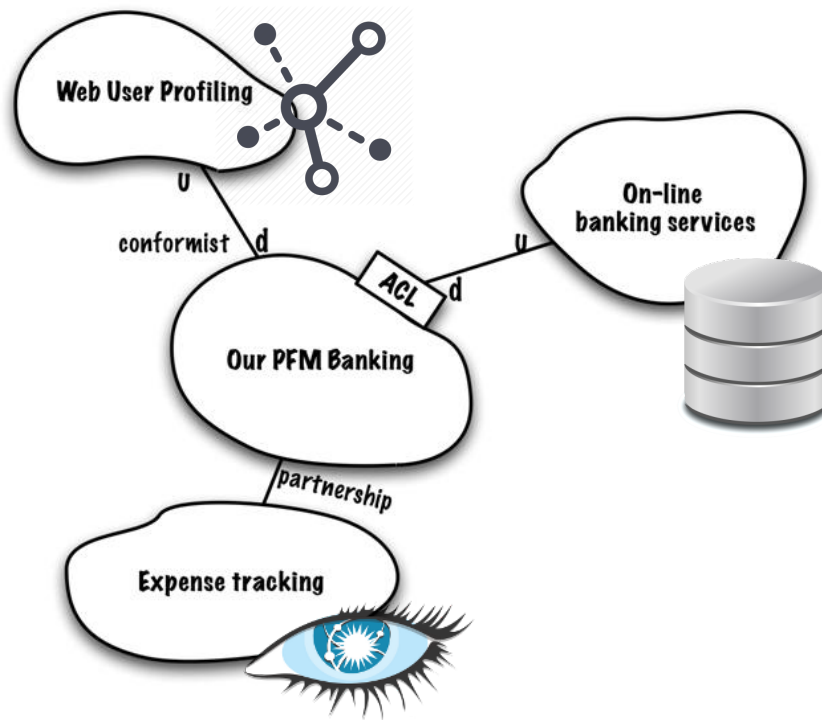


www.infoq.com/articles/ddd-contextmapping

www.infoq.com/presentations/ddd-microservices-2016

CHOOSE (AND USE) DATA STORES APPROPRIATELY

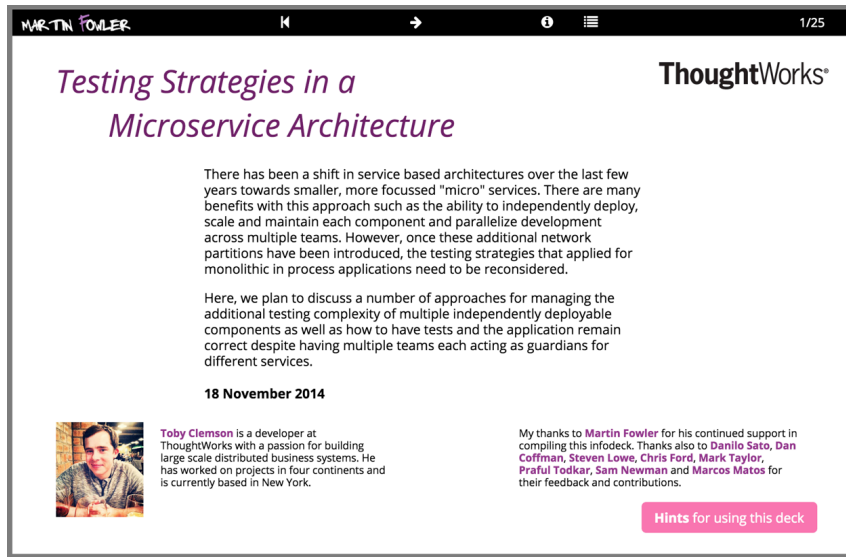
- RDBMS
 - VALUABLE FOR **STRUCTURED** DATA
- **CASSANDRA** IS AWESOME
 - BUT DON'T TREAT IT LIKE AN RDBMS!
- DON'T BUILD A GRAPH WITH RDBMS
 - USE **NEO4J, TITAN** ETC
- DATAGRIDS E.G. HAZELCAST
 - CACHING, **DISTRIBUTED PROCESSING**





7. PRIDE – TESTING IN THE **WORLD OF TRANSIENCE**

PREVIOUSLY...



MARTIN FOWLER 1/25


Testing Strategies in a Microservice Architecture

ThoughtWorks®

There has been a shift in service based architectures over the last few years towards smaller, more focussed "micro" services. There are many benefits with this approach such as the ability to independently deploy, scale and maintain each component and parallelize development across multiple teams. However, once these additional network partitions have been introduced, the testing strategies that applied for monolithic in process applications need to be reconsidered.

Here, we plan to discuss a number of approaches for managing the additional testing complexity of multiple independently deployable components as well as how to have tests and the application remain correct despite having multiple teams each acting as guardians for different services.

18 November 2014

 Toby Clemson is a developer at ThoughtWorks with a passion for building large scale distributed business systems. He has worked on projects in four continents and is currently based in New York.

My thanks to [Martin Fowler](#) for his continued support in compiling this infodeck. Thanks also to [Danilo Sato](#), [Dan Coffman](#), [Steven Lowe](#), [Chris Ford](#), [Mark Taylor](#), [Praful Todkar](#), [Sam Newman](#) and [Marcos Matos](#) for their feedback and contributions.

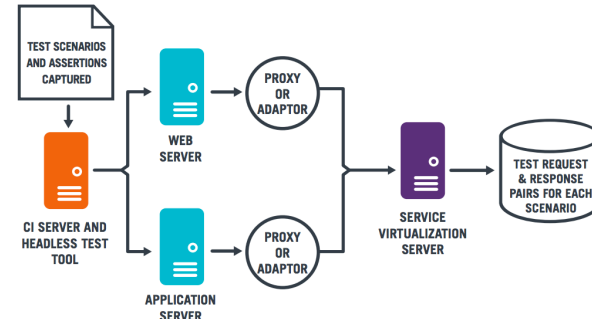
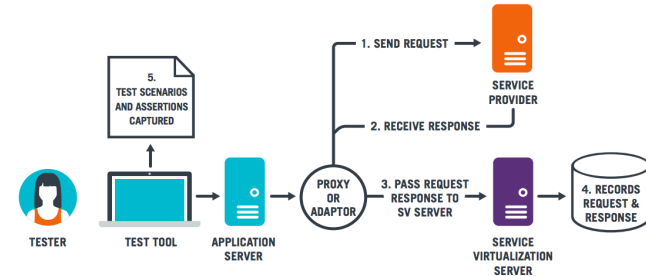
Hints for using this deck

martinfowler.com/articles/microservice-testing/

- LOCAL VERIFICATION
 - CONSUMER-BASED CONTRACTS
- END-TO-END
 - BDD-STYLE CRITICAL PATH
- REMEMBER THE TEST PYRAMID

SERVICE VIRTUALISATION / API SIMULATION

- VIRTUALISE REQUEST/RESPONSE OF SERVICES
 - UNAVAILABLE
 - EXPENSIVE TO RUN
 - FRAGILE/BRITTLE
 - NON-DETERMINISTIC
 - CANNOT SIMULATE FAILURES



[HTTPS://DZONE.COM/ARTICLES/CONTINUOUSLY-DELIVERING-SOA](https://dzone.com/articles/continuously-delivering-soa)

SERVICE VIRTUALISATION

- CLASSICS

- CA SERVICE VIRTUALIZATION
- PARASOFT VIRTUALIZE
- HPE SERVICE VIRTUALIZATION
- IBM TEST VIRTUALIZATION SERVER

- NEW KIDS ON THE BLOCK

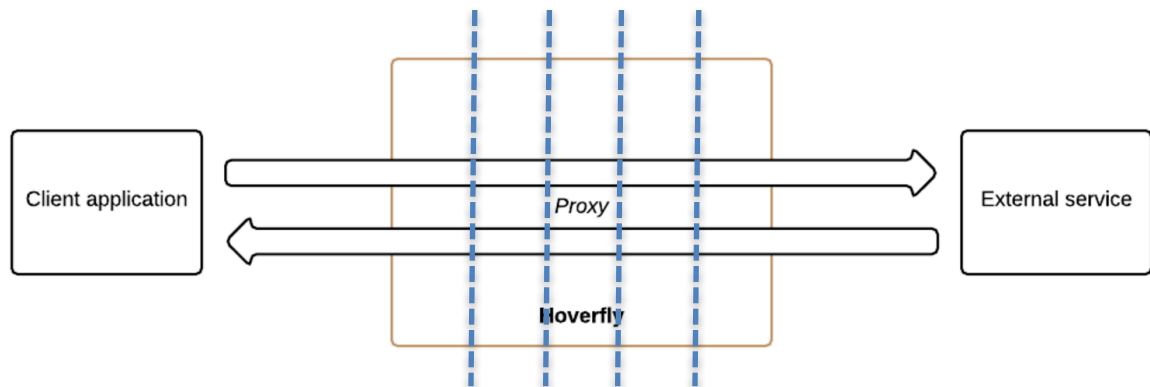
- HOVERFLY
- WIREMOCK
- VCR/BETAMAX
- MOUNTEBANK
- MIRAGE

HOVERFLY



- LIGHTWEIGHT **SERVICE VIRTUALISATION**
 - OPEN SOURCE (APACHE 2.0)
 - GO-BASED / SINGLE BINARY
 - WRITTEN BY @SPECTOLABS
- FLEXIBLE **API SIMULATION**
 - HTTP / HTTPS
 - MORE PROTOCOLS TO FOLLOW?

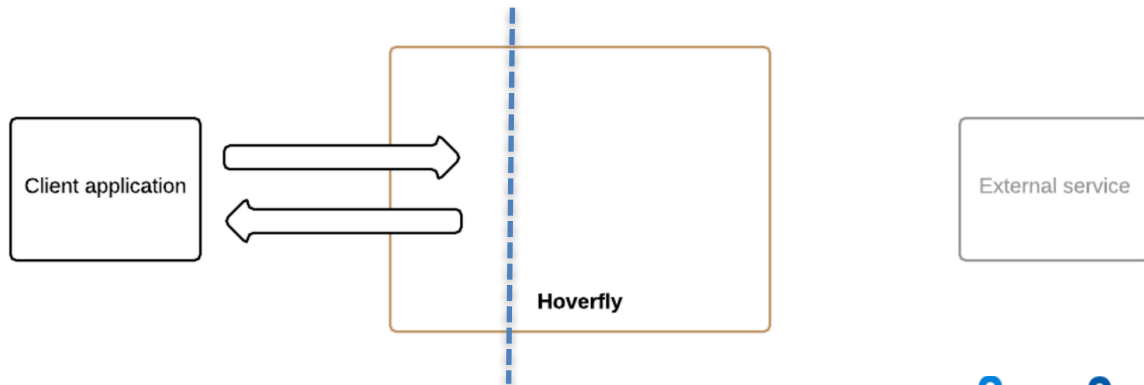
Capture mode



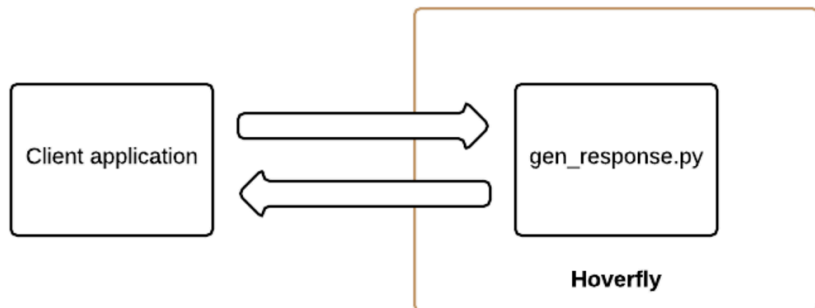
- Middleware
 - Remove PII
 - Rate limit
 - Add headers

Simulate mode

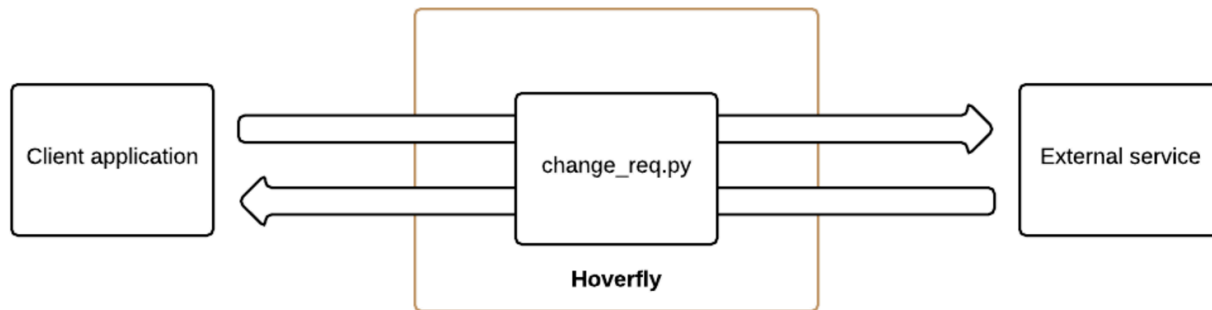
- Middleware
 - Fault injection
 - Chaos monkey



Synthesize mode



Modify mode



HOVERFLY JUNIT RULE

```
public class HoverflyRuleTest {

    @ClassRule
    public static HoverflyRule hoverflyRule = HoverflyRule.buildFromClassPathResource("test-service.json").build();

    private RestTemplate restTemplate;

    @Before
    public void setUp() {
        restTemplate = new RestTemplate();
    }


    @Test
    public void shouldBeAbleToMakeABooking() throws URISyntaxException {
        // Given
        final RequestEntity<String> bookFlightRequest = RequestEntity.post(new URI("http://www.my-test.com/api/bookings"))
            .contentType(APPLICATION_JSON)
            .body("{\"flightId\": \"1\"}");

        // When
        final ResponseEntity<String> bookFlightResponse = restTemplate.exchange(bookFlightRequest, String.class);

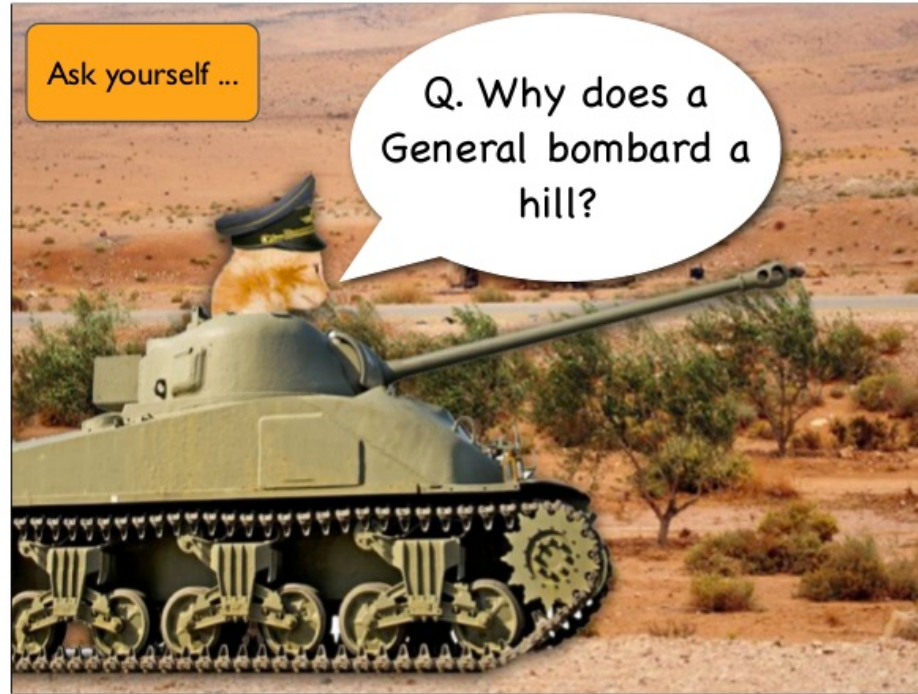
        // Then
        assertThat(bookFlightResponse.getStatusCode()).isEqualTo(CREATED);
        assertThat(bookFlightResponse.getHeaders().getLocation()).isEqualTo(new URI("http://localhost/api/bookings/1"));
    }
}
```

RIGHT, LET'S WRAP THIS UP...

AVOID BEING A SINNER...

- STRATEGY
 - PRINCIPLES/PREPARATION
 - RESPONSIBILITIES
- 
- DEFINED AND SHARED
 - UNDERSTOOD AND APPROPRIATE
 - CLEAR AND COMMUNICATED

STRATEGY (@SWARDLEY STYLE)



STRATEGY (@SWARDLEY STYLE)



STRATEGY

Speaker Deck Published on Jun 9, 2016

Wardley Map

Value

Genesis Custom Built Off the shelf Commodity

DevOps Culture Velocity OODA Loop Data Agility App Ops Microservices CI/CD Registry Orchestration Ops Tools Container Runtime IaaS

Published June 9, 2016 in Technology

Making Sense of it All by Adrian Colyer

Published June 9, 2016 in Technology

Adrian Colyer 8 Presentations

★ Star this Talk 1 Star

Published In Technology

Stats 203 Views

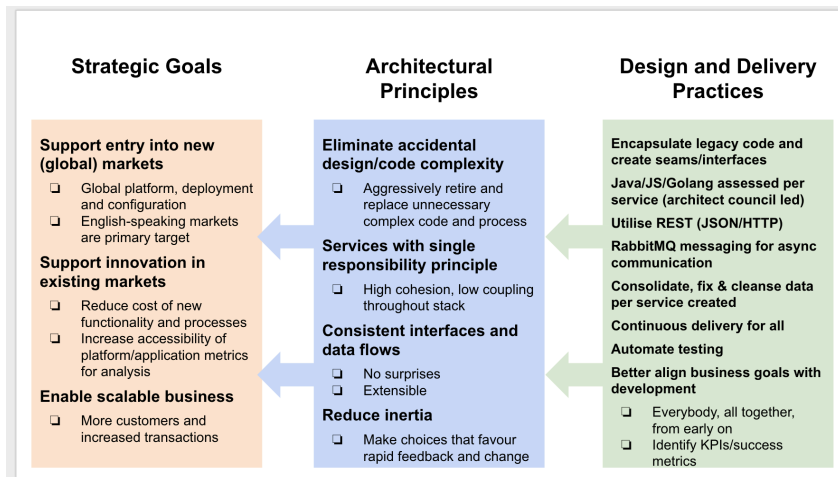
Share

Twitter, Facebook

</> Embed

Direct Link

Download PDF



PRINCIPLES/PREPARATION – IN THE OLDEN DAYS

“GIVE ME SIX HOURS TO CHOP DOWN A TREE
AND I WILL SPEND THE FIRST FOUR SHARPENING THE AXE”

ABRAHAM LINCOLN

PRINCIPLES/PREPARATION – WHAT I SEE...

“GIVE ME SIX HOURS TO CHOP DOWN A TREE
AND I WILL BEGIN PLANNING THE CONSTRUCTION OF A LARGE-SCALE LUMBER YARD
AND ON THE SECOND DAY I WILL DISCOVER NO-ONE KNOWS HOW TO USE AN AXE
AND ON THE THIRD DAY I WILL BUY A HANDSAW AND BEGIN SAWING”

ME, ATTEMPTING TO BE HUMOUROUS

PRINCIPLES / PREPARATION

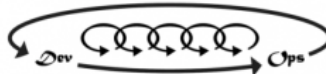
The First Way:
Systems Thinking



The Second Way:
Amplify Feedback Loops

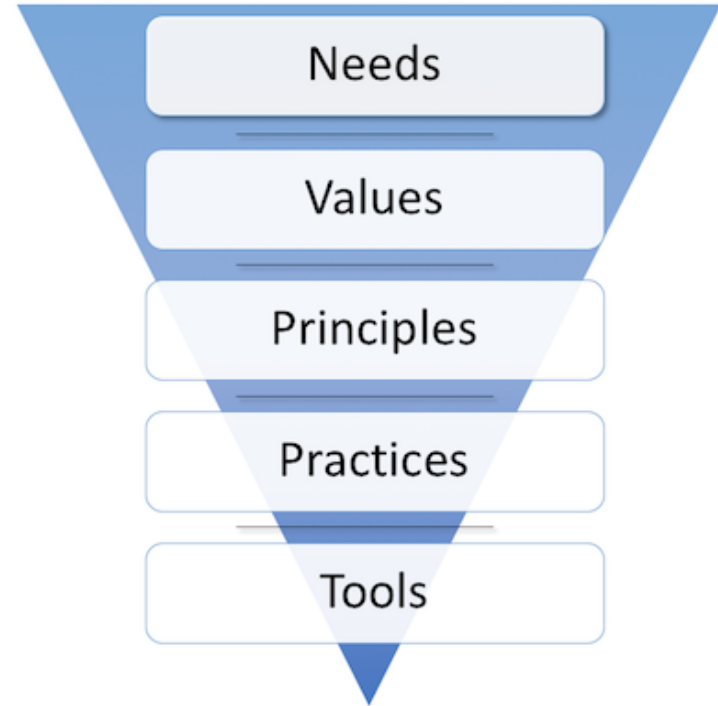


The Third Way:
Culture Of Continual Experimentation And
Learning



We assert that the Three Ways describe the values and philosophies that frame the processes, procedures, practices of DevOps, as well as the prescriptive steps.

Gene Kim



RESPONSIBILITIES



RESPONSIBILITIES

- DEFINE RESPONSIBILITIES

- RACI / RASCI

- ARCHITECTURE

- SUPER IMPORTANT IN MICROSERVICES
 - LESS IVORY TOWERS, MORE SIM CITY

- DEVOPS

- NOT A FREE-FOR-ALL
 - NO FULLSTACK HEROS

“Outage Start Up” RACI Chart

Task/Positions	Maint. Technician	Maint. Supervisor	Prod Supervisor	Reliability Engineer	Safety	PdM Tech.
Verify Safety on all equipment	C	A	C	I	R	
Verify Equipment Reliability	C	A	I	R		C
Verify Equipment Functions	R	A	R	C	I	C
Clean Up	R	A	I			
Inspect and Return Tools	R	A				
Meeting on Lessons Learned From Outage	C	R	I	C	C	C

Responsibility
Accountable
Consulted
Informed

“the Doer”
“the Buck stops here”
“in the Loop”
“kept in the picture”

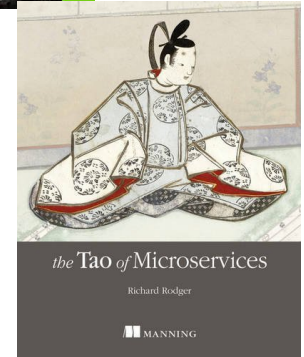
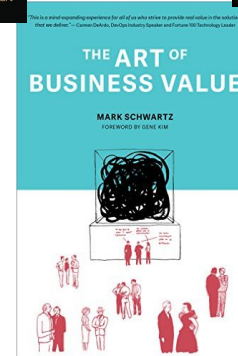
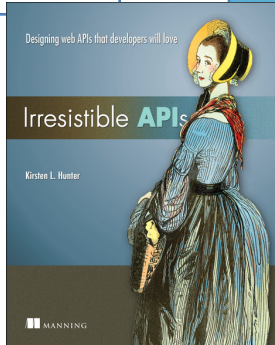
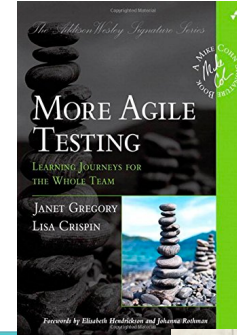
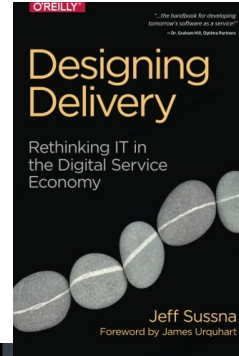
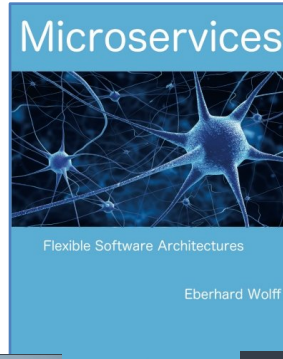
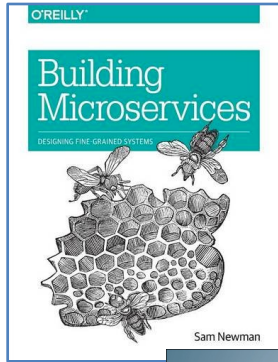
THE SEVEN (MORE) DEADLY SINS OF MICROSERVICES

1. **LUST** - USING THE (UNEVALUATED) LATEST AND GREATEST TECH
2. **GLUTTONY** - COMMUNICATION LOCK-IN
3. **GREED** - WHAT'S MINE IS MINE (WITHIN THE ORGANISATION)
4. **SLOTH** - GETTING LAZY WITH NFRS
5. **WRATH** - BLOWING UP WHEN BAD THINGS HAPPEN (TXNS AND OPS)
6. **ENVY** - THE SHARED SINGLE DOMAIN (AND DATA STORE) FALLACY
7. **PRIDE** - TESTING IN THE WORLD OF TRANSIENCE

THE SEVEN (MORE) DEADLY SINS OF MICROSERVICES

1. **LUST** - USING THE (**UNEVALUATED**) LATEST AND GREATEST TECH
2. **GLUTTONY** - **COMMUNICATION** LOCK-IN
3. **GREED** - WHAT'S MINE IS MINE (WITHIN THE **ORGANISATION**)
4. **SLOTH** - GETTING LAZY WITH **NFRS**
5. **WRATH** - BLOWING UP WHEN **BAD THINGS HAPPEN** (TXNS AND OPS)
6. **ENVY** - THE SHARED SINGLE DOMAIN (AND **DATA STORE**) FALLACY
7. **PRIDE** - **TESTING** IN THE WORLD OF TRANSIENCE

(MORE) BEDTIME READING



THANKS...

@DANIELBRYANTUK

DANIEL.BRYANT@OPENCREDO.COM

HTTP://MUSERVICESWEEKLY.COM/

(CREDIT TO **TAREQ ABEDRABBO** FOR INSPIRATION/GUIDANCE)