

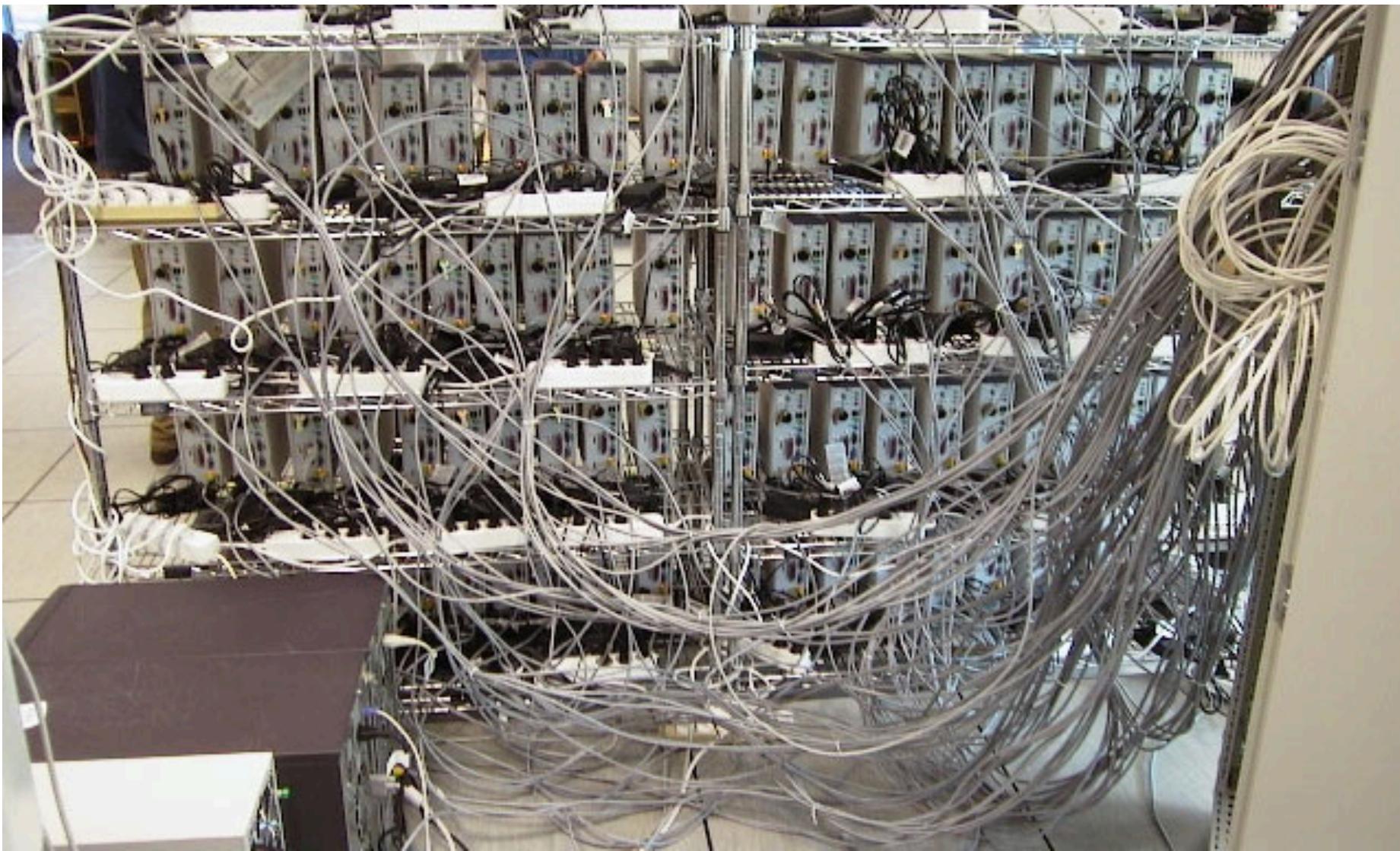
Motivation

Oxford University
Software Engineering
Programme
January 2018



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons
Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Distributed Computing



Why is Distributed Computing important?

- Scale
- Evolution
- Integration
- Cross-boundary
- Resilience



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Learning from

THE AMAZON

TECHNOLOGY

PLATFORM

Many think of Amazon as “that hugely successful online bookstore.” You would expect Amazon CTO Werner Vogels to embrace this distinction, but in fact it causes him some concern. “I think it’s important to realize that first and foremost Amazon is a technology company,” says Vogels. And he’s right. Over the past years, Vogels has helped Amazon grow from an online retailer (albeit one of the largest, with more than 55 million active customer accounts) into a platform on which more than 1 million active retail partners worldwide do business. Behind Amazon’s successful evolution from retailer to technology platform is its SOA (service-oriented architecture), which broke new technological ground and proved that SOAs can deliver on their promises.

Vogels came to Amazon from Cornell University, where he was working on high-availability systems and the management of scalable enterprise systems. He maintains that research spirit at Amazon, which regularly must solve problems never before encountered. “Maybe other companies call it research. We just call it development,” he points out.

Interviewing Vogels is ACM Turing Award winner and Microsoft Technical Fellow Jim Gray.

application, running on a Web server, talking to a database on the back end. This application, dubbed Obidos, evolved to hold

all the business logic, all the display logic, and all the functionality that Amazon eventually became famous for: similarities, recommendations, Listmania, reviews, etc.

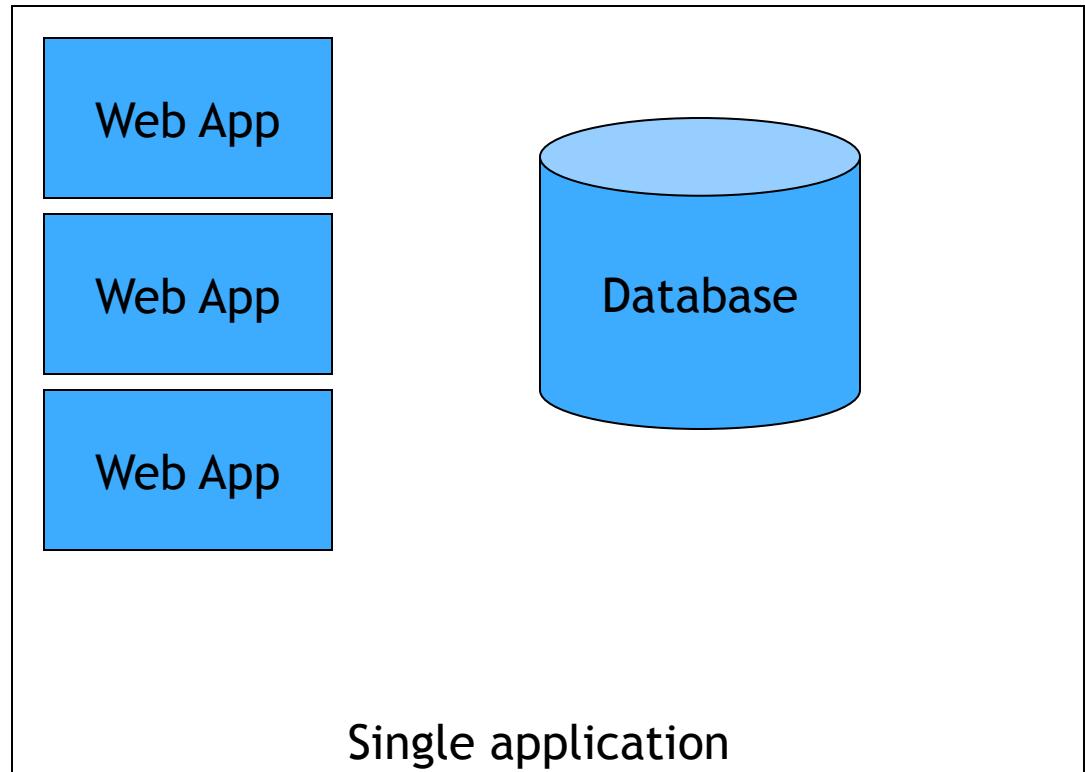
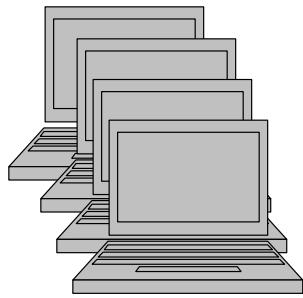


Source: Interview with Werner Vogels, ACM Queue
<https://queue.acm.org/detail.cfm?id=1142065>

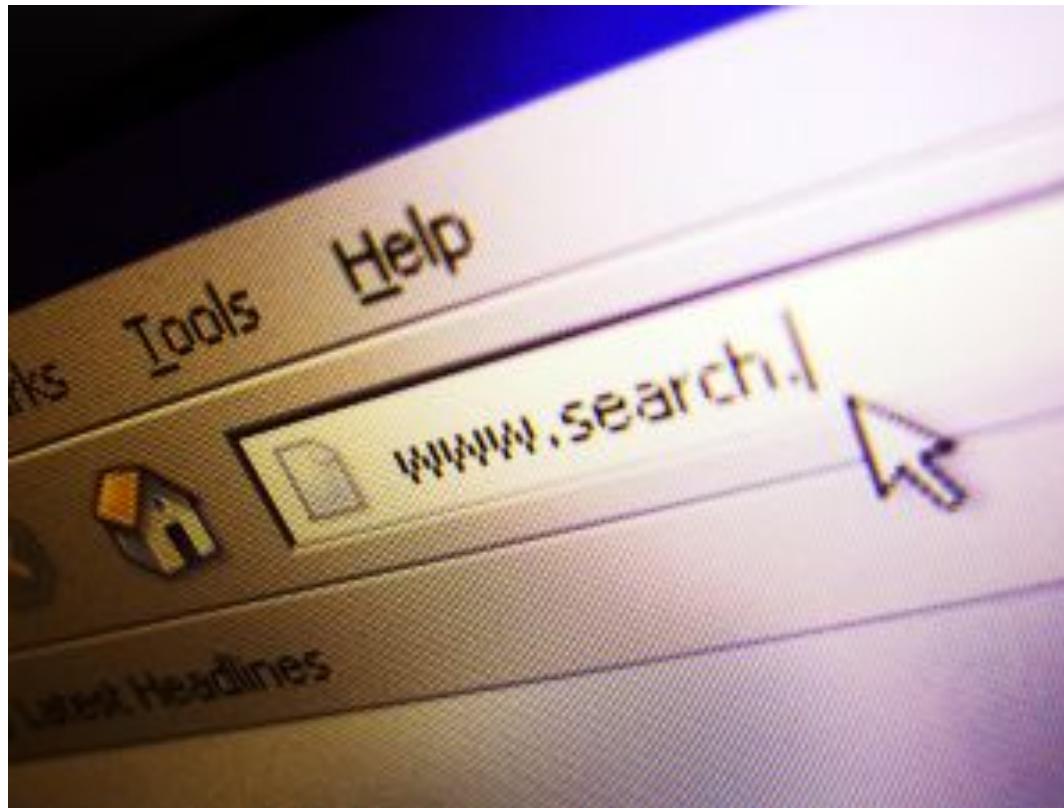


© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

“Obidos”

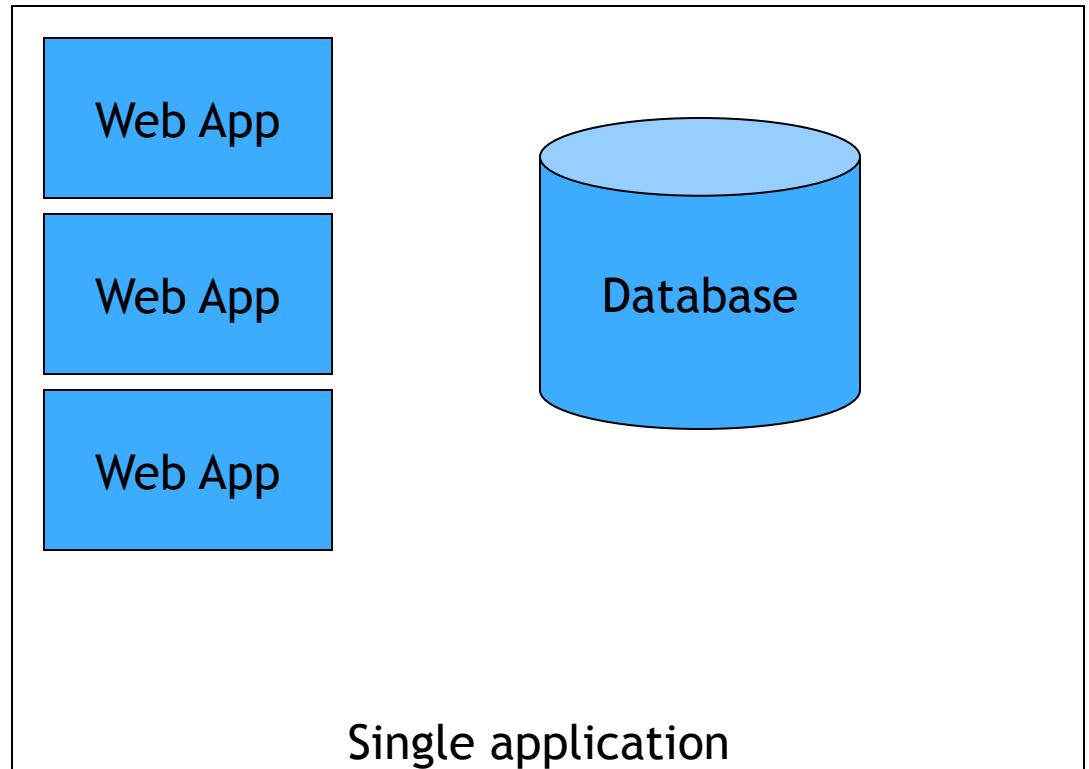
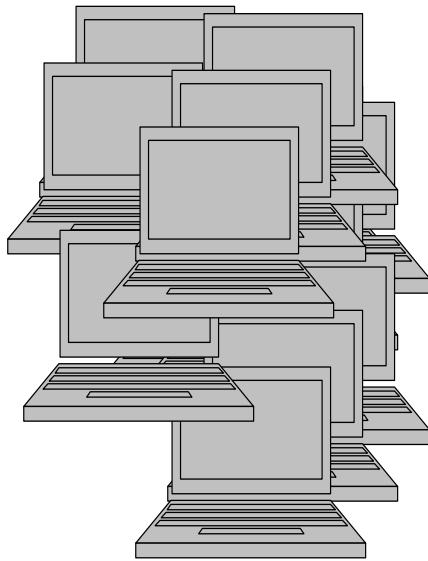


But it was Successful!

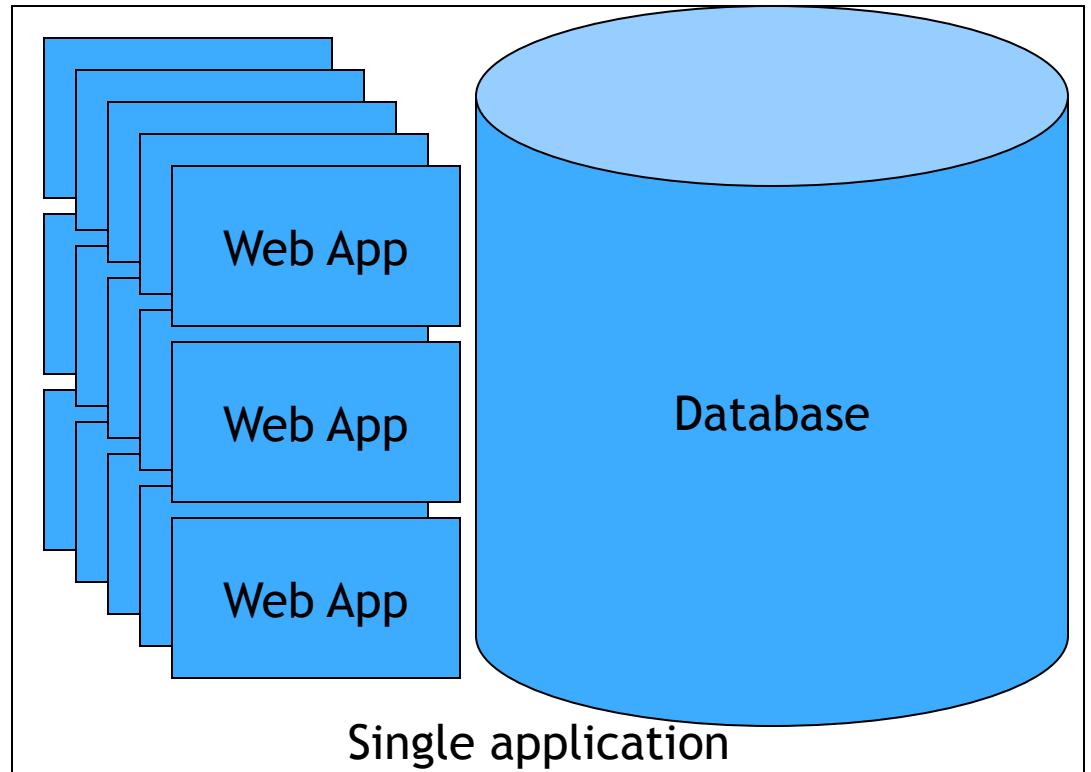
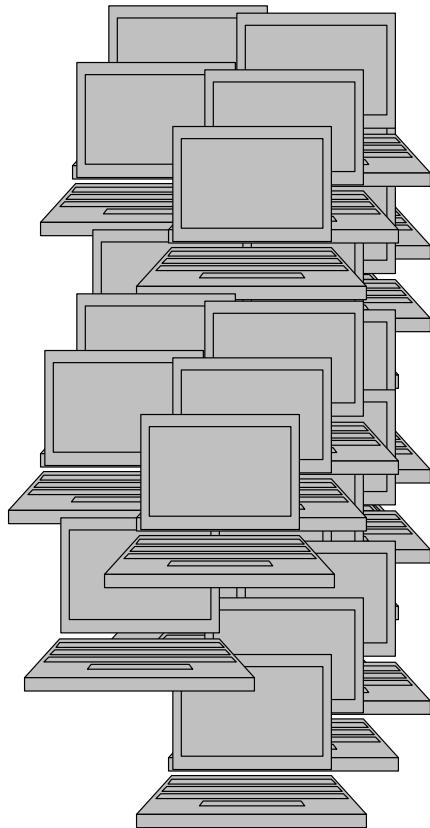


© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Internet Scale Up



... to bursting point



Problems

- Too many complex pieces of software in a single system
- No evolution possible
- Need to scale independently
 - Parts sharing resources with other unknown code paths
- No isolation
- No clear *ownership*



Database scaling

- Databases a shared resource
- Hard to scale-out
- Front-end and backend shared by
 - Too many teams
 - Too many processes



A new model

- In 2001 decided on a new approach
- SOA based – even before the term was in common usage
- Encapsulating the data with the business logic that operates on the data
- Only access through a published service interface
- No direct database access is allowed from outside the service
- No data sharing among the services.



Growth

- Amazon services in the hundreds
- A typical visit to the homepage may include calls to 100 services
- Caching reduces the actual network traffic
- Fully distributed, decentralized
- The web servers are just one client into the service fabric



Matched by business growth

- Amazon is supporting many new businesses
- Books, CDs, Electronics, Toys, Tools and Hardware,...
- Plus millions of independent retailers sharing the Amazon platform



Lessons learnt

- **Isolation**
 - Service Orientation promotes ownership and control
- **Scalability**
 - By preventing direct database access, can scale the services without affecting clients
- **Need a common service-access mechanism**
 - Aggregation
 - Routing
 - Tracking



Organization

- “Each service has a team associated with it, and that team is completely responsible for the service—from scoping out the functionality, to architecting it, to building it, and operating it... **You build it, you run it**” Werner Vogels, CTO, Amazon
 - Promotes Customer Focus and Innovation
 - Gives developers direct access to customers
 - And experience of how their code performs





© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Why did SOA evolve?

- Directly came out of XML
 - Understanding the schema and structure of messages
 - Especially within the “fabric” not just at the endpoints
- What’s different?
 - Metadata and third-party usage
 - Policies, Governance, etc
 - Inherent Security



Service



<http://www.flickr.com/photos/yjv>



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Tightly coupled

- Tightly coupled systems have significant problems:
 - Errors, delays and downtime spread through the system
 - The resilience of the whole system is based on the weakest part
 - Cost of upgrading or migrating spreads
 - Hard to evaluate the useful parts from the dead weight

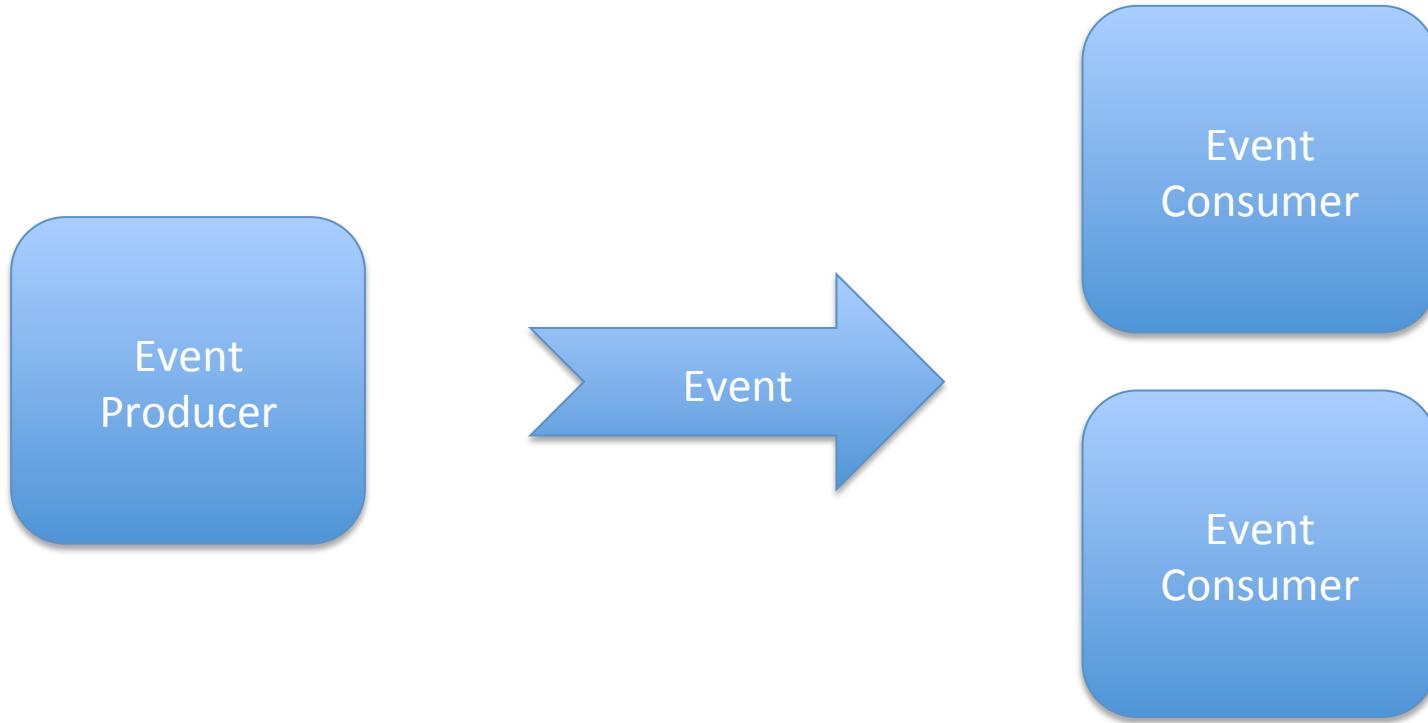


Loose coupling



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Event Driven Architecture



“Event Driven Architecture”

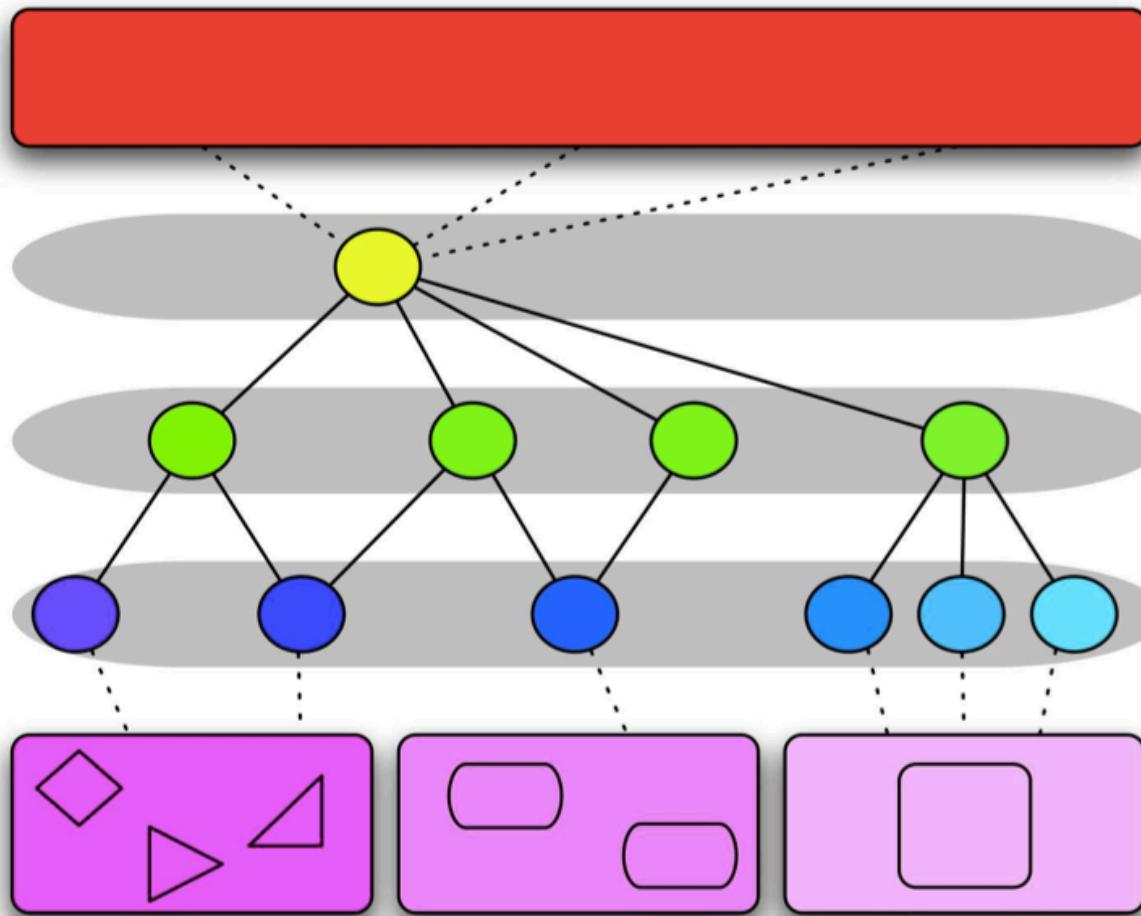
- Actually how Apache (and WSO2) work(s)
 - Mailing lists = topics
- Can be layered with reliable delivery
- Used a lot in high-volume logging, trading environments, fraud detection, etc
- *Requires a very different mindset*



EDA



SOA



business processes

orchestration service layer

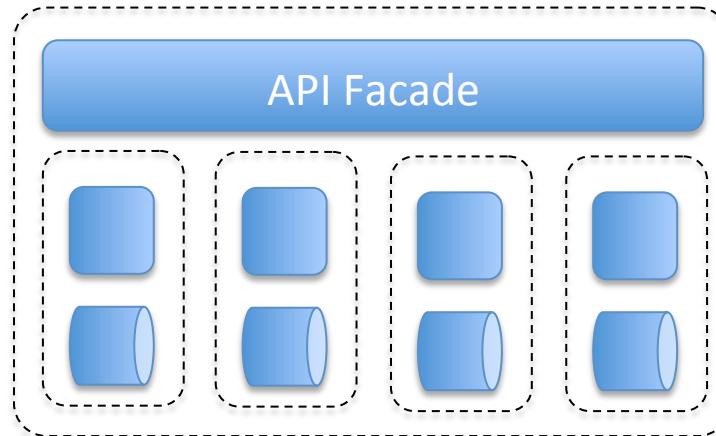
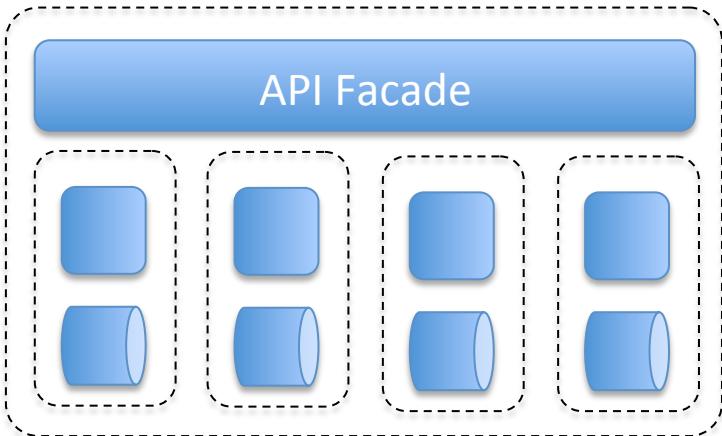
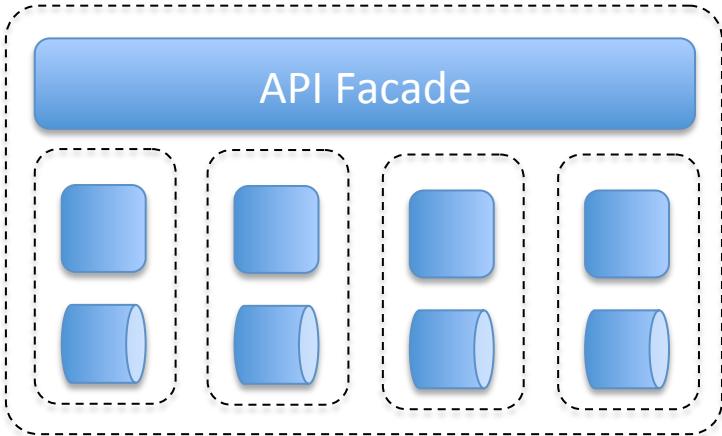
business service layer

application service layer

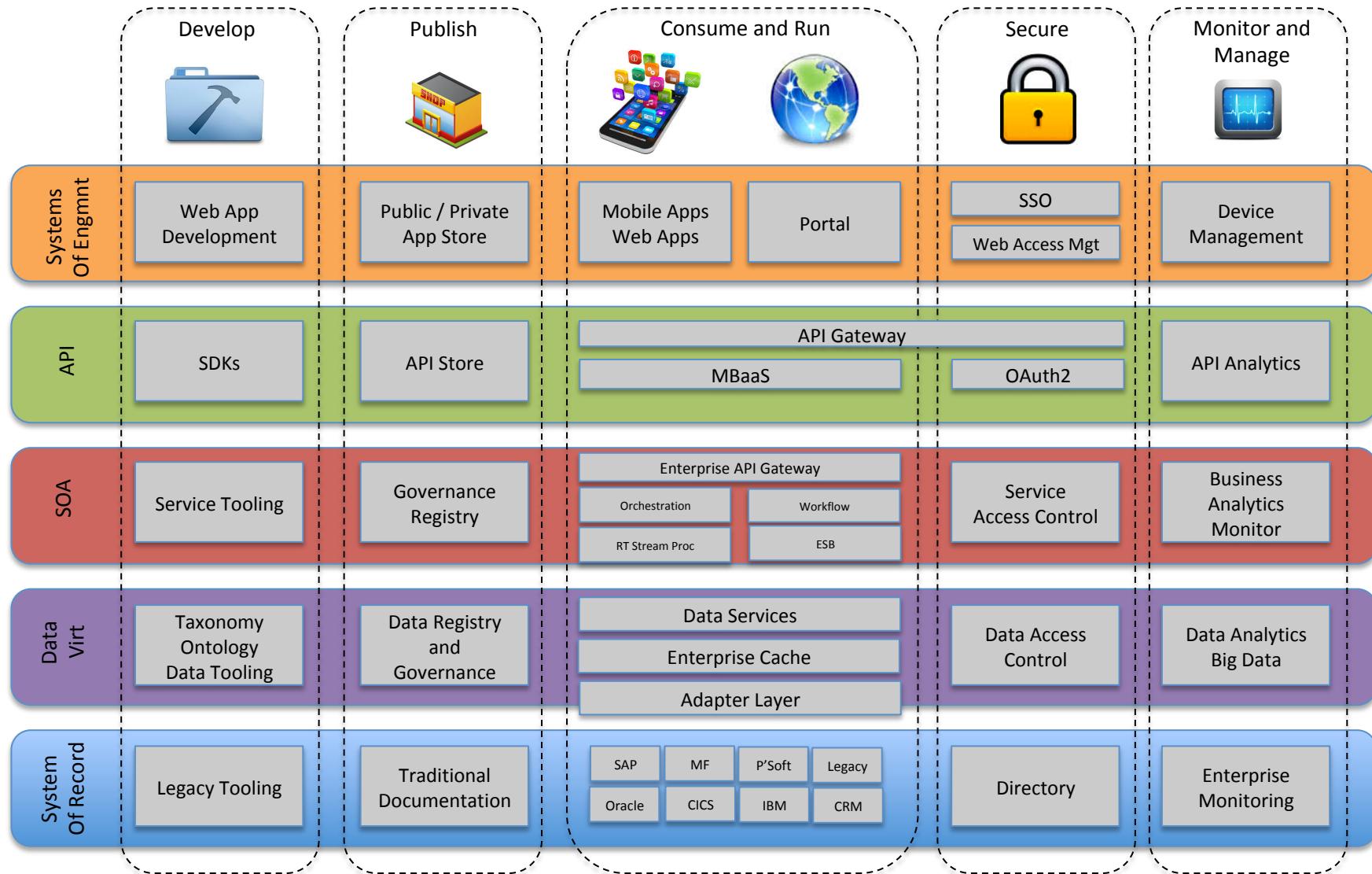
application layer



Micro and Macro Services



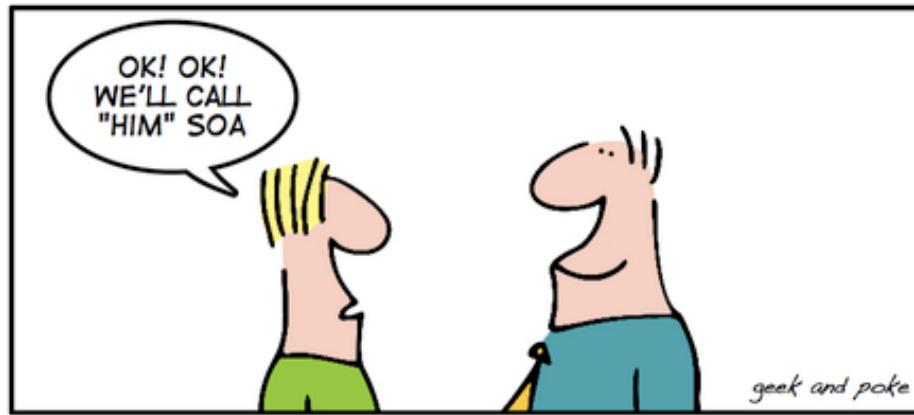
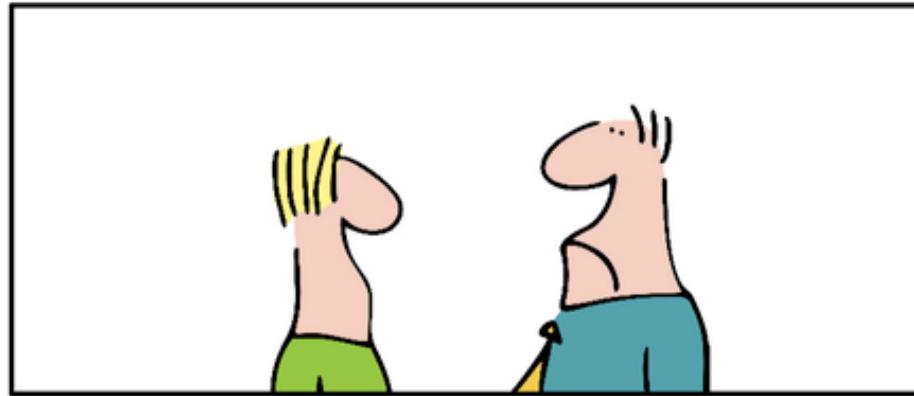
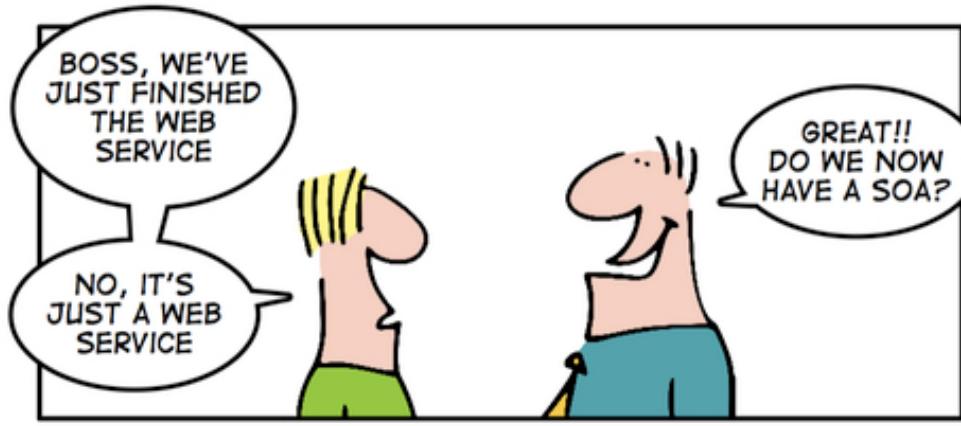
End-to-End Architecture and Functional Requirements for a Connected Business



Web Services != SOA

- There are SOA's that don't use Web Services
 - We will see some in the case studies...
- There are systems that use Web Services that are not Service Oriented
 - e.g. Taking an existing text/TCP model and tunnelling over SOAP





HOW TO GET A SOA



© Paul Fremantle 2C
Attribution-NonCom
See <http://creativecommons.org/licenses/by-nc-sa/2.0/>

Business to Government



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>



IT- og Telestyrelsen
Ministeriet for Videnskab
Teknologi og Udvikling

OIO SOI



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

OIO SOI

- Danish Government wanted to simplify electronic business
 - Especially for Business-to-Government (B2G)
- Potential savings of 630m Euros by digitalizing business
- Requirements
 - Reliable delivery
 - Secure – encrypted and signed messages
 - Support small businesses



OIO SOI

- **Several aspects**
 - A registry for service lookup
 - A profile of transport protocols
 - Open Source toolkits for Java and .NET
 - A reference implementation of a message handler
 - A legal framework
- **Some existing framework**
 - A nationwide digital certificate framework
 - A standard XML syntax for invoices and orders (UBL2)



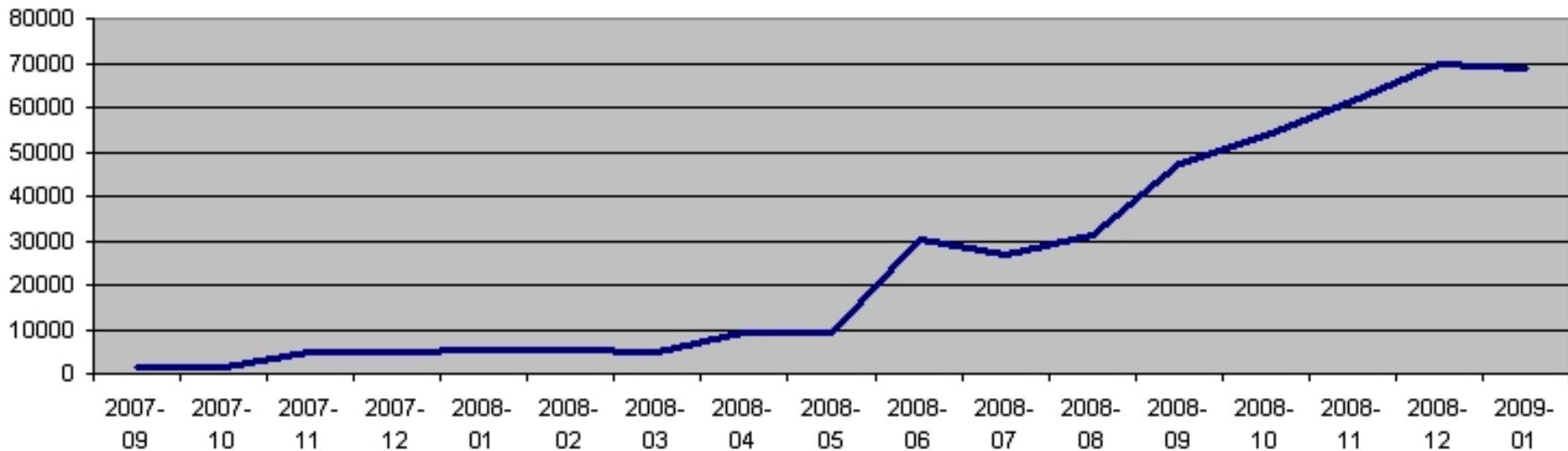
Logical architecture

- This is logically a complete peer-to-peer architecture
 - With only a central registry
- Any company can talk to any other company
- Even those with only mail accounts
- Cannot track all the requests!



Results

Documents via RASP



18,500 companies sending invoices via RASP

Mandatory to send invoices to all government agencies

Scanning companies and a web gateway allow bridging



[Home](#)[Finance & Accounting](#)[Employment Services](#)[Procurement](#)[Media & Events](#)[Contact Us](#)**About NHS SBS****Our Services****Working with clients****Working with suppliers****Supplier FAQs****Postal Invoicing****E-Invoicing**- **E-Invoicing FAQs****Notices****Working with GP practices****Careers****Feedback****Business Intelligence****Personal Health Budgets**[Home](#) ▶ [Working with suppliers](#) ▶ [E-Invoicing](#)

E-invoicing

Want to take steps to get paid quicker and submit your invoices in a way that doesn't involve paper, postage or manual handling?

e-Invoicing through Tradeshift is an exciting new business platform that is designed to make life simpler for organisations and we are working in partnership with them to deliver e-Invoicing for suppliers to our NHS clients.

Registering with Tradeshift is quick and simple and the benefits are immediate. If you are an SME or low volume supplier then the web-based portal at <http://www.tradeshift.com/supplier/nhs-sbs/> is likely to be the best solution, offering immediate access to submit and review invoices wherever you are.

If you submit a higher volume of invoices, you may wish to consider integrating your existing invoicing software with the Tradeshift system. This is a simple process, with support available, to help you see just how beneficial e-Invoicing could be for your business.

Why e-invoicing?

- **Easy setup** - it's quick and simple to get started and just as simple to use
- **Free to use** - no setup fee, transaction fees or service charges
- **Improved communication** - track the status of your invoices and run key reports
- **Instant validation** - take advantage of 15 pre-submission checks to ensure your invoice is right first time

Contact Us

T: 0303 123 1177

E: SBs-W.Payables@nhs.net**Share this**

- Delicious
- Dig It
- Facebook
- LinkedIn
- Twitter

June 2016: Tradeshift raise \$75m



WHAT IS TRADESHIFT? SOLUTIONS BY ROLES RESOURCES FREE INVOICING

ENGLISH ▾

SIGN UP LOG IN

REQUEST DEMO

Innovate the way you buy. |

Procurement, payables, supplier management—and the only open network that's simple for employees, free for suppliers, and agile enough for you.

Watch our story

Let's get started

AIRFRANCE KLM



DHL

CBRE



ZURICH

NHS
Shared Business Services



TRADESHIFT OVERVIEW

Everything you need to know about what we do. [Learn more »](#)



WHY WE'RE DIFFERENT

"You call that an open network? Seriously..." [Read me »](#)



SPEND MATTERS REVIEW

Jason Busch examines our e-procurement tool. [Download »](#)

Netflix - Watch TV Programs

https://signup.netflix.com

Offline Mail Inbox (124,820) - pa WSO2 WSO2, Inc. - Calenda + bitmark Shorten with bit.ly Gmail - Inbox (720)

Questions? Call 0800 096 6379 - 24

Buy / Redeem Gift Member

NETFLIX

1 MONTH FREE TRIAL

Watch TV programmes & films anytime, anywhere. Only £5.99 a month.

Start Your Free Month



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Netflix

- A REST and Cloud based SOA approach
- Continuous Delivery
- 100% Based in the cloud
- See excellent presentations from Adrian Cockcroft

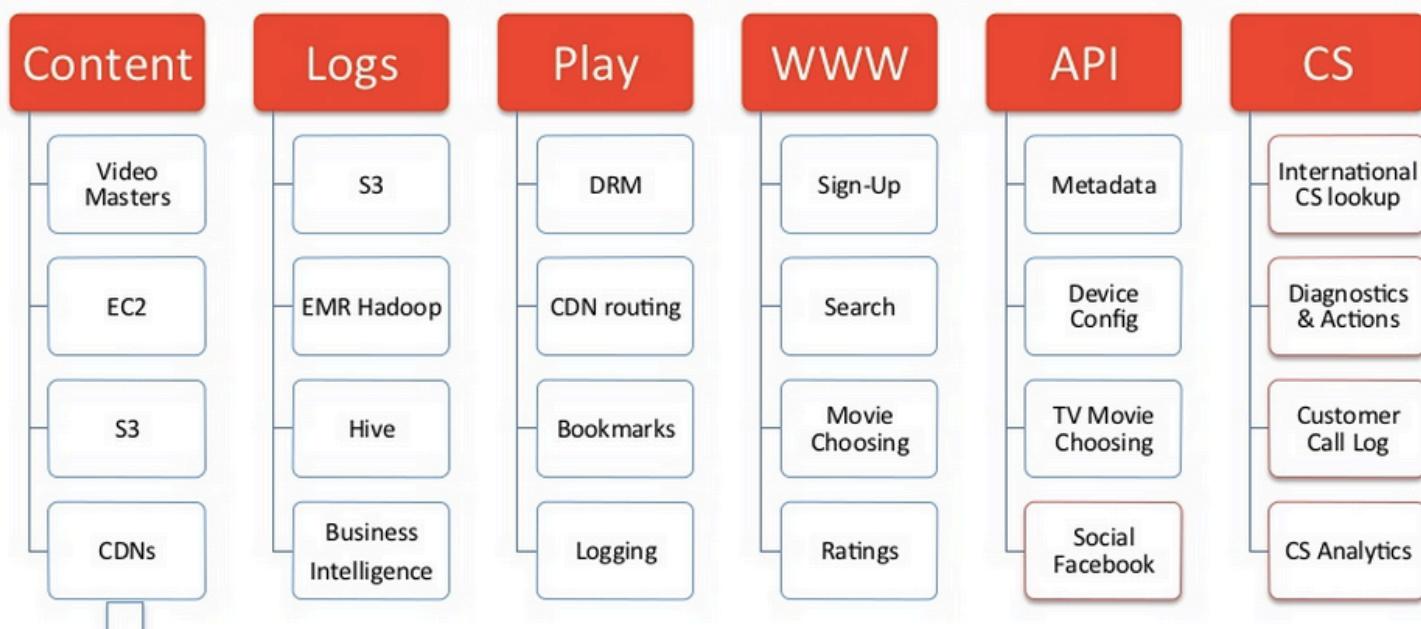
– e.g.

[http://www.slideshare.net/adrianco/
global-netflix-platform](http://www.slideshare.net/adrianco/global-netflix-platform)



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Netflix Deployed on AWS



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Platform Services

- Discovery – service registry for “applications”
- Introspection – Entrypoints
- Cryptex – Dynamic security key management
- Geo – Geographic IP lookup
- Platformservice – Dynamic property configuration
- Localization – manage and lookup local translations
- Evcache – eccentric volatile (mem)cached
- Cassandra – Persistence
- Zookeeper - Coordination
- Various proxies – access to old datacenter stuff



The (in)famous Chaos Monkey

- Randomly kills machines
- Yes, production systems
- Proves that the system is resilient



Twitter Architecture

- Open Sourced their technology:
 - Finagle
 - <http://twitter.github.io/finagle/>
 - Called an RPC system, but completely asynchronous
 - Based on “Services”



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

[http://monkey.org/~marius/talks/ twittersystems/#4](http://monkey.org/~marius/talks/twittersystems/#4)

Late 2012 architecture

Many **open source** components

- Memcache, redis, MySQL, etc.
- Necessarily heterogeneous

Organized around **services**

- Distinct responsibilities
- Isolated from each other
- Distributed computation and data
- RPC between systems

Multiplexing HTTP frontend

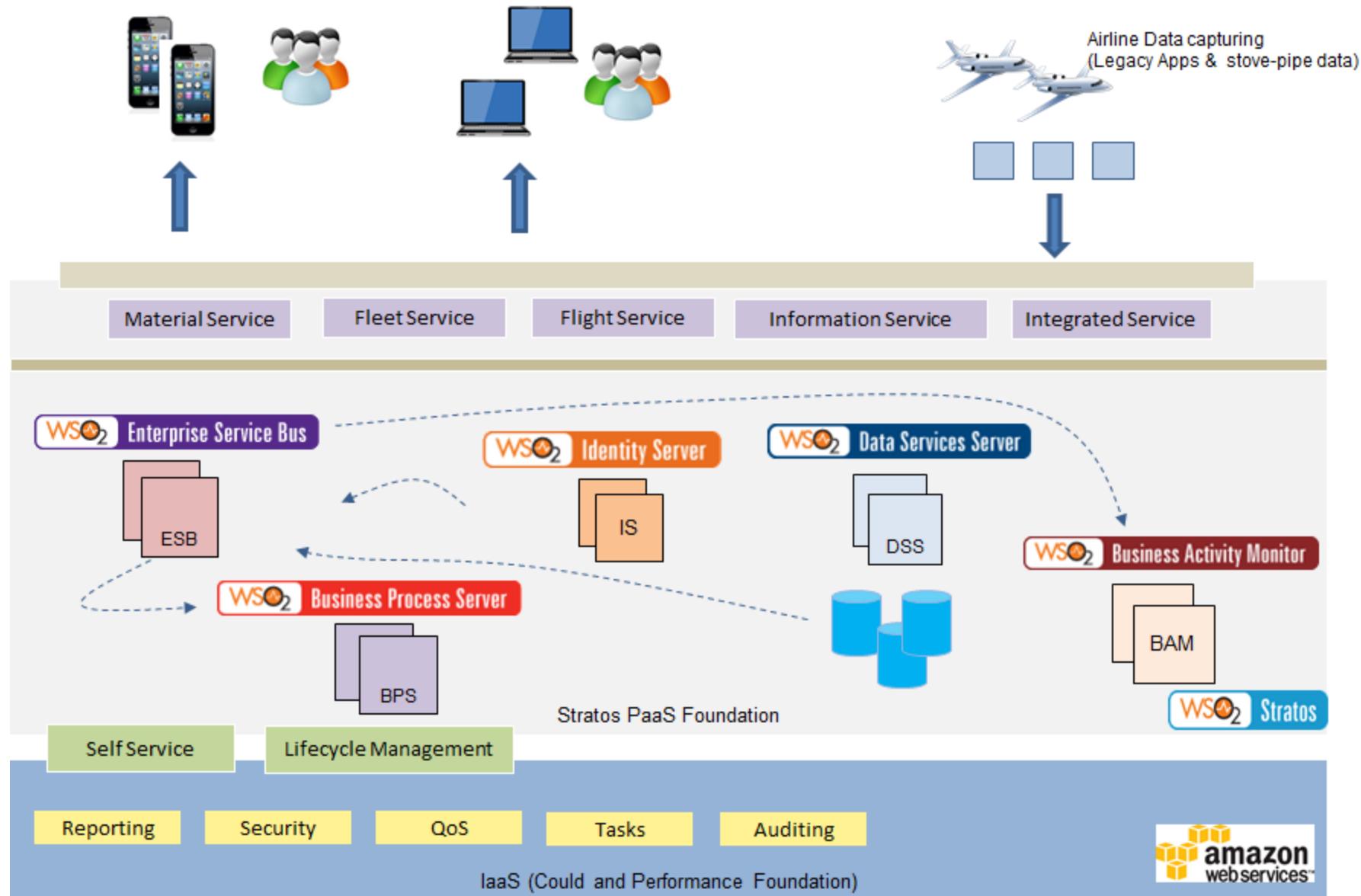
- Crucial for modularity, load balancing



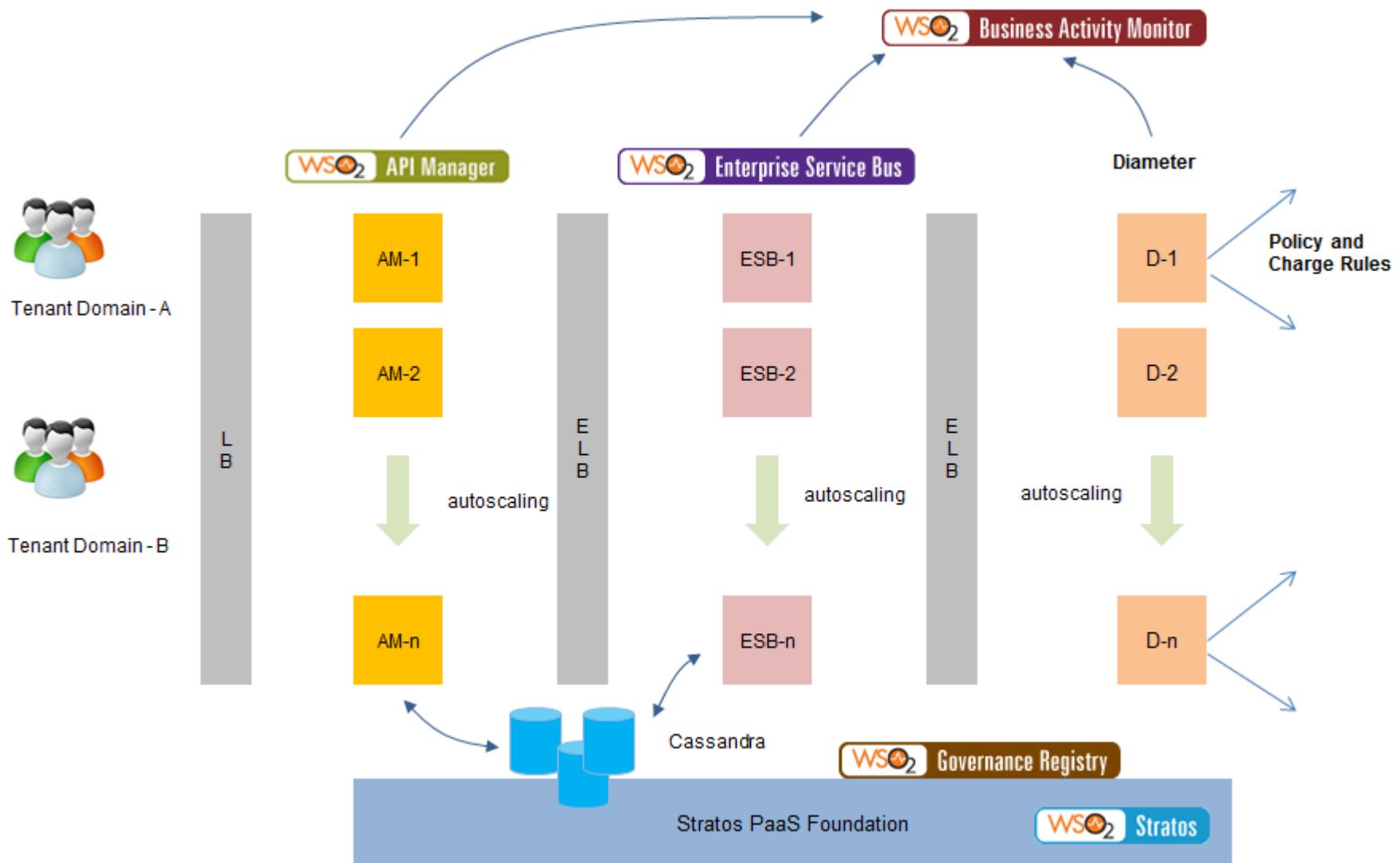
Boeing Digital Airline



Case Study : Boeing - A PaaS based Integration and API ecosystem



Case Study : Multi-tenanted Mobile Orchestration Gateway Platform

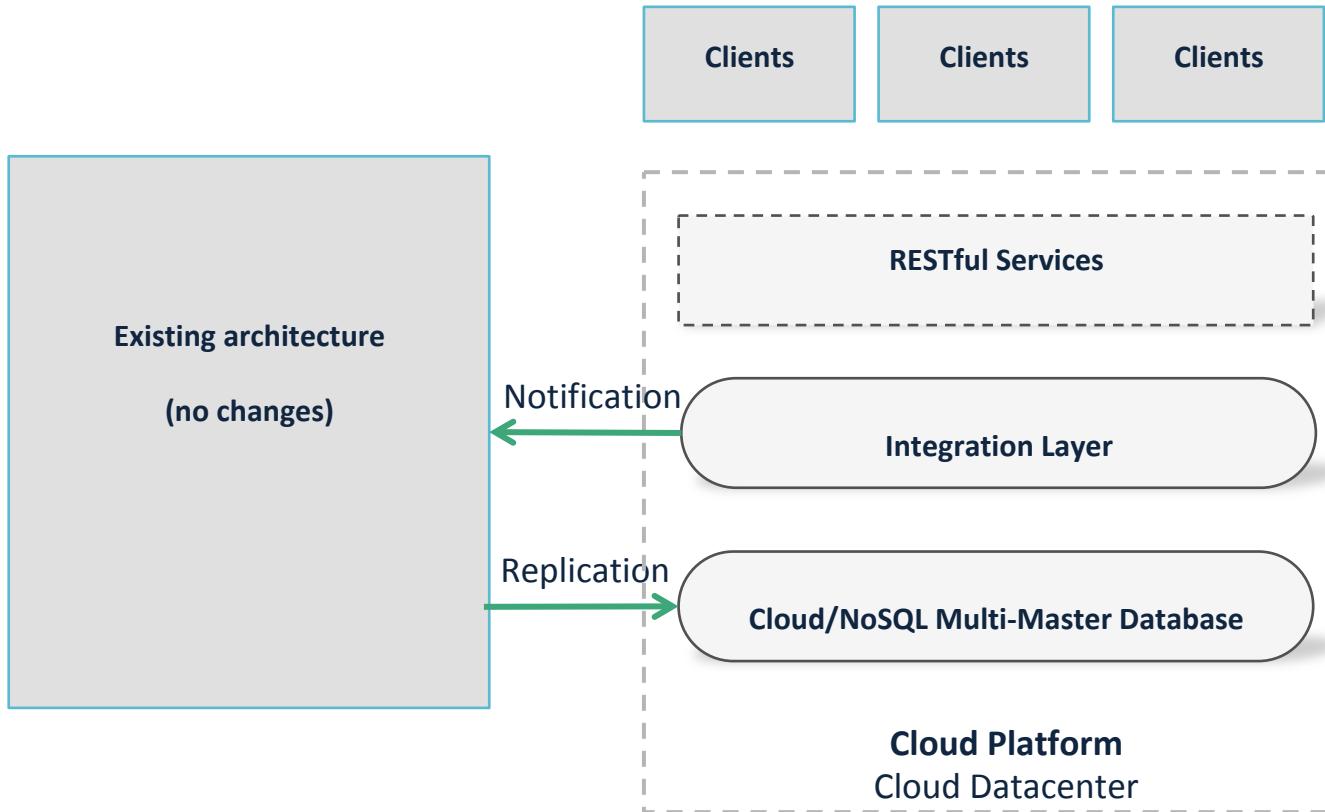


Pay TV company

- Needed to scale up to provide instant pay-as-you-go on mobile devices
- Support Disaster Recovery (DR)
- Elastic Scale e.g. during an important football match



Architecture



Anti-patterns

- Use a full waterfall model
- Don't budget time for integration test
 - Assume that standard coding unit test->integration test will work
- Build unit tests that don't test interoperability
 - E.g. Simulate XML request/response inside the calling system rather than calling a remote system
- Wait until all the systems are ready before starting any integration test
 - A delay to one system will hold up testing all the others
- Don't bother with continuous build and test
 - Even better build by hand
 - **Even better** test by hand too
- Have a nice complex process to hand over from development to test
 - That way each defect will take a long time
- Wait until the project is failing to find out your team doesn't have the skills



Questions?



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>