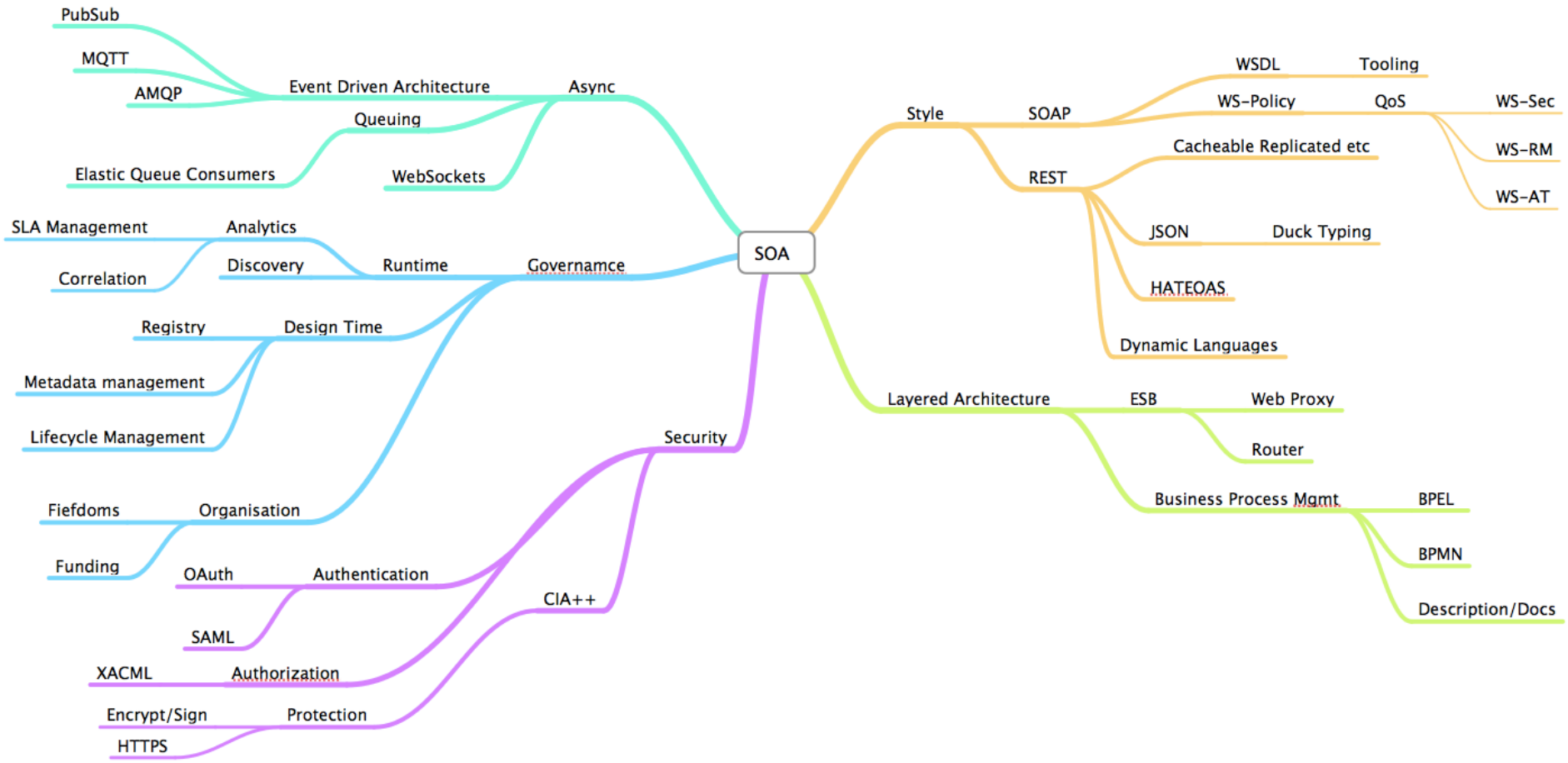
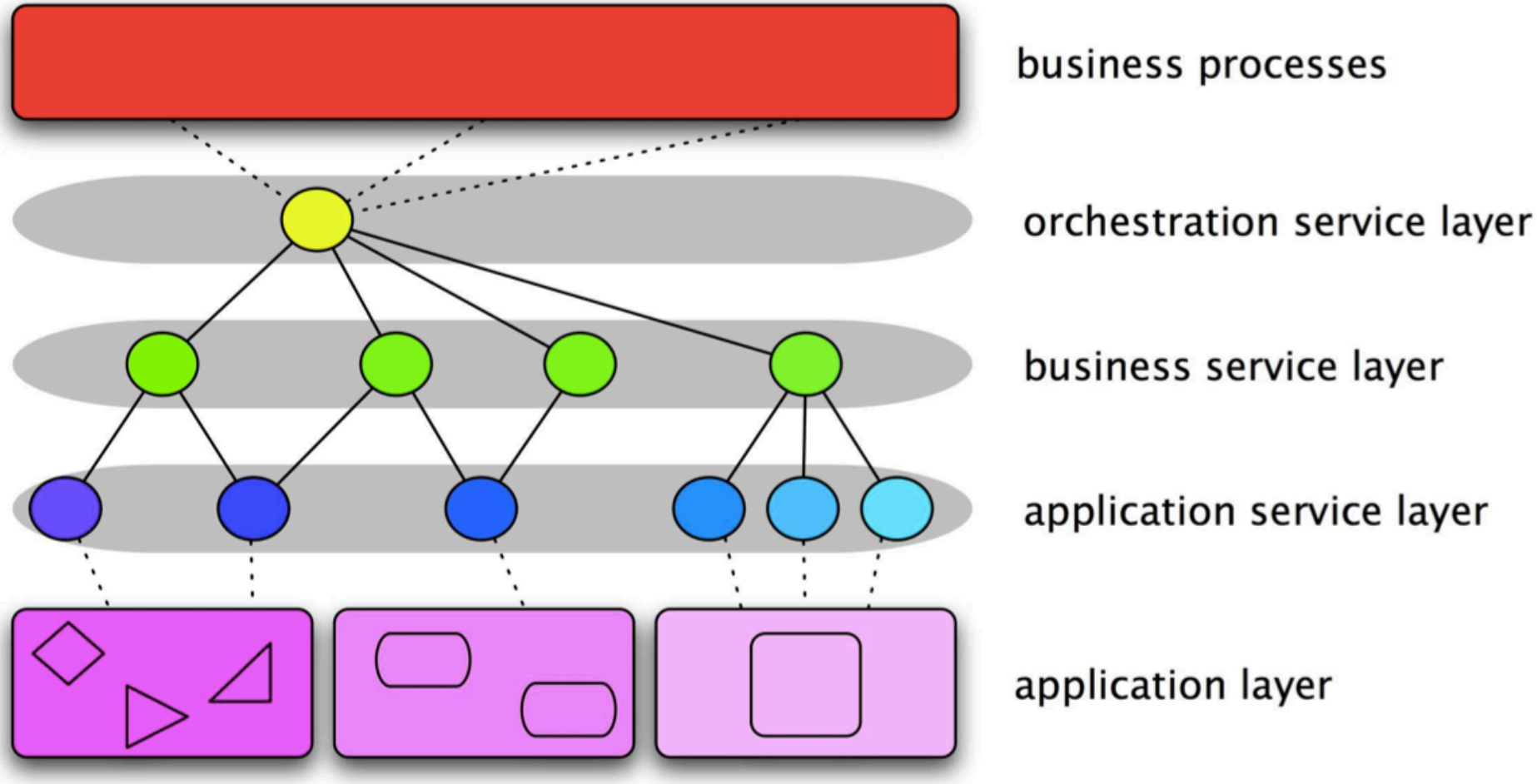
Conclusions, Evolution of SOA, Futures

Oxford University
Software Engineering
Programme
May 2017



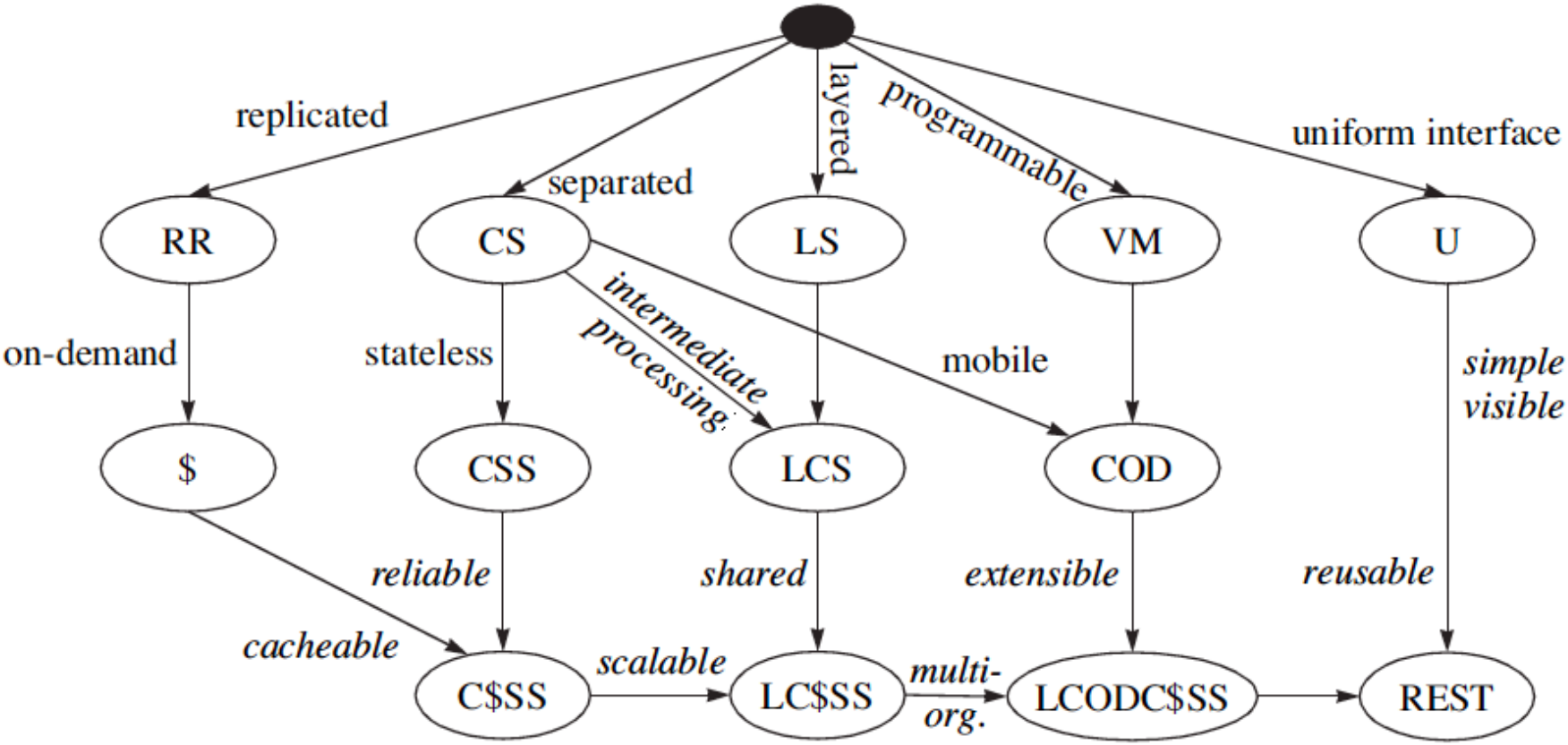
SOA



SOAP and WS-*

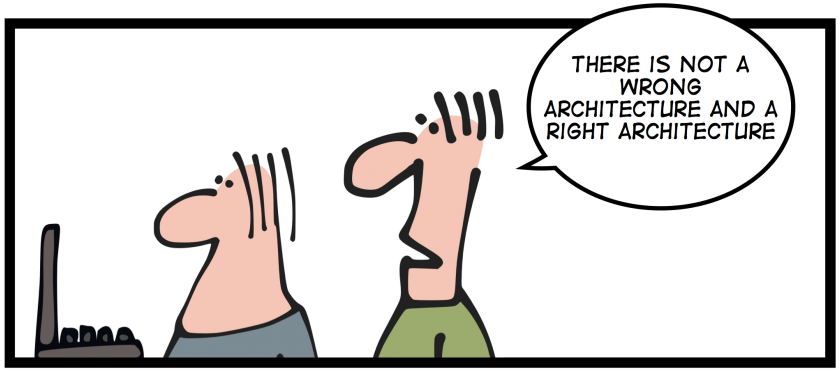
- Composable
  - WS-Security, ReliableMessaging, etc
- Transport independent
- Tooling
- Schemas and WSDLs
  - Governance

REST

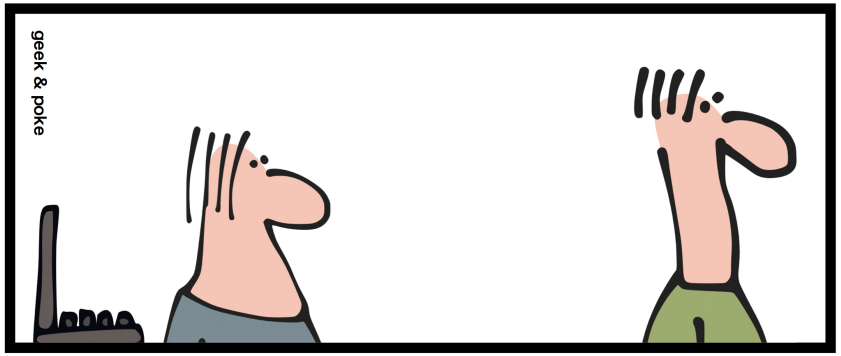


HATEOAS

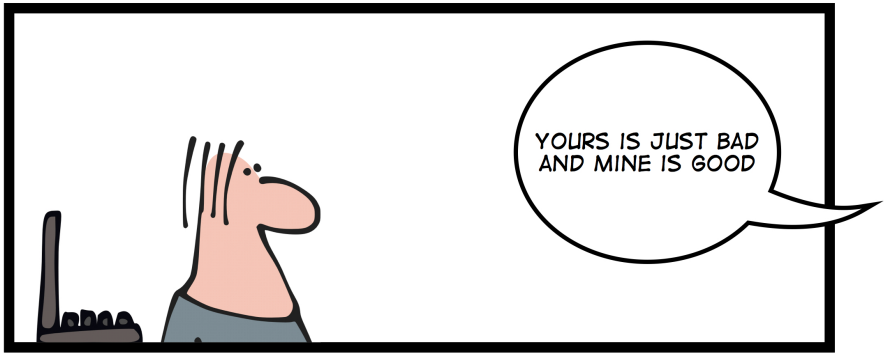
```
201 Created
Location: http://starbucks.example.org/order/1234
Content-Type: application/xml
Content-Length: ...

<order xmlns="http://starbucks.example.org/">
  <drink>latte</drink>
  <cost>3.00</cost>
  <next xmlns="http://example.org/state-machine"
    rel="http://starbucks.example.org/payment"
    uri="https://starbucks.example.com/payment/order/1234"
    type="application/xml"/>
</order>
```



IT ARCHITECTURE IS NOT ALWAYS SIMPLE



FORTUNATELY...



... MOST OF THE TIME IT IS

Design Governance

- Interfacing SOA into the build/test/ production
- Encouraging Service Re-Use
- Lifecycle and Dependency Management
- Notification

Runtime Governance

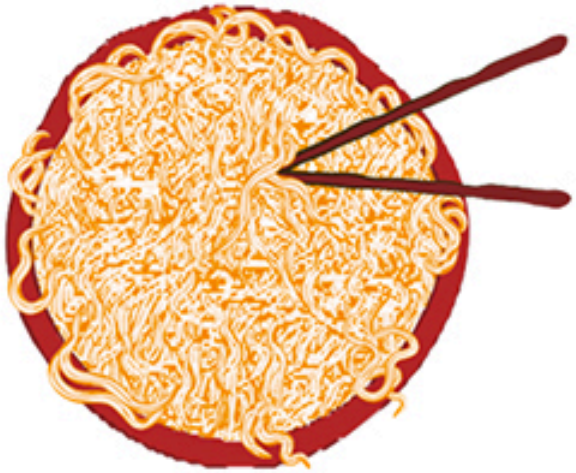
- Monitoring
- SLA management
- Correlation of activities into flows
- How do you maintain a running application when it depends on 10s, 100s or 1000s of remote services?

Services vs APIs

- Focus on the consumer
  - Self-signup and subscription
  - Tracking and usage
  - Developer portals and ease-of-use
  - Monetization

1990s and earlier

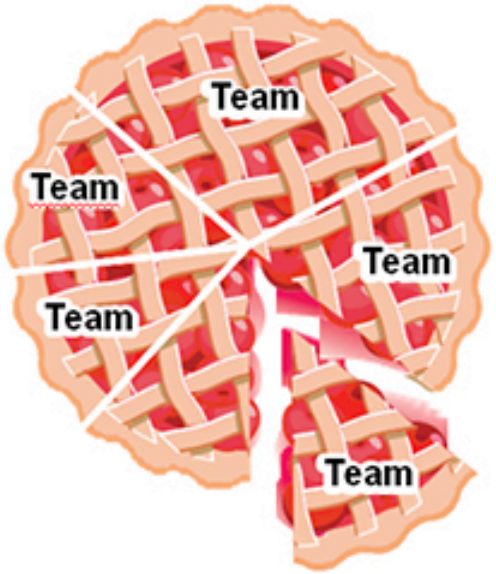
Pre-SOA (monolithic)
Tight coupling



For a monolith to change, all must agree on each change. Each change has unanticipated effects requiring careful testing beforehand.

2000s

Traditional SOA
Looser coupling



Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

2010s

Microservices
Decoupled

Developers can create and activate new microservices without prior coordination with others. Their adherence to MSA principles makes continuous delivery of new or modified services possible.

Source: PwCw

1990s and earlier | **2000s** | **2010s**

Coupling

Pre-SOA (monolithic)
Tight coupling

Traditional SOA
Looser coupling

Microservices
Decoupled

Source: PwC

ESBs and Intermediaries

- ESB Patterns
  - Façade
  - Hub
  - Federated
  - Monitoring point
  - Transformation
- ESB vs Registry or both?
  - Or a utopia where every service works directly with every other?

Orchestration and Composition

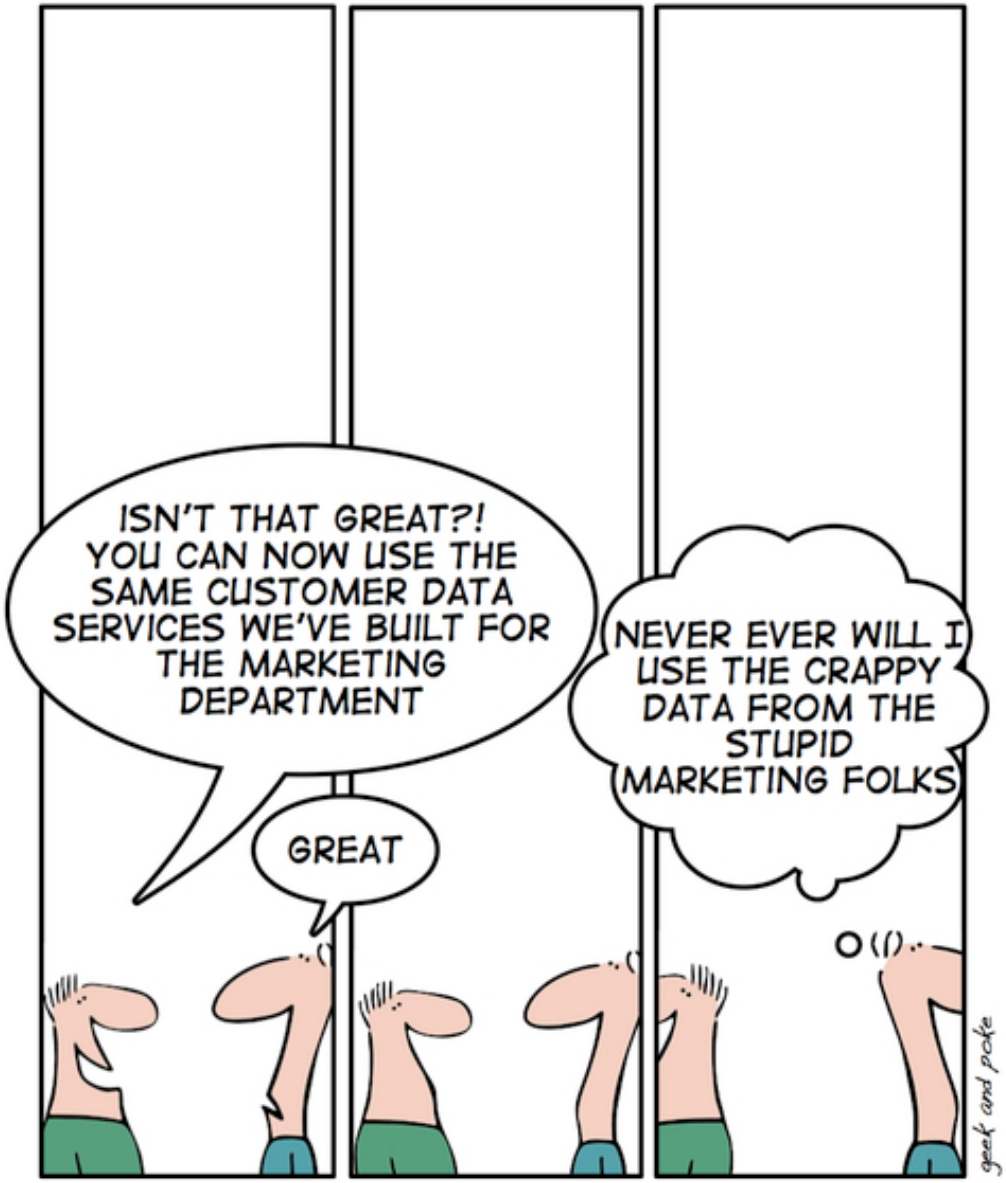
- BPMN, BPEL
- Executable Documentation?
- Visibility and Monitoring

Design Considerations

- Granularity of Services
  - Microservices
  - Monolith First? Microservice First?
- Ensuring that SOA is being used for a good reason:
  - Scale
  - Organizational boundaries
  - Evolvability
- Where to draw the boundaries?
  - Between services
  - Between microservices and services
  - Between ESB and BPM
  - Between organizations
- Are your layers right?

Organizational issues

- Funding models
- Fiefdoms
- Ecosystems / Value Webs
- Shadow IT / Cloud



THE BENEFITS OF A SOA

SOA and Cloud

- SOA is loose-coupling between applications and applications

- Cloud is loose-coupling between applications and infrastructure

What else?

Thanks!

paul@fremantle.org
@pzfreo
LinkedIn/Facebook/Slideshare/
Github