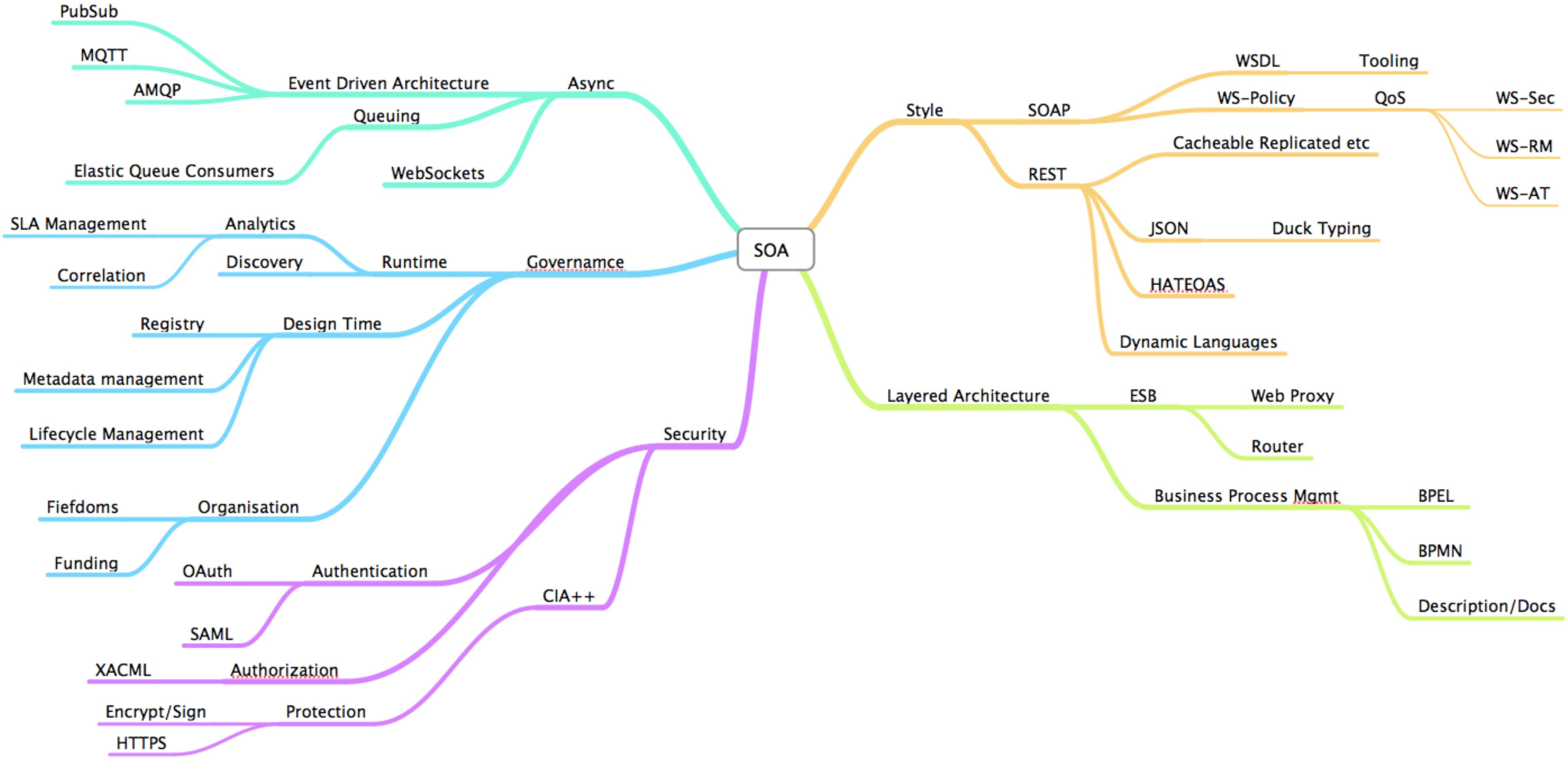
Conclusions, Evolution of SOA, Futures

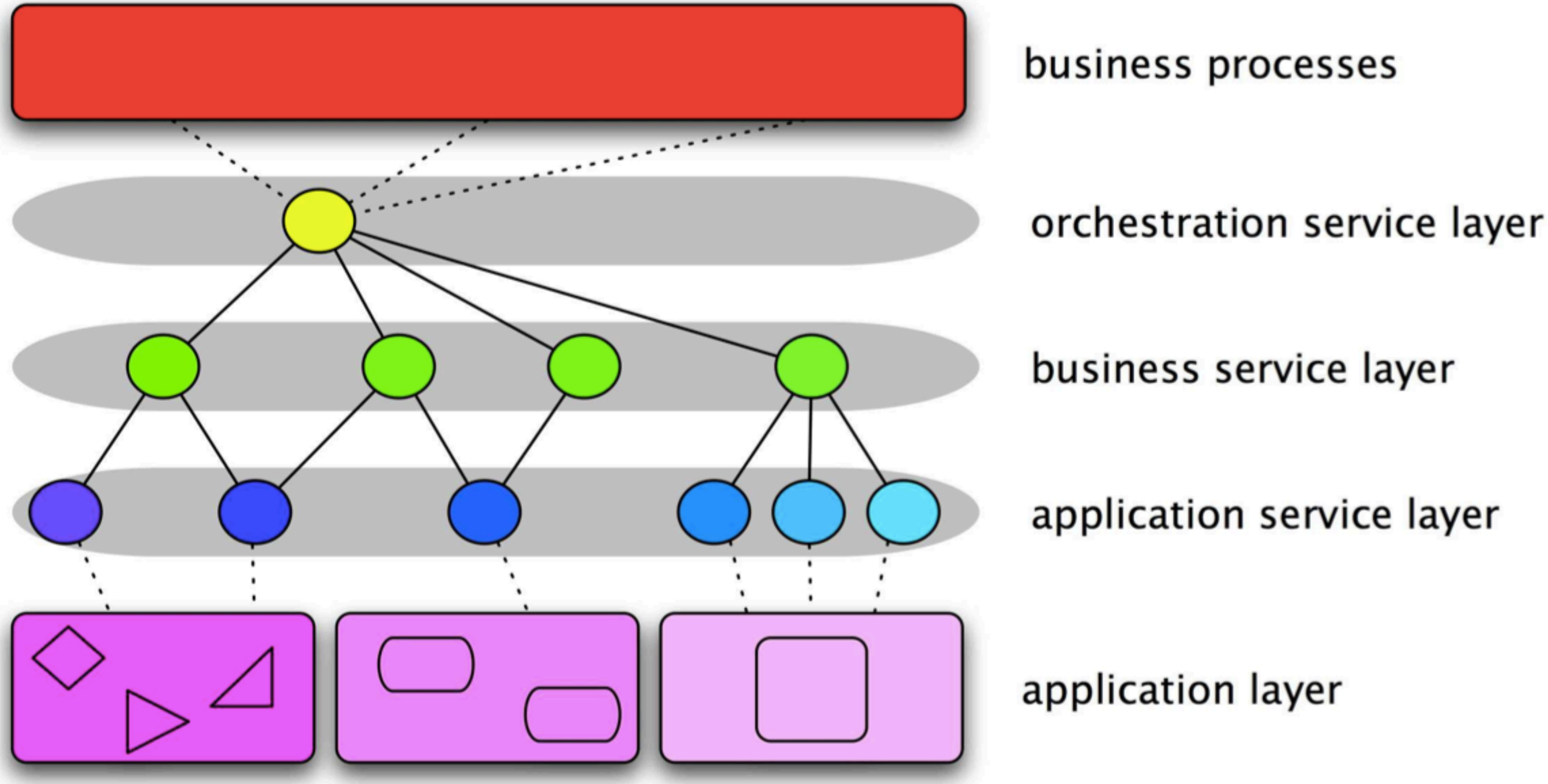
Oxford University
Software Engineering Programme

June 2016



- SOA
  - Async
    - Event Driven Architecture
      - PubSub
      - MQTT
      - AMQP
    - Queuing
      - Elastic Queue Consumers
    - WebSockets
  - Governamce
    - Runtime
      - Analytics
        - SLA Management
        - Correlation
      - Discovery
    - Design Time
      - Registry
      - Metadata management
      - Lifecycle Management
    - Organisation
      - Fiefdoms
      - Funding
  - Security
    - Authentication
      - OAuth
      - SAML
    - Authorization
      - XACML
    - CIA++
      - Protection
        - Encrypt/Sign
        - HTTPS
  - Style
    - SOAP
      - WSDL
        - Tooling
      - WS-Policy
        - QoS
          - WS-Sec
          - WS-RM
          - WS-AT
    - REST
      - Cacheable Replicated etc
      - JSON
        - Duck Typing
      - HATEOAS
      - Dynamic Languages
  - Layered Architecture
    - ESB
      - Web Proxy
      - Router
    - Business Process Mgmt
      - BPEL
      - BPMN
      - Description/Docs

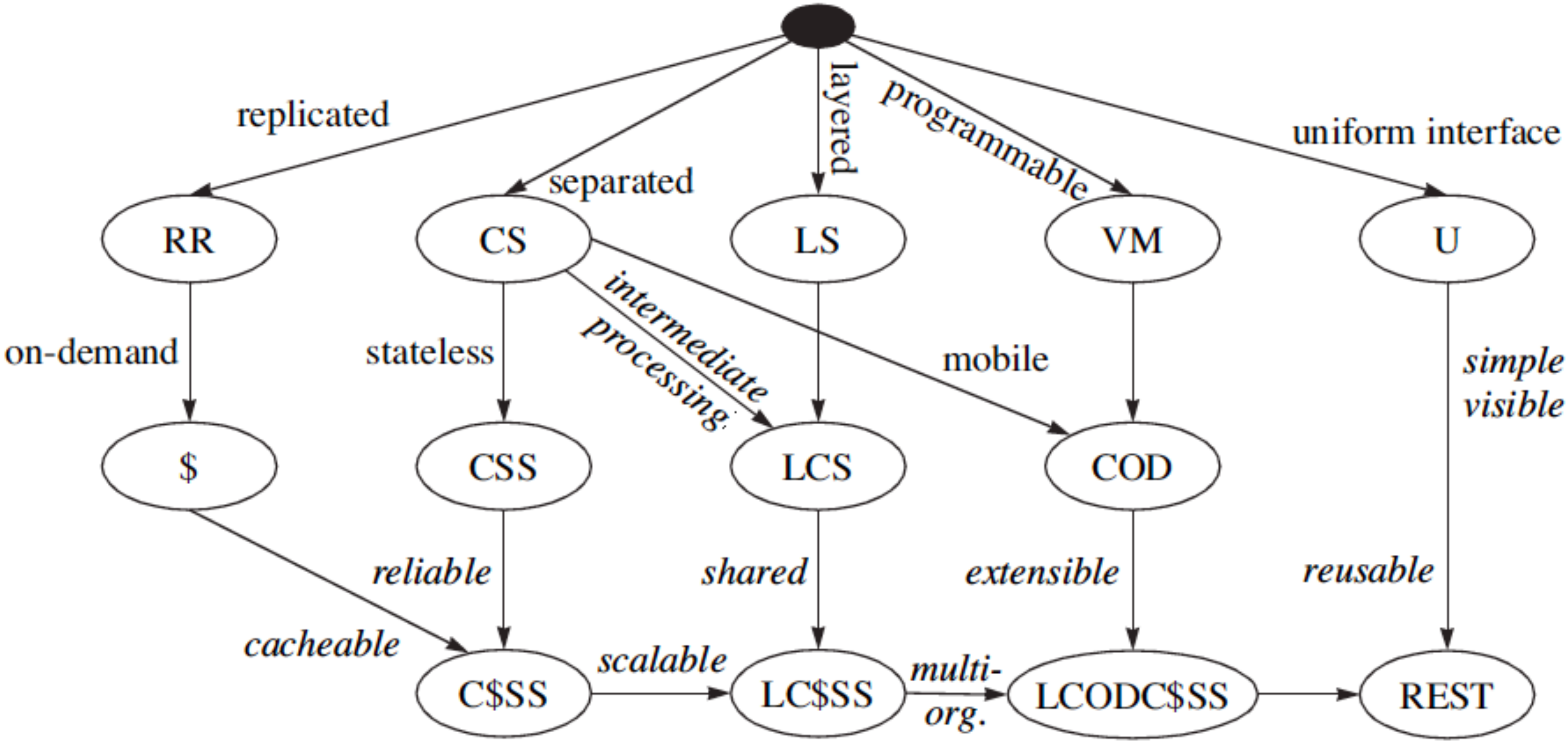
SOA


business processes
orchestration service layer
business service layer
application service layer
application layer

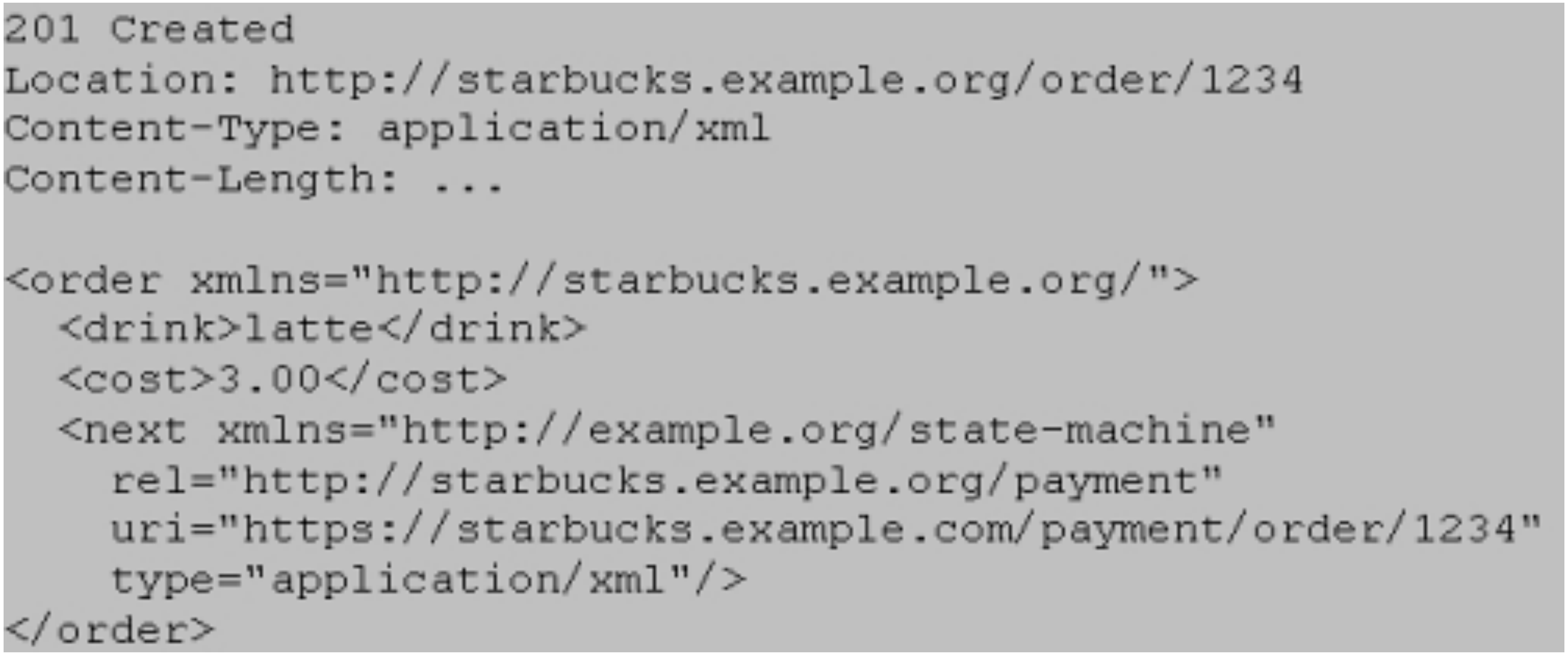
SOAP and WS-*

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

REST

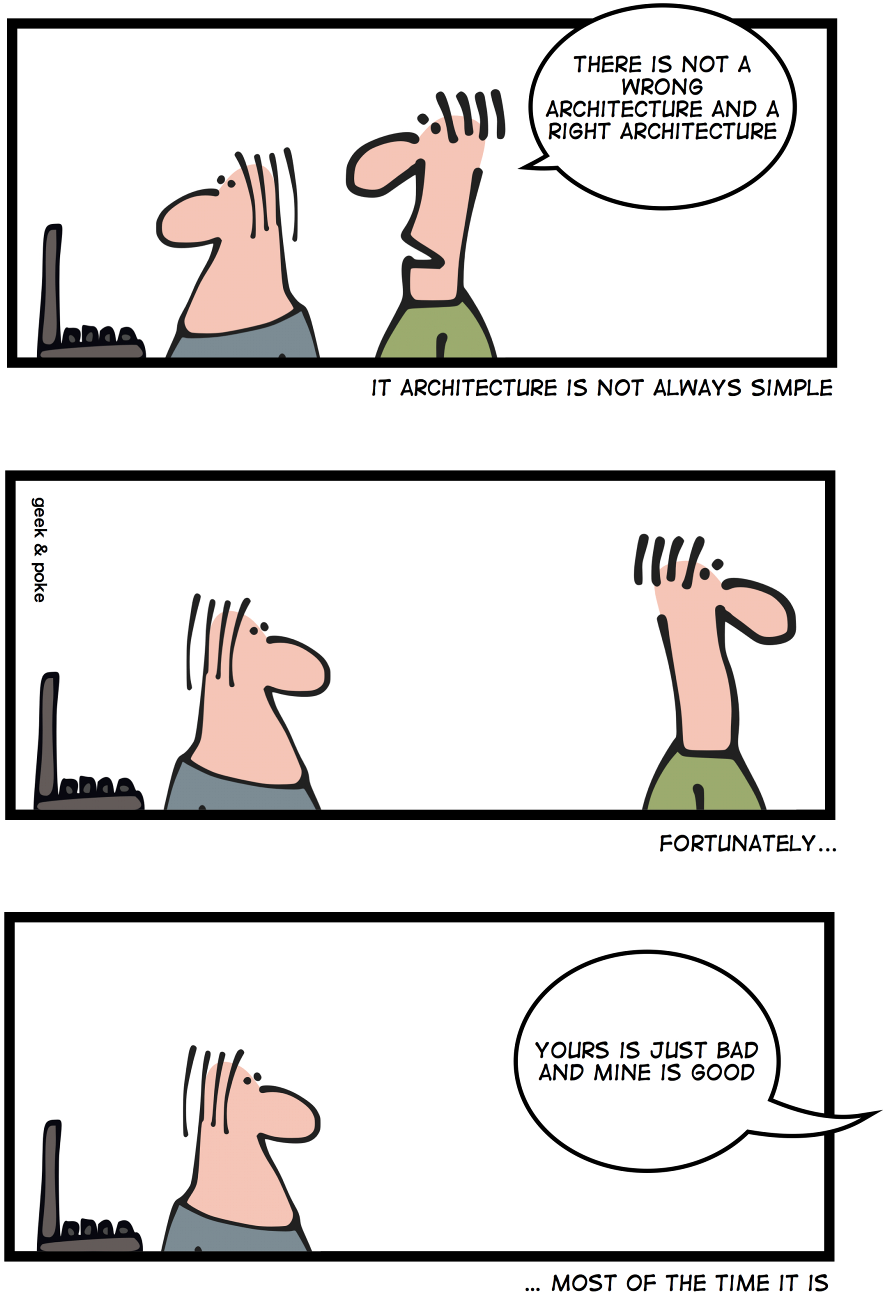

replicated
separated
layered
programmable
uniform interface
RR
CS
LS
VM
U
on-demand
stateless
intermediate processing
mobile
simple visible
$
CSS
LCS
COD
reliable
shared
extensible
reusable
cacheable
scalable
multi-org.
C$SS
LC$SS
LCODC$SS
REST

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>
```


THERE IS NOT A WRONG ARCHITECTURE AND A RIGHT ARCHITECTURE
IT ARCHITECTURE IS NOT ALWAYS SIMPLE
geek & poke
FORTUNATELY...
YOURS IS JUST BAD AND MINE IS GOOD
... MOST OF THE TIME IT IS

Design Governance

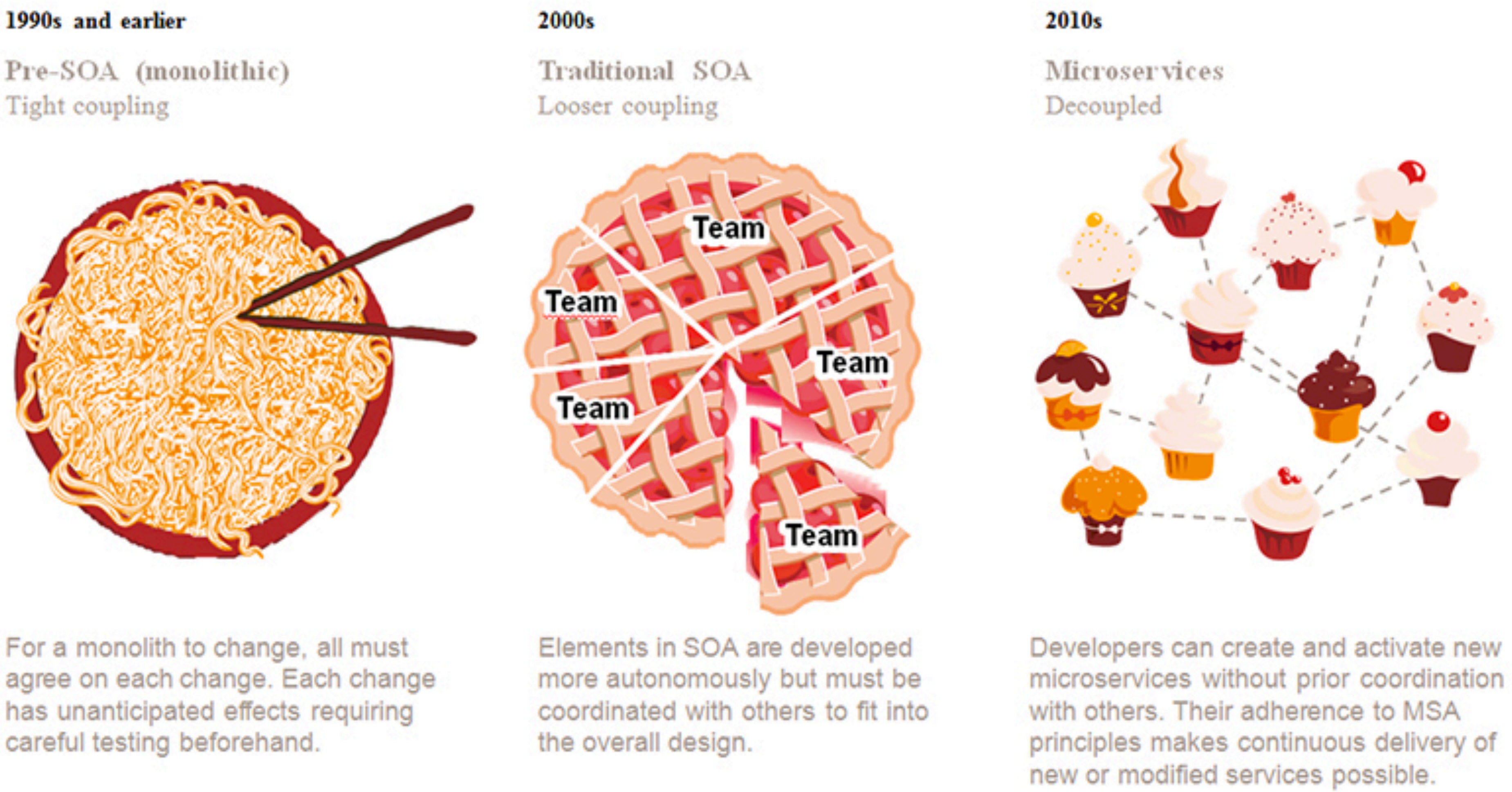
- Interfacing SOA into the build/test/production
- Encouraging Service Re-Use
- Lifecycle and Dependency Management
- Notification

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

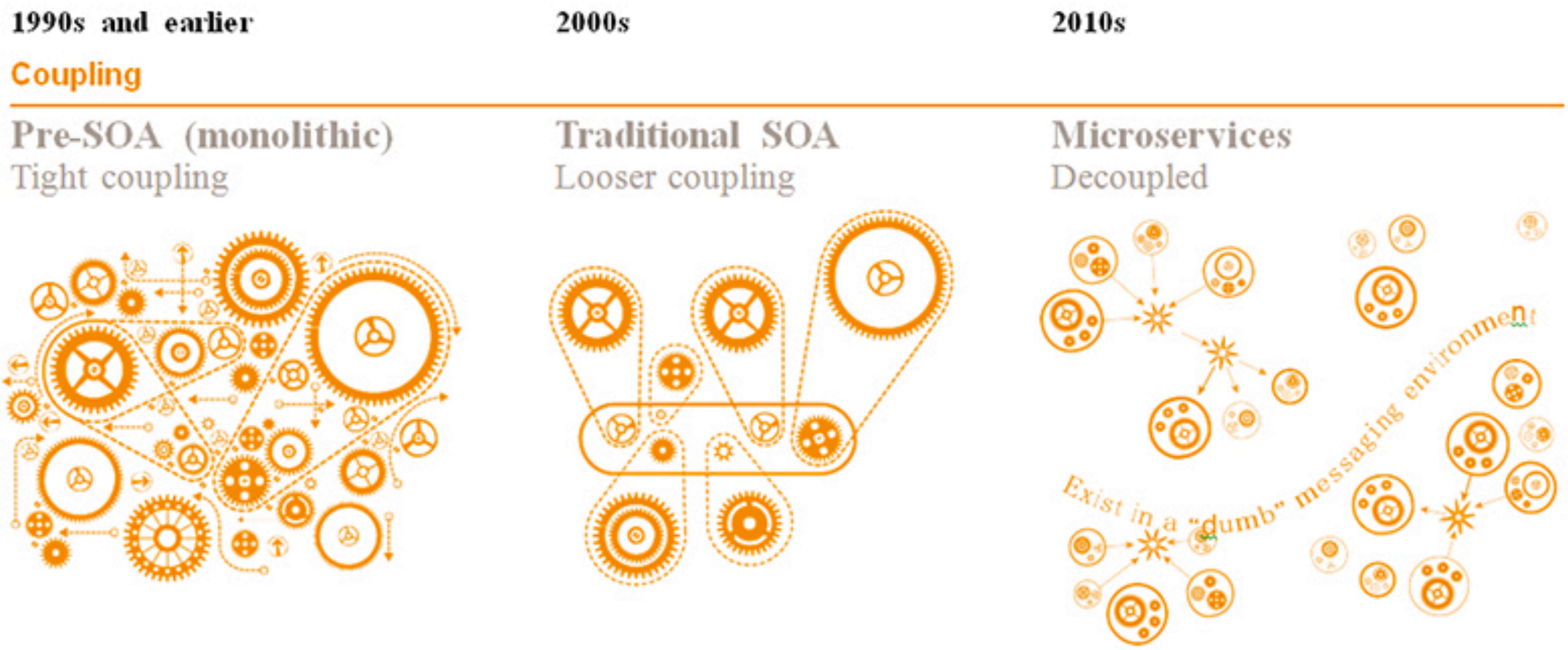
Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

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

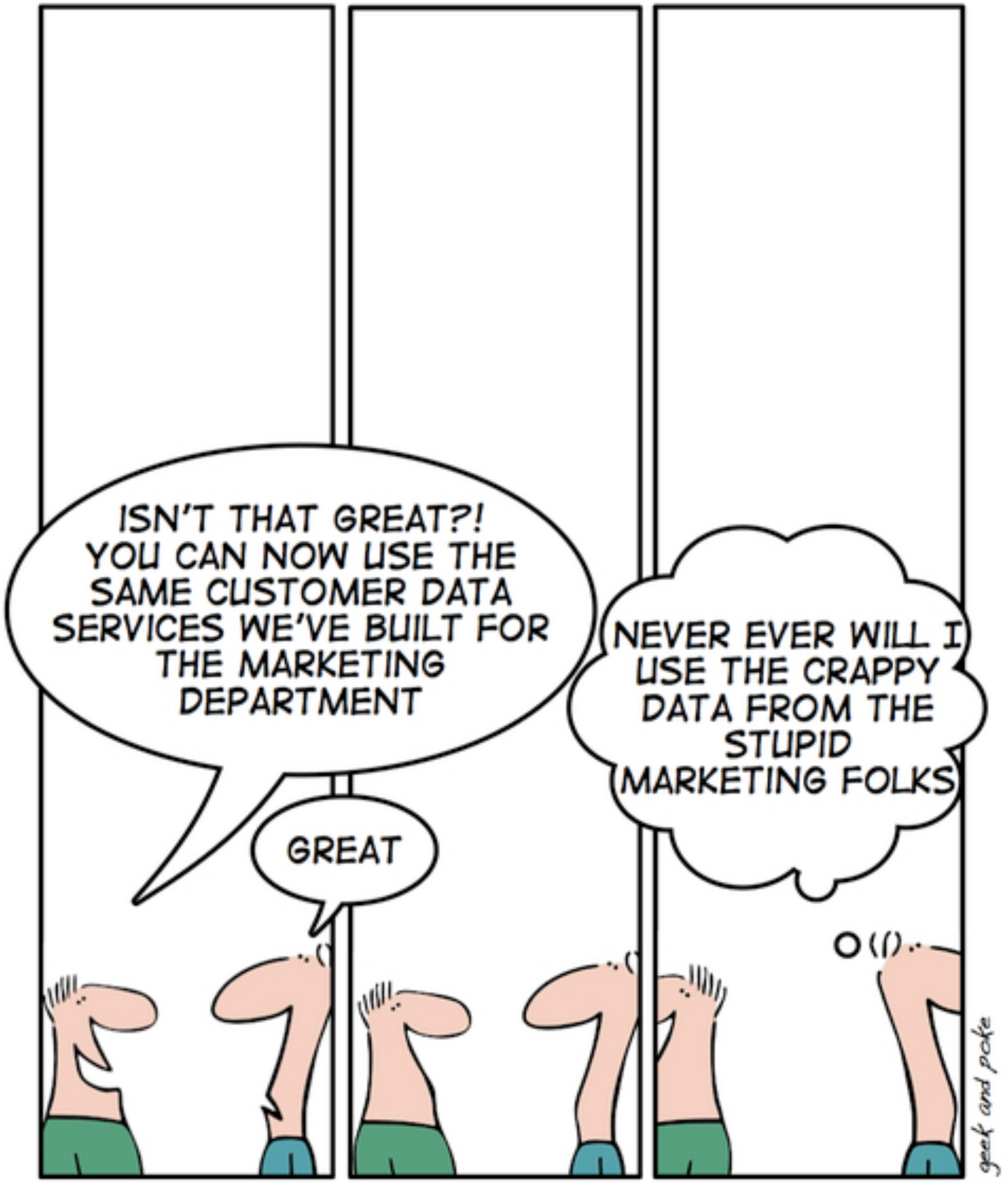
- BPMN, BPEL
- Executable Documentation?
- Visibility and Monitoring

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


ISN'T THAT GREAT?! YOU CAN NOW USE THE SAME CUSTOMER DATA SERVICES WE'VE BUILT FOR THE MARKETING DEPARTMENT
GREAT
NEVER EVER WILL I USE THE CRAPPY DATA FROM THE STUPID MARKETING FOLKS
geek and poke

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