Microservices and SOA

CON7364

Luis Weir
Oracle Ace Director
Principal, Capgemini

Robert Wunderlich
Sr. Principal Product Manager

September 22, 2016
Microservices and SOA
Oracle OpenWorld
San Francisco | September 18-22, 2016
Table of Contents

1. What is a Microservice
2. Microservices vs. SOA
3. Reference Architecture
4. Use Cases
Introduction
Resume

Luis Weir
Oracle Ace Director – Cloud Principal at Capgemini UK

I am an Oracle Ace Director, Cloud Principal and a Thought Leader specialised in Oracle Fusion Middleware & Oracle PaaS. With more than 15 years of experience implementing IT solutions across the globe, I have been exposed to a wide wide variety of business problems many of which I’ve helped solved by adopting SOA architectural styles such as traditional SOA, API management and now Microservices. My current focus is in assisting organisations define and implement solutions and strategies that can help them realise the benefits that such technologies have to offer.

I am very passionate about technology. I have be the lead authored of two books (Oracle SOA Governance 11g Implementation and Oracle API Management 12c Implementation), I am a regular blogger and speaker in major conferences and events. A well-known industry expert especially when it comes to Oracle middleware technologies I am also an OTN certified SOA black belt.

Latest Media:
- Systematic Approach for Migrating to Oracle Cloud SaaS (http://bit.ly/1Xr6acs)
- Oracle Magazine Jan/Feb 2016 (http://ora.cl/Vhh)
- API Management Implementation (http://ora.cl/Gcw)

2nd Place
1st OTN Cloud Hackathon
June, 2016

Cloud Contribution Award
SOA Community
March, 2016

Oracle API Management
12c Implementation
Learn how to successfully implement API management using Oracle's API Management Solution 12c.

Oracle SOA Governance
11g Implementation
Successfully implement SOA governance using Oracle SOA Governance 11g with the help of practical examples and case studies.

2nd Place 1st OTN Cloud Hackathon June, 2016 Cloud Contribution Award SOA Community March, 2016 Oracle API Management 12c Implementation Learn how to successfully implement API management using Oracle's API Management Solution 12c. Oracle SOA Governance 11g Implementation Successfully implement SOA governance using Oracle SOA Governance 11g with the help of practical examples and case studies.

2nd Place 1st OTN Cloud Hackathon June, 2016 Cloud Contribution Award SOA Community March, 2016 Oracle API Management 12c Implementation Learn how to successfully implement API management using Oracle's API Management Solution 12c. Oracle SOA Governance 11g Implementation Successfully implement SOA governance using Oracle SOA Governance 11g with the help of practical examples and case studies.
What is a Microservice
The three aspects of Microservices according to me!!
What is a Microservice?

“Loosely coupled service oriented architecture with bounded context”,
– Adrian Cockcroft, April 2015

Functional decomposition of systems into manageable and independently deployable components,
Microservice Architectures by Dr. Andreas Schroeder (http://bit.ly/1TOGZK8)

@kellabyte @mamund I used to call what we did "fine grain SOA". So microservices is SOA with emphasis on small ephemeral components
The **three** aspects of Microservices Architecture (according to me!)
Run on its own process

Deployed independently

Isolates faults

Scales independently

Is stateless

Owns its data

Microservice
Architectural

- Bounded context
- Single responsibility
- API gateways
- Choreographed
- Polyglot
- Smart endpoint and dump pipe
Organisational

Teams organized around business capabilities

Small teams

Products not projects

You build it you run it

Culture of automation

Decentralised governance

Microservice
Microservice vs. SOA
Key differences and communalities
“The value of the term microservices is that it allows to put a label on a useful subset of the SOA terminology”,
Martin Fowler (minute 14), GOTO conference, Berlin November 2014

- SOA: Typically adopted to deliver horizontal integrations
- Traditional SOA (i.e. AIA): Best for vertical integrations
- Microservices Architecture: Not for integration. Best for building modern systems

Wrong comparison… The difference lies in the realisation style to implement SOA

Inspiration from Martin Fowler’s Microservices presentation at GOTO conference, Berlin November 2014 (minute 14)
https://www.youtube.com/watch?v=wgdBVIX9faA
Microservices vs SOA – Technical Stack

### Legacy Monolith
- **Application Services** (ie. CICS)
- **Batch Services** (ie JCL, JES, 3rd p.)
- **Operating System** (ie. z/OS, z/VSE)
- **Hypervisor** (ie. PR/SM – Type 1)
- **Hardware Resources** (ie. DADB, IDMS, IEDN)
- **Storage** (ie. DB2, IMS/DB)
- **Mainframe** (ie IBM System Z, S/360)

### Modern Monolith
- **Services**
- **Application** (ie. ESB)
- **Application Server**
- **Bin/Libs (MREs, Interpreters, etc)**
- **Container Engine**
- **Guest OS (VMs)**
- **Hypervisor (type 1 or 2)**
- **Host OS**
- **Hardware Resources**
- **Any Hardware**
- **Relational Database**

### Microservice Architecture
- **Node**
- **Scala**
- **Java**
- **Ruby**
- **Jolie**
- **Cassandra**
- **Mongo**
- **Neo4j**
- **Oracle**
- **HBase**

### Comparison
- **Any Hardware**
- **Relational Database**
## Architectural

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Traditional SOA</th>
<th>MSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolith pattern (<a href="http://bit.ly/1Gjr2Y0">http://bit.ly/1Gjr2Y0</a>)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>API gateway pattern (<a href="http://bit.ly/1WTyNLJ">http://bit.ly/1WTyNLJ</a>)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Orchestration (<a href="http://bit.ly/1U0SWil">http://bit.ly/1U0SWil</a>)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Choreography (<a href="http://bit.ly/1ssALZQ">http://bit.ly/1ssALZQ</a>)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Decouple Contract (<a href="http://bit.ly/1O8mVpm">http://bit.ly/1O8mVpm</a>)</td>
<td>Yes</td>
<td>Could be….</td>
</tr>
<tr>
<td>Bounded Context (<a href="http://bit.ly/1o7AK8B">http://bit.ly/1o7AK8B</a>)</td>
<td>Some times</td>
<td>Yes</td>
</tr>
<tr>
<td>Ubiquitous Language (<a href="http://bit.ly/1c8nXQe">http://bit.ly/1c8nXQe</a>)</td>
<td>Some times</td>
<td>Yes</td>
</tr>
<tr>
<td>Bulkhead (<a href="http://bit.ly/1c8nXQe">http://bit.ly/1c8nXQe</a>)</td>
<td>Not really…</td>
<td>Yes</td>
</tr>
<tr>
<td>Client-side Service Discovery (<a href="http://bit.ly/1OunUyg">http://bit.ly/1OunUyg</a>)</td>
<td>Initially only (service registry)</td>
<td>Recommended</td>
</tr>
</tbody>
</table>
Choreography vs Orchestration – Which one is which?
Bounded Context

“Gather together those things that change for the same reason, and separate those things that change for different reasons” – The single responsibility principle by Robert C. Martin, November 2009, http://bit.ly/1VDgw79

“Domain driven design (DDD) divides up a large system into Bounded Contexts, each of which can have a unified model – essentially a way of structuring Multiple Canonical Models.”

Sales Context
- Opportunity
- Pipeline
- Territory
- Product
- Sales Person

Support Context
- Customer
- Ticket
- Defect
- Product Version

Use Bounded Context to Separate Concerns

How to eat the Elephant?

One piece at a time!

Understand the problem. Slice and dice your elephant by defining boundaries in the business capabilities. Modernise one piece at the time. Starting small.
Recommend

**Domain Driven Design & Microservices by Eric Evans**

- Goto Berlin, Nov 2014
- https://www.youtube.com/watch?v=yPvef9R3k-M

**Principles of Microservices By Sam Newman**

- Devoxx Belgium, Nov 2015
- https://www.youtube.com/watch?v=PFQnNFe27kU
Organisational

Traditional Operations Model

Development and support teams organized by technologies resulting in siloes (Conway’s law in action)

- UI Dev Team
- SOA Dev Team
- Database Dev Teams
- Project Teams
- Coms Gaps
- UI Support Team
- SOA Support Team
- DB Support Team

MSA Operations Model

Multi-disciplinary [small] teams organized by business capability resulting in modular systems

- Customer Microservice
- Orders Microservice
- Items Microservice
- Shipment Microservice
- DevOps / Continuous Delivery
Recommend

Modeling Microservices at Spotify with Petter Mahlen

At a microservices talk in March in Sweden, Petter Mahlen, Backend Infrastructure Engineer at Spotify, spoke to a packed house at Jfokus about microservices.

http://tinyurl.com/msasspotify
Reference Architecture
A reference architecture suitable for SOA 2.0: Microservices, API Management and Integration
SOA 2.0 Reference Architecture

SYSTEMS OF ENGAGEMENT

- Special Purpose APIs
- Presentation APIs
- Public [Consumer] APIs
- Partner [B2B] APIs

Single Purpose APIs

Business APIs

API Applications

- CX
- HCM
- ERP
- EPM
- Legacy, etc

SaaS

Message Pipe

Microservices

SYSTEMS OF RECORDS

SYSTEMS OF INNOVATION

SYSTEMS OF DIFFERENTIATION
### SOA 2.0 Capability Model

#### Single Purpose APIs

<table>
<thead>
<tr>
<th>Federated AuthN/AuthZ</th>
<th>API Key Validation</th>
<th>Call Aggregation</th>
<th>Tailored Contracts</th>
<th>Thread Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread Protection</td>
<td>Client Backend Logic Scripting</td>
<td>Platform/Backend APIs</td>
<td>Push Nots/Websockets</td>
<td>Polyglot Consumer SDKs</td>
</tr>
</tbody>
</table>

#### Business APIs

<table>
<thead>
<tr>
<th>AuthN/AuthZ/API Key Validation</th>
<th>Policy Enforcement</th>
<th>HTTP Routing</th>
<th>Calls Aggregation</th>
<th>Light Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light Scripting</td>
<td>In-memory Cache</td>
<td>Rate Limiting/Throttling</td>
<td>Streaming</td>
<td>REST/SOAP Conversions</td>
</tr>
</tbody>
</table>

#### API Applications

<table>
<thead>
<tr>
<th>System AuthN/AuthZ</th>
<th>Connectivity Adapters</th>
<th>Connection &amp; Session Management</th>
<th>Complex Orchestrations &amp; Logic</th>
<th>Protocol/Transport Conversions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Transformation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Microservices

<table>
<thead>
<tr>
<th>Polyglot Programming</th>
<th>Polyglot Persistency</th>
<th>Single Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choreography</td>
<td>Stack Independence</td>
<td>Auto Scaling</td>
</tr>
</tbody>
</table>

#### Message Pipe

<table>
<thead>
<tr>
<th>Message routing</th>
<th>Light transformation</th>
<th>Reliable Messaging</th>
<th>Push Listener</th>
<th>Queuing/De-queueing</th>
</tr>
</thead>
</table>
SOA 2.0 In the Oracle Cloud PaaS

Single Purpose APIs
- Mobile Cloud
- API Platform Cloud
- Java Cloud
- App Container Cloud

Business APIs
- API Platform Cloud

API Applications
- SOA Cloud
- Java Cloud

Microservices
- App Container & Container Cloud
- DB and NoSQL Clouds

Message Pipe
- Integration Cloud
- Messaging Cloud
Use Case
Modern application based on Microservice Architecture with integration to on-premises legacy systems for synchronization
Conceptual Solution Architecture

Customer Relations Domain
- Contact
- Customer

Order & Fulfillment Domain
- Order
- Product
- Shipment

ERP Domain
(P2C, R2C, HR, GL, Billing, etc.)

API Gateway(s)

Mobile Apps

Web Apps Applications

Message Pipe
- Bounded Context
- Sync Communication
- Async Communication
- Choreography
- Managed API
- Microservice
- Monolith Service
1) User access URL and renders page
2) User performs action in client side (i.e., Update personal details) which triggers an API [PUT] request
3) A customer business API resource is invoked i.e., [PUT]/customers/{person id}. The person update details are passed in the HTTP body as a JSON. API key and user token are also passed
4) Request is validated (key, user token), policies applied and if successful request PUT request is routed to the relevant customer microservice endpoint
5, 6, 7) The microservice (implemented in Node.JS) executes the business logic which results in updating the customer personal details JSON object in the NoSQL database and also triggering an update event by calling the messaging cloud API. A HTTP 200 response is send back if all goes OK
8, 9) A HTTP 200 response is send back with a small JSON object in the body with an acknowledgment (i.e. { status: "no errors" })
10, 11, 12) Once ICS detects a new message in the topic, it dequees the message, transforms it and via the connectivity agent, calls the relevant PLSQL API to update the customer record
Conclusion

Why adopt Microservices?
Conclusion - Why adopt Microservices Architecture (I)

**Modularity**

*Eat the elephant one piece at the time.* Phased implementation approach. Starting small. Small teams owning full lifecycle of their piece (a business capability) [23]

**Segmented complexity**

Separate a big problem into smaller problems handled by small teams ensures mental models are retained avoiding a “legacy in the making” [5]

**Ease of deployment/speed**

Moving away from entire system deployment (i.e. “one line change to a million-line-long monolithic”). Deploy services independently and fast (i.e. with containers) and introduce automation (continuous delivery) [23]
Conclusion - Why adopt Microservices Architecture (II)

**Scalability & resilience**

Scale independently and possibly on-demand. Bulkheads to isolate problems and avoid whole system failures (avoiding the cascade effect), Then purposely test resilience [23]

**Breaking organizational silos**

Organize small teams based on business capabilities in order to avoid organizational silos being reflected in the way systems are built (Conway’s law [32])

**Enabling cloud transition**

Building container-based modular applications whilst adhering to basic principles (like 12 factor [30], Lehman’s law [31], and the reactive manifesto [33]) cloud adoption is a real option
About Capgemini and Sogeti

With more than 180,000 people in over 40 countries, Capgemini is a global leader in consulting, technology and outsourcing services. The Group reported 2015 global revenues of EUR 11.9 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at www.capgemini.com.

Sogeti is a leading provider of technology and software testing, specializing in Application, Infrastructure and Engineering Services. Sogeti offers cutting-edge solutions around Testing, Business Intelligence & Analytics, Mobile, Cloud and Cyber Security. Sogeti brings together more than 23,000 professionals in 15 countries and has a strong local presence in over 100 locations in Europe, USA and India. Sogeti is a wholly-owned subsidiary of Cap Gemini S.A., listed on the Paris Stock Exchange.

www.capgemini.com
www.sogeti.com

The information contained in this presentation is proprietary. Copyright © 2016 Capgemini and Sogeti. All rights reserved. Rightshore® is a trademark belonging to Capgemini.
Integrated Cloud
Applications & Platform Services