

Contract Testing

Toni Masotti
masotti@bikeleasing.de
Bikeleasing-Service

November 25, 2023

Outline

- 1 Problem statement
 - Company growth
 - Test Typology
 - How to test microservices?
- 2 Testing complex distributed architectures
 - Contract Testing
- 3 Demo Time
- 4 Worth thinking about



Let's start

It all started with Bob...



...and it continues with Bob

It all started with Bob...



Bob

They do work on
my local machine
though 🙄

BOB ! the changes you
made do not work on the
live system 😡



Angry chef

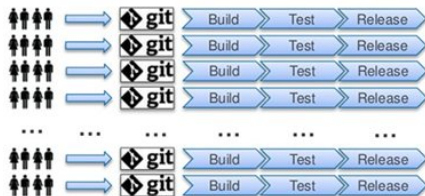
Bob has learnt the lesson...





Extreme examples...

- 50 Million Deployments a Year
- Software enhancements delivered every second



Gigantic Web of Micro-services at Amazon



Everybody loves microservices

- Big promise: Independent deployability

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture
- Fast onboarding

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture
- Fast onboarding
- Cost-efficiency on the long run

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture
- Fast onboarding
- Cost-efficiency on the long run
- Cloud native

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture
- Fast onboarding
- Cost-efficiency on the long run
- Cloud native
- Tech-agnostic approach

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture
- Fast onboarding
- Cost-efficiency on the long run
- Cloud native
- Tech-agnostic approach
- Scalability out of the box

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture
- Fast onboarding
- Cost-efficiency on the long run
- Cloud native
- Tech-agnostic approach
- Scalability out of the box
- Improved Security (no SPOF, no SPOC)

Everybody loves microservices

- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture
- Fast onboarding
- Cost-efficiency on the long run
- Cloud native
- Tech-agnostic approach
- Scalability out of the box
- Improved Security (no SPOF, no SPOC)
- Optimized Time to Market

Everybody loves microservices

“ Chief among the benefits of service-enabling an enterprise's application landscape are **increased organizational agility** and **reduced overall cost of implementing change**.

A SOA increases organizational agility by placing high-value business functions in **discrete, reusable services**, and then connecting and orchestrating these services to satisfy core business processes.

It reduces the cost of change by **reducing the dependencies** between services, allowing them to be **rapidly recomposed and tuned** in response to change or unplanned events.

”



Consumer-Driven Contracts: A Service Evolution Pattern

This article discusses some of the challenges in evolving a community of service providers and consumers. It describes some of the coupling issues that arise when service providers change parts of their contract, particularly document schemas, and identifies two well-understood strategies - adding schema extension points and performing "just enough" validation of received messages - for mitigating such issues. Both strategies help protect consumers from changes to a provider contract, but neither of them gives the provider any insight into the ways it is being used and the obligations it must maintain as it evolves. Drawing on the assertion-based language of one of these mitigation strategies - the "just enough" validation strategy - the article then describes the "Consumer-Driven Contract" pattern, which imbues providers with insight into their consumer obligations, and focuses service evolution around the delivery of the key business functionality demanded by consumers.

12 June 2006



Ian Robinson

Ian Robinson is a Principal Consultant with Thoughtworks. He specializes in helping clients create sustainable service-oriented development capabilities that align business and IT from inception through to operation. He has written guidance for Microsoft on implementing service-oriented systems with Microsoft technologies, and has published articles on business-oriented development methodologies and distributed systems design - most recently in *The Thoughtworks Anthology*. He is currently co-authoring a book on Web-friendly enterprise software.

CONTENTS

- Evolving a Service: An Example
- Interlude: Burdened With Services
- Schema Versioning
- Extension Points
- Breaking Changes
- Schematron
- Consumer-Driven Contracts
- Provider Contracts
- Consumer Contracts
- Consumer-Driven Characteristics
- Summary of Contract Characteristics
- Implementation
- Benefits
- Liabilities

APPLICATION INTEGRATION

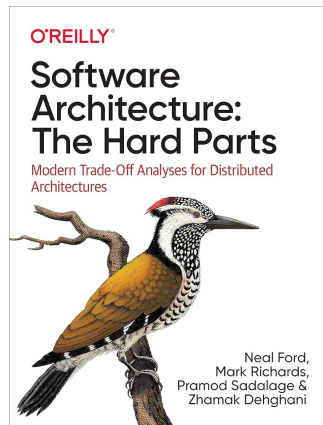
WEB SERVICES

Is always a tradeoff

“ *There is only one hard rule in software development: **Everything is a tradeoff.***

– Neal Ford

”



1 Problem statement

- Company growth
- Test Typology
- How to test microservices?

2 Testing complex distributed architectures

- Contract Testing

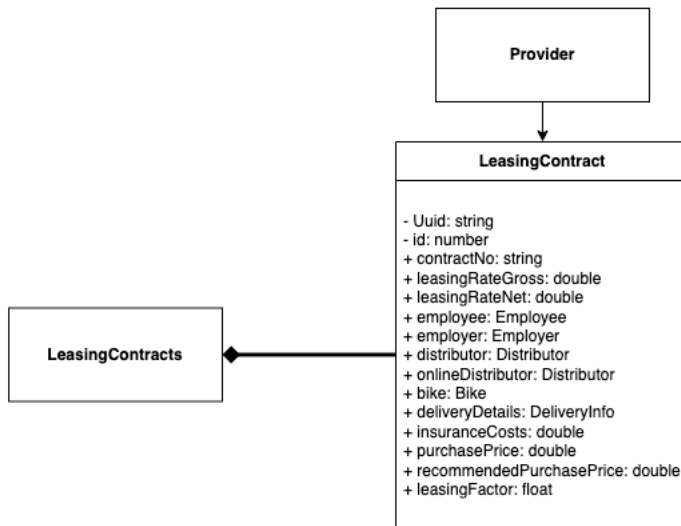
3 Demo Time

4 Worth thinking about

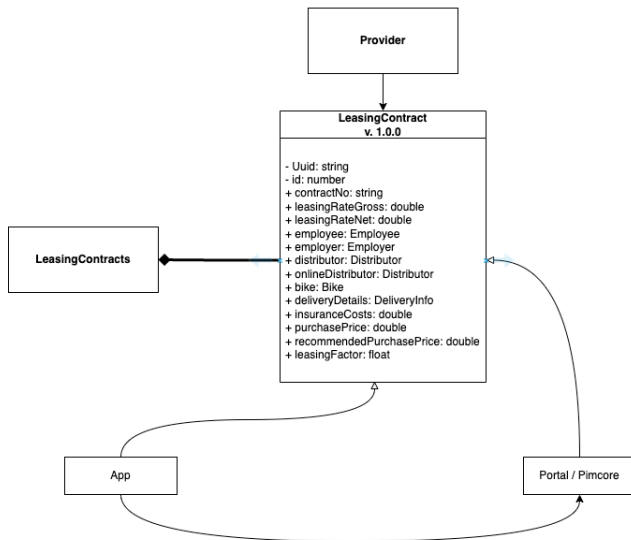
Bob's company is being very successfull...



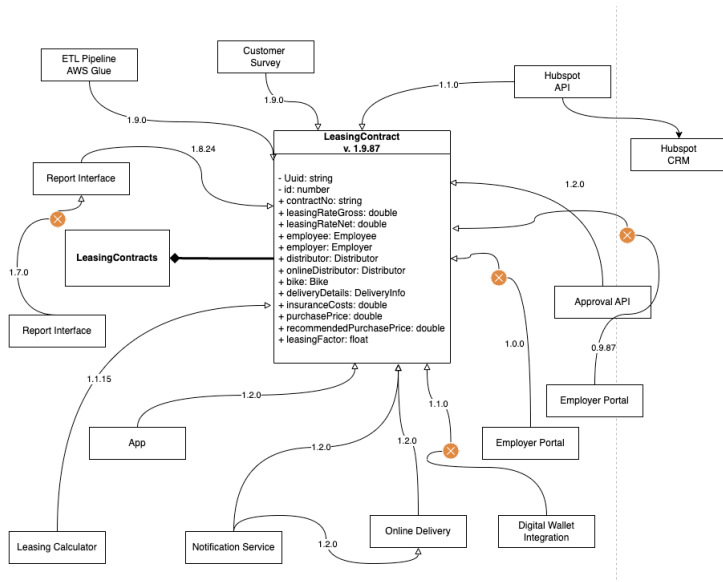
A company growth... so its architecture



A company growth... so its architecture



A company growth... so its architecture



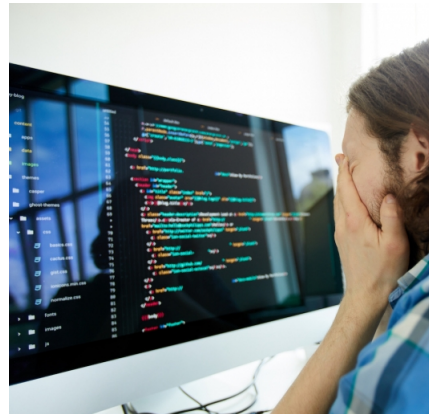
Bob is happy...



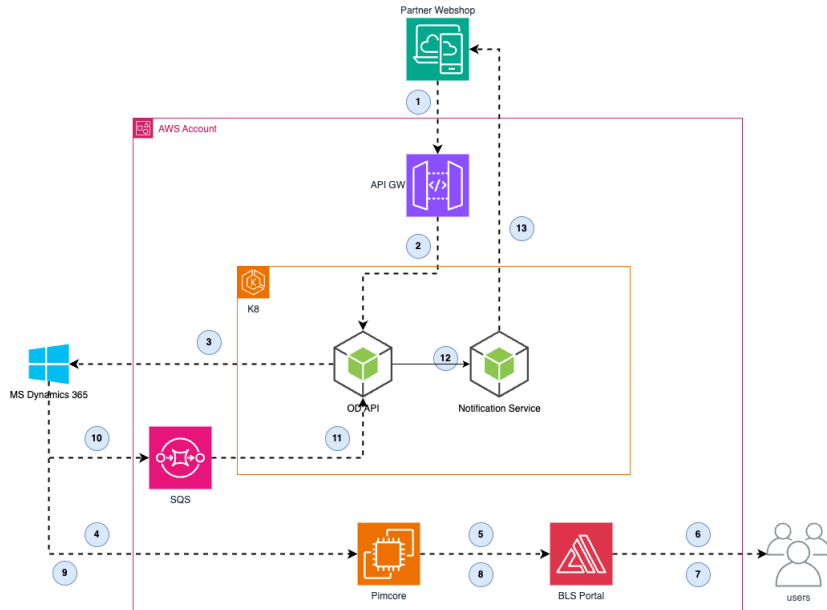
The Problem...

“ *Every time we touch one service, all the others break :(* ”

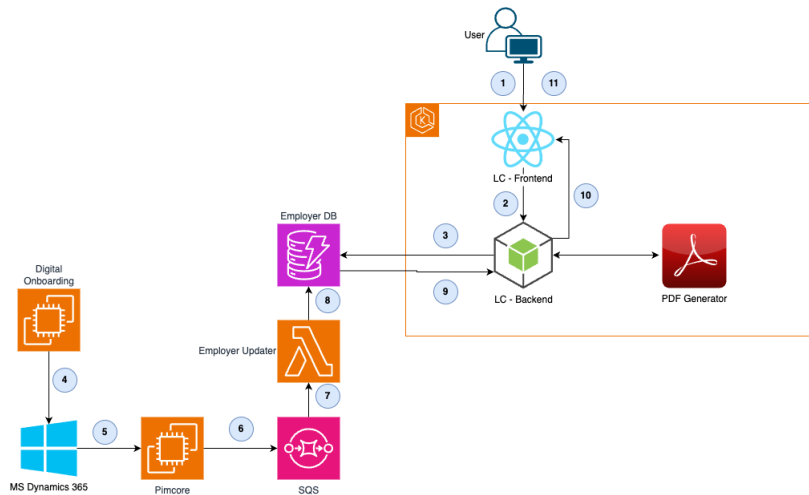
- Dependency hell
- Devs don't have confidence in deploying changes
 - Consequences of changes are hard to track
 - Distributed architectures are hard to test



Architecture example: Online Delivery



Architecture example: Leasing Calculator



Updating a microservice: the sad Atlassian case

```
{  
  "users": "Mike",  
  "address": "Something"  
}
```

```
1 {  
2   "user": "Mike",  
3   "address": "Something"  
4 }  
5
```



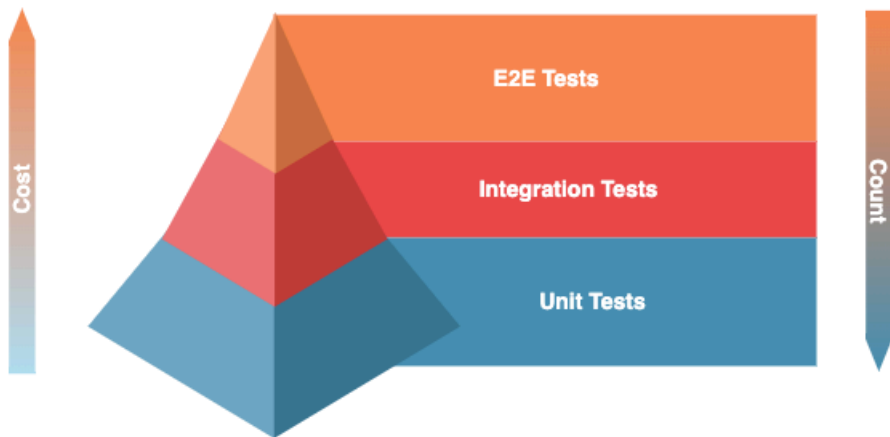
Something's gone wrong



Our team has been notified. If the problem persists,
please contact Atlassian Support.

[Reload page](#)

Test Typology



The problem with Unit Tests alone

- Local changes, even if tested, do not assure that the system as whole is working
- Mocks are not guaranteed to really represent the other part of the system

Problem with E2E Tests

- Hard or impossible to test some scenarios (system not available)
- Expensive (timely and money)

Distributed \neq Decoupled



Is this one or three pipelines?

Mocks

- Mocking

Problem Statement

When organizations grow and so do the number of services

- How can we know, that the **whole** system works, when I publish changes?
- How can I avoid dependency on E2E tests for deployment and still be confident?
 - Testing microservices in isolation is not enough

Many hops...

- Show here what it would mean for us to have pure e2e tests

Problems

- Local changes, even if tested, do not assure that the system as whole is working

Problems

- Local changes, even if tested, do not assure that the system as whole is working
- Loosely coupled architecture but tightly coupled teams. Such systems are - no matter the architecture - still monolithic from a CD perspective

Embedded Animation

- 1 Problem statement
 - Company growth
 - Test Typology
 - How to test microservices?
- 2 Testing complex distributed architectures
 - Contract Testing
- 3 Demo Time
- 4 Worth thinking about

Hintergrund

- ja, something

Pacts

Pact

A formal agreement between parties or individual. Synonyms: agreement, protocol, deal

— Oxford English Dictionary

- 1 Problem statement
 - Company growth
 - Test Typology
 - How to test microservices?
- 2 Testing complex distributed architectures
 - Contract Testing
- 3 Demo Time
- 4 Worth thinking about

Hintergrund

- ja, something

- 1 Problem statement
 - Company growth
 - Test Typology
 - How to test microservices?
- 2 Testing complex distributed architectures
 - Contract Testing
- 3 Demo Time
- 4 Worth thinking about

Hintergrund

- ja, something