Contract Testing

Toni Masotti
masotti@bikeleasing.de
Bikeleasing-Service

December 1, 2023

Outline

1. Introduction.. get the ball rolling
2. The problem(s) we're trying to solve
3. Option C : Contract Testing
4. Demo Time - Hands on lab
5. Concluding thoughts

1 Introduction.. get the ball rolling

2 The problem(s) we're trying to solve

3 Option C : Contract Testing

4 Demo Time - Hands on lab

5 Concluding thoughts

It all started with Bob...

...and it continues with Bob

It all started with Bob…

They do work on my local machine though 😢

Bob

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

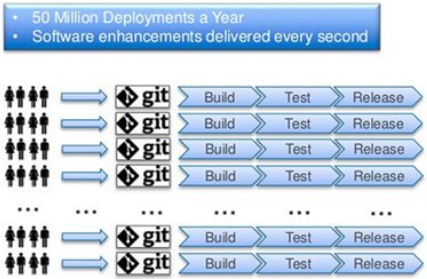
Angry chef

Bob has learnt the lesson...


MICROSERVICES
MICROSERVICES EVERYWHERE
imgflip.com/memegenerator.net


Financial
Healthcare
Governments
Transport
Education
Telecom
Retail
Technology
Children's Hospital Boston
HM Revenue & Customs
CitySprint
ACU
etisalat
ebay
cisco
BDO
HCA
Ministry of Business, Innovation & Employment
SCHOLASTIC
TELECOM
StubHub
CONCUR.
ING
KAISER PERMANENTE
Transport for London
Mercedes-Benz
WAGENINGEN
Telefónica O2
TicketNetwork
NUTANIX
Fidelity
McKESSON
Homeland Security
UBER
Ucla
verizon
Travis Perkins
Trimble
SPECTRUM HEALTH
UNITED
UNIVERSITY OF MICHIGAN
west
Wickes
T-Systems

Extreme examples...



Benefits of microservices

- Big promise: Independent deployability

Benefits of microservices

- Big promise: Independent deployability
- Fast feedback loops

Benefits of microservices

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

Benefits of microservices

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

Benefits of microservices

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

Benefits of microservices

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

Benefits of 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

Benefits of 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

Benefits of 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

Benefits of microservices

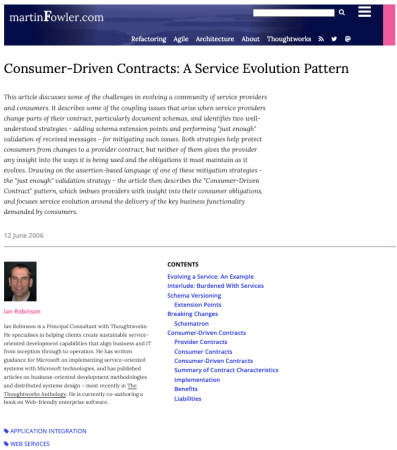
- Big promise: Independent deployability
- Fast feedback loops
- Strong team focus, ideally crossfunctional
- Loosely coupled architecture
- Fast onboarding
- Cost-efficiency on the long run
- Cloud native
- Tech-agnostic approach
- Scalability out of the box
- Improved Security (no SPOF, no SPOC)

Benefits of 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

Two key factors for success

> *"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."*



martinFowler.com

Refactoring Agile Architecture About Thoughtworks

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 specialises 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.

APPLICATION INTEGRATION

WEB SERVICES

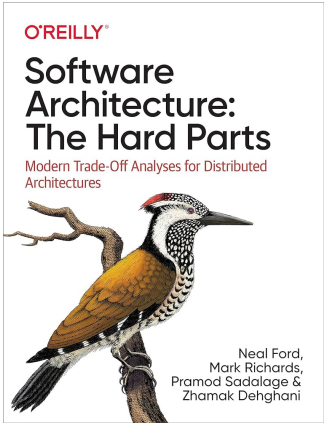
CONTENTS

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 Contracts
Summary of Contract Characteristics
Implementation
Benefits
Liabilities

But...

It's always a tradeoff

> *The first (and probably only) law of software architecture states* ***"Everything in software architecture is a tradeoff"***
>
> *– Neal Ford*



1 Introduction.. get the ball rolling

2 The problem(s) we're trying to solve

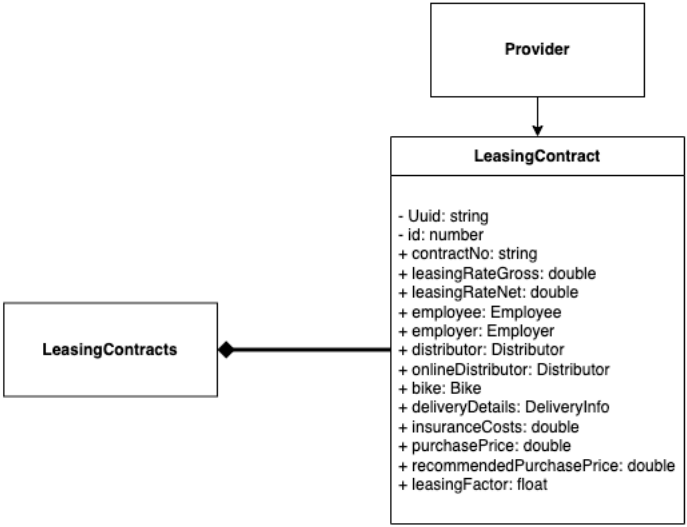
3 Option C : Contract Testing

4 Demo Time - Hands on lab

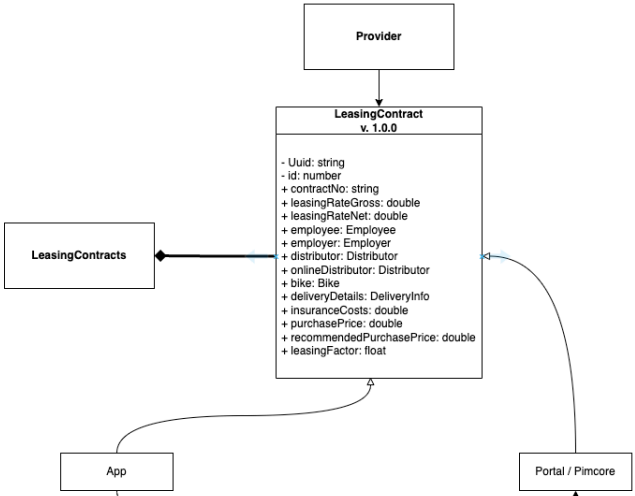
5 Concluding thoughts

Bob's company is being very successfull...

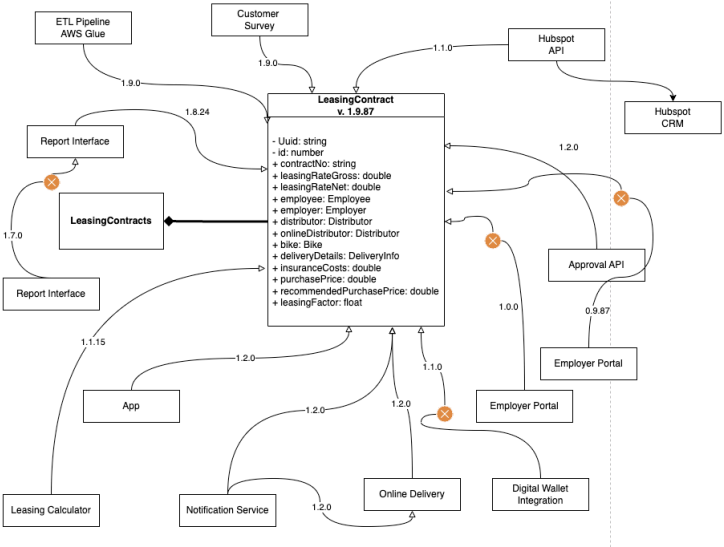
The shy beginning...



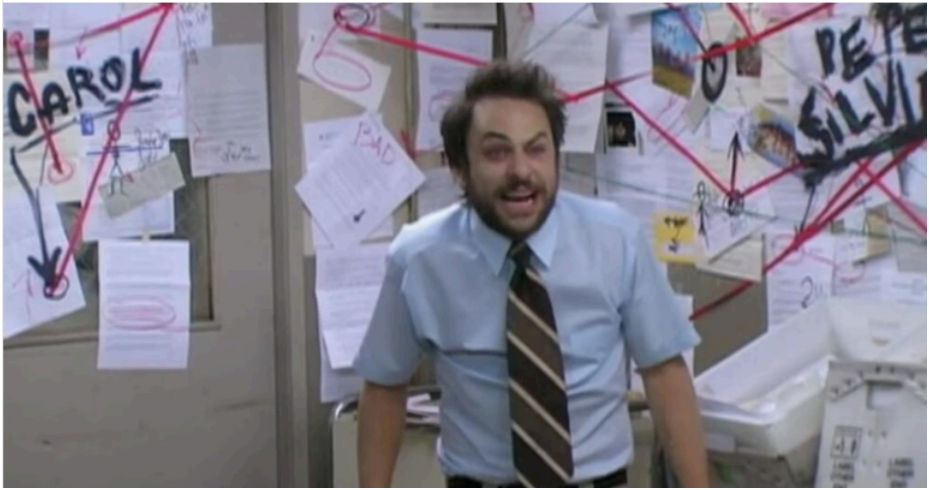
Baby's first steps...



Bigger steps...



...towards madness

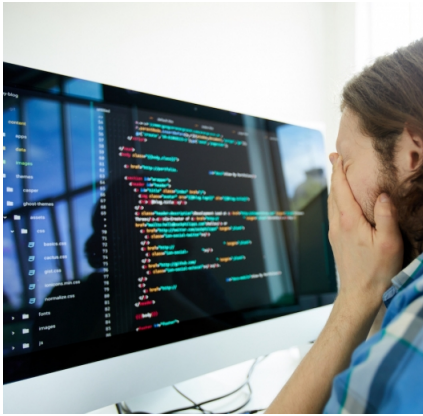


Toni on a good day

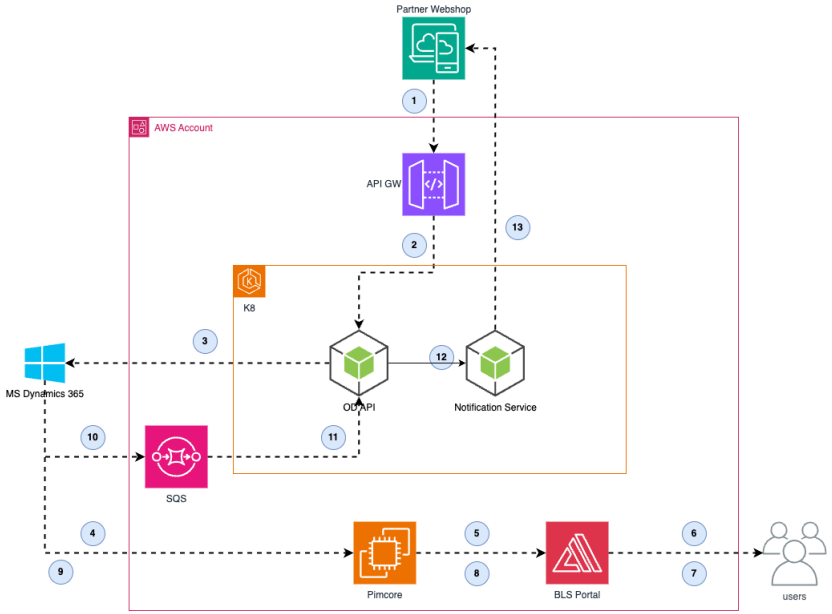
The Problem(s)...

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

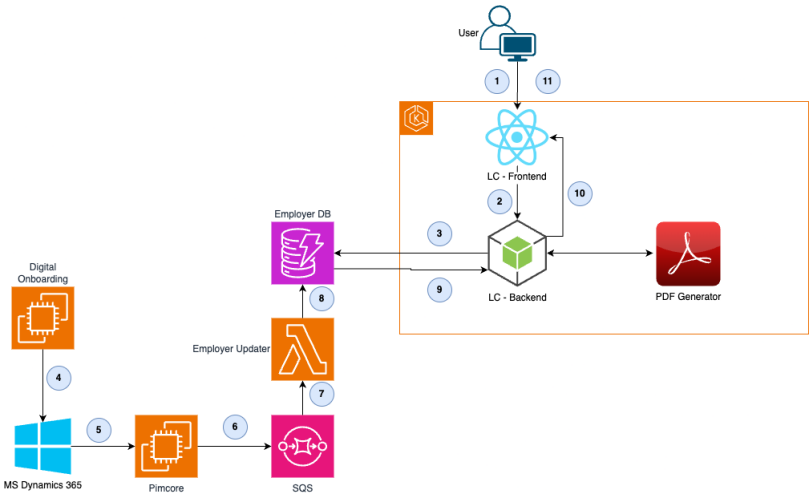
- Risk of dependency hell
- Devs loose confidence in deploying changes
  - Consequences of changes are hard to track
  - Distributed architectures are *hard* to test



Architecture example: Online Delivery



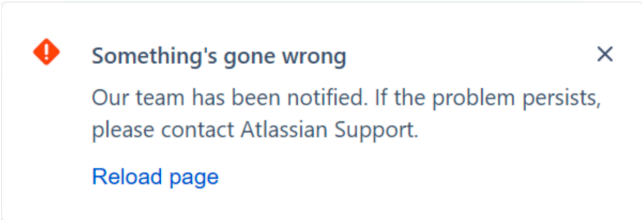
Architecture example: Leasing Calculator



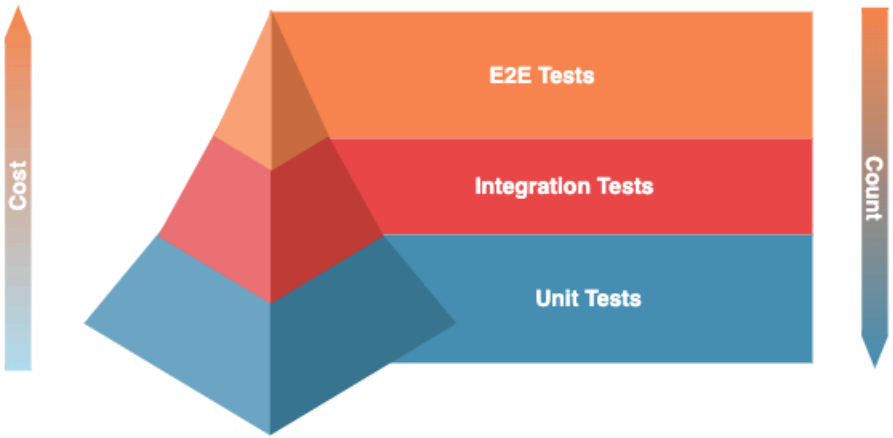
Updating a microservice: the sad Atlassian case

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

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

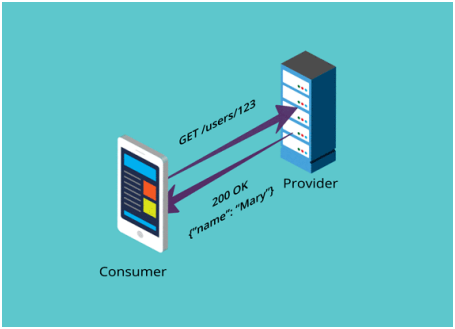


Test Pyramid

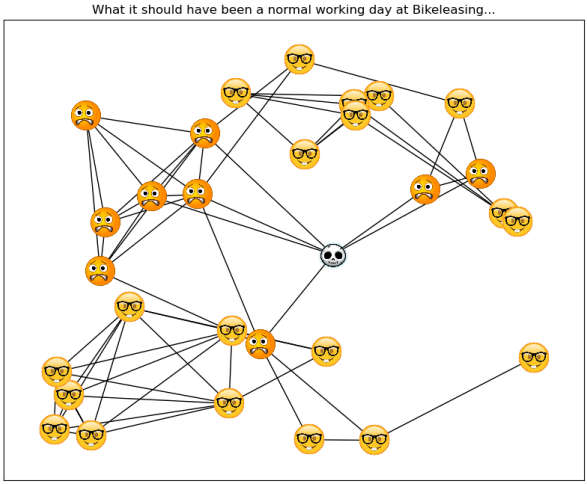


The problem with Unit Tests alone

- Local tests do not assure that the system as whole is still working
- Mocks are not guaranteed to really represent the other part of the system



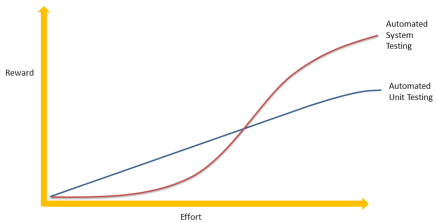
Error Propagation



Problem with E2E Tests

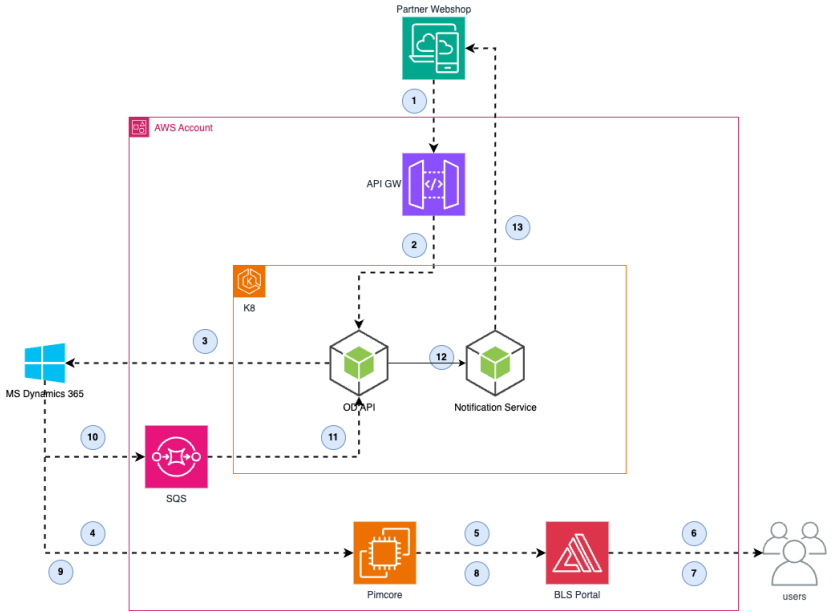
They're very important (*qua* realistic), **but**...

- they're hard to run locally (especially in a distributed system)
- it's hard or impossible to test some scenarios (e.g. system not available)
- they're timely and financially expensive
  - The longer we need to find out what's wrong, the more expensive it is
- not few companies fall in the trap of *excessive test setup*
- they're prone to flakiness



Link: Trinitatum Blog (2015-03-02)

Architecture example



Distributed $\neq$ Decoupled

Is this one or three pipelines?

What do we need?

We need a way to **test and ensure correct data exchange** between services, with the following characteristics:

- **Fast feedback** on changes
- **Confidence** in our changes
- **Reproducibility** of tests
- **Realistic** tests
- **Cheap** tests
- **Easy** to write and maintain

What do we need?

Seen so far:

- **Option A**: Mocks
- **Option B**: E2E / System Tests with real infrastructure

Is there an option C?

1 Introduction.. get the ball rolling

2 The problem(s) we're trying to solve

3 Option C : Contract Testing

4 Demo Time - Hands on lab

5 Concluding thoughts

Contracts

Contracts

A formal agreement between parties or individual. Synonyms: pact, agreement, protocol, deal
— Oxford English Dictionary

Contract Testing

A contract test is a test at the boundary of an external service, verifying that it meets the contract expected by a consuming service.
— Martin Fowler

A pact between who?

- **Consumer**: the service that consumes the information (typically closer to the user)
- **Provider**: the service that provides the information (typically closer to the data)

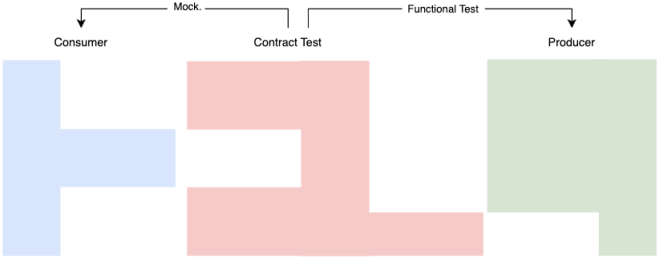
A pact between who?

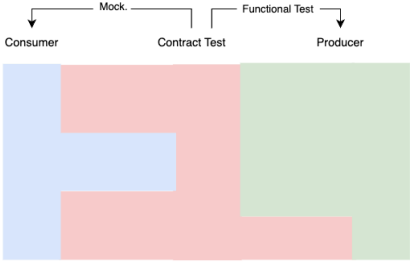
- **Consumer**: the service that consumes the information (typically closer to the user)
- **Provider**: the service that provides the information (typically closer to the data)

Where do we find this architecture?

- REST, GraphQL, gRPC, SOAP ...APIs
- Microservices communication (over TCP, HTTP, Quic, AMQP, ...)
- SOA
- Other kinds of distributed systems (IoT, ...)
- Client-Server
- Messaging systems (Kafka, RabbitMQ, SQS ...)
- Notifying systems (Webhooks, SNS, ...)

The role of contracts in testing

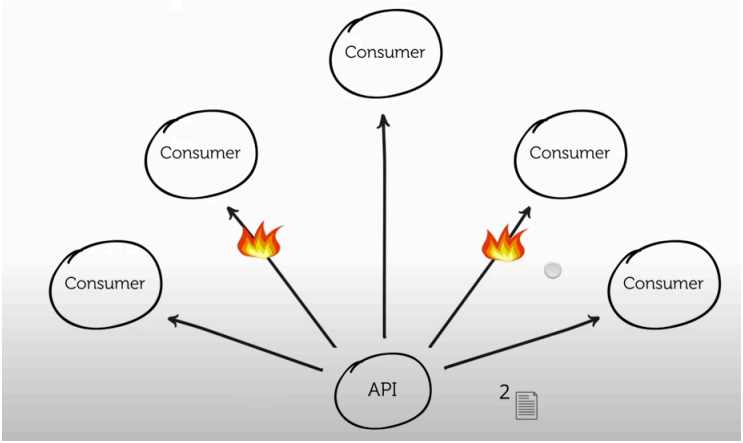




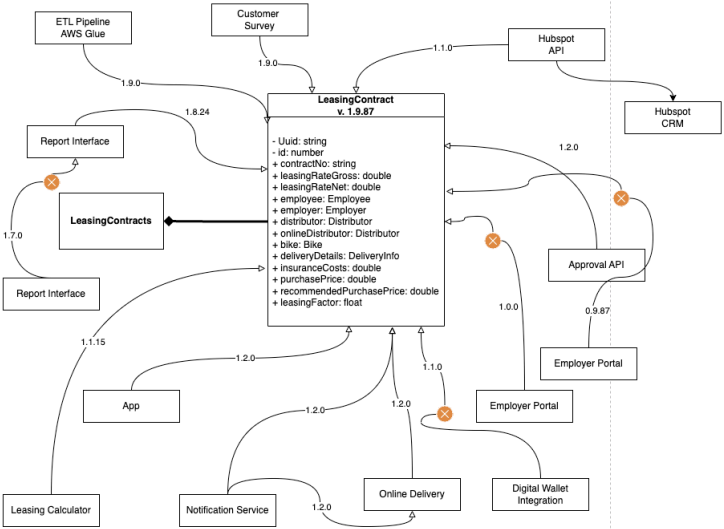
Who does initiate the contract?

Both parties can initiate the contract, **but**...
Here I will present you the **Consumer Driven Contracts** approach.

Provider driven contracts

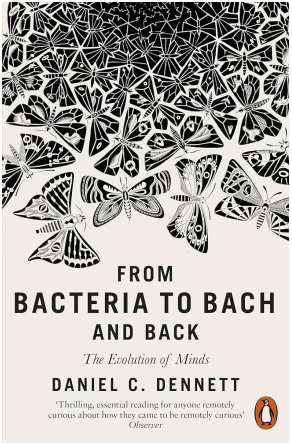


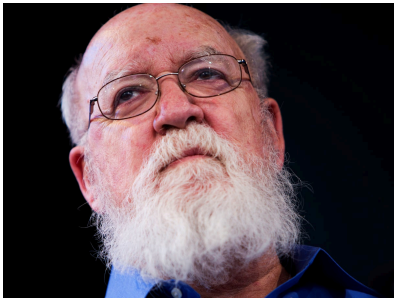
Provider driven contracts



Consumer Driven Contracts

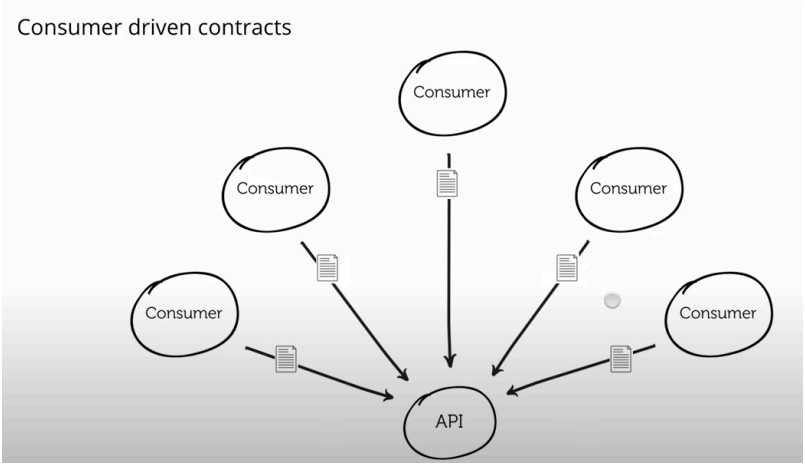
> *A strange inversion of reasoning...– D. Dennet*



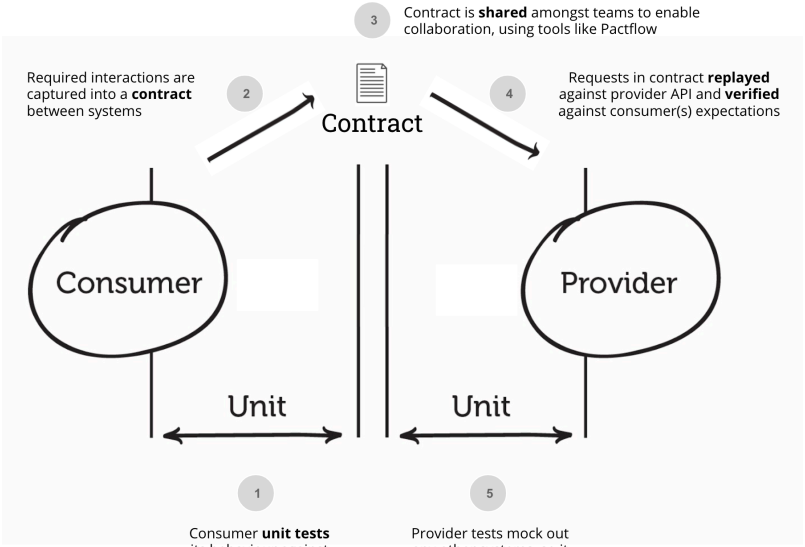


Consumer Driven Contracts

> *A strange inversion of reasoning...– D. Dennet*



Pact - Overview



1 Introduction.. get the ball rolling

2 The problem(s) we're trying to solve

3 Option C : Contract Testing

4 Demo Time - Hands on lab

5 Concluding thoughts

Hintergrund

1 Introduction.. get the ball rolling

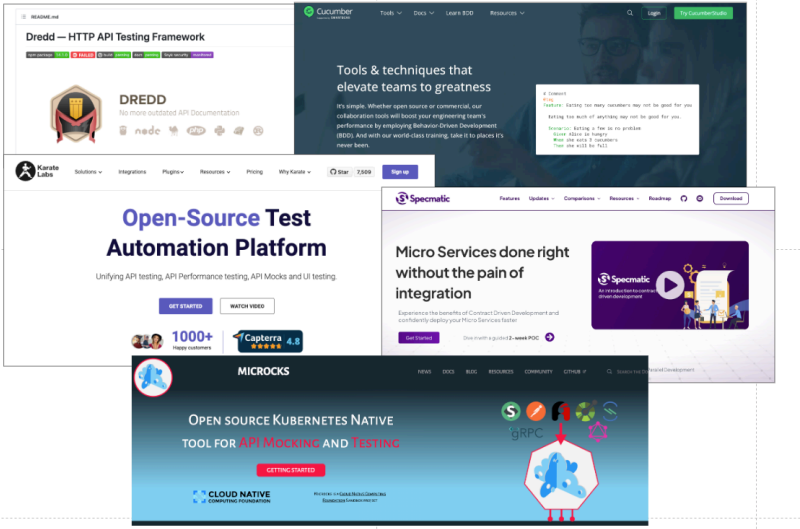
2 The problem(s) we're trying to solve

3 Option C : Contract Testing

4 Demo Time - Hands on lab

5 Concluding thoughts

Other tools


Dredd — HTTP API Testing Framework
DREDD
No more outdated API Documentation
Cucumber
Tools
Docs
Learn BDD
Resources
Login
Try CucumberStudio
Tools & techniques that elevate teams to greatness
It's simple. Whether open source or commercial, our collaboration tools will boost your engineering team's performance by employing Behavior-Driven Development (BDD). And with our world-class training, take it to places it's never been.
Karate Labs
Solutions
Integrations
Plugins
Resources
Pricing
Why Karate
Star 7,509
Sign up
Open-Source Test Automation Platform
Unifying API testing, API Performance testing, API Mocks and UI testing.
GET STARTED
WATCH VIDEO
1000+
Happy customers
Capterra
4.8
Specmatic
Features
Updates
Comparisons
Resources
Roadmap
Download
Micro Services done right without the pain of integration
Experience the benefits of Contract Driven Development and confidently deploy your Micro Services faster
Get Started
MICROCKS
NEWS
DOCS
BLOG
RESOURCES
COMMUNITY
GITHUB
OPEN SOURCE KUBERNETES NATIVE TOOL FOR API MOCKING AND TESTING
GETTING STARTED
gRPC
CLOUD NATIVE COMPUTING FOUNDATION

CDC vs. PDC

- **Consumer Driven Contract**
  - Open and incomplete
  - Multiple and collaborative
  - Bounded stability and immutability
- **Provider Driven Contract**
  - Closed and complete
  - Singular and authoritative
  - Bounded stability and immutability

Benefits of CDC

- Customer centric
  - Focus on what to deliver to customers
  - Fine grained insights about dynamics of the system
- Contract change as proposal
- Fast feedback loop, with realistic scenarios, while keeping tests isolated
- Improved communication between teams

Consequences for the development process

- Design by contract (SOLID, especially the **I**)
- Need for separated API Layers
- More focus on the client-facing parts of the system


THANK
YOU