andrena OBJECTS

Experts in agile software engineering

Mit Pact REST-Schnittstellen innerhalb und außerhalb des Teams definieren und stabilisieren

Entwicklertag Karlsruhe 2019

03.06.19
Karlsruhe

Eva Ziebarth

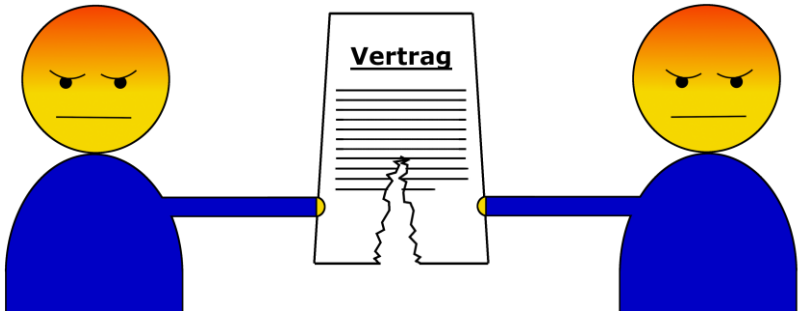
eva.ziebarth@andrena.de

Marvin Kranz

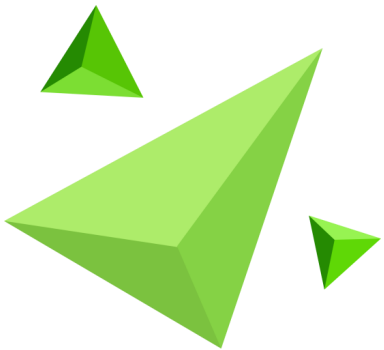
marvin.kranz@andrena.de

Agenda

1. Motivation
2. Consumer Driven Contract Testing – Was ist das?
3. Einführung im Projekt
4. Fazit

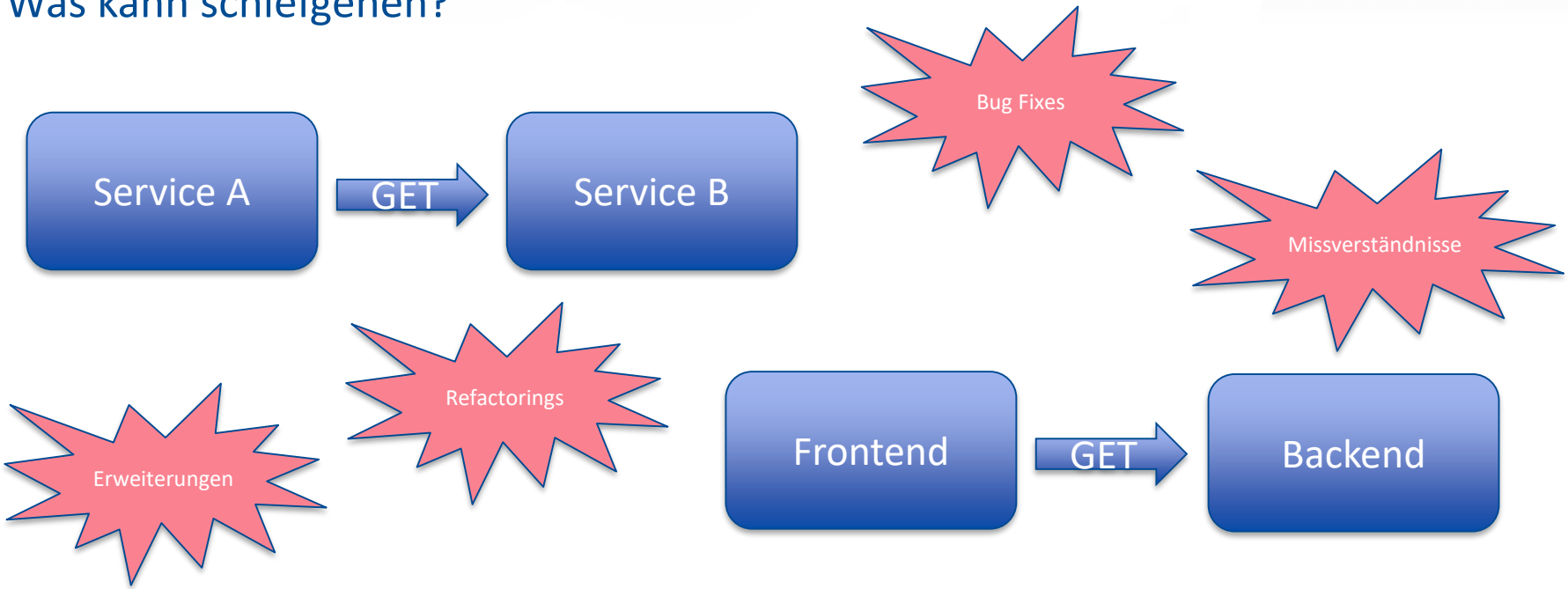


Motivation



Motivation - Schnittstellen innerhalb eines Teams

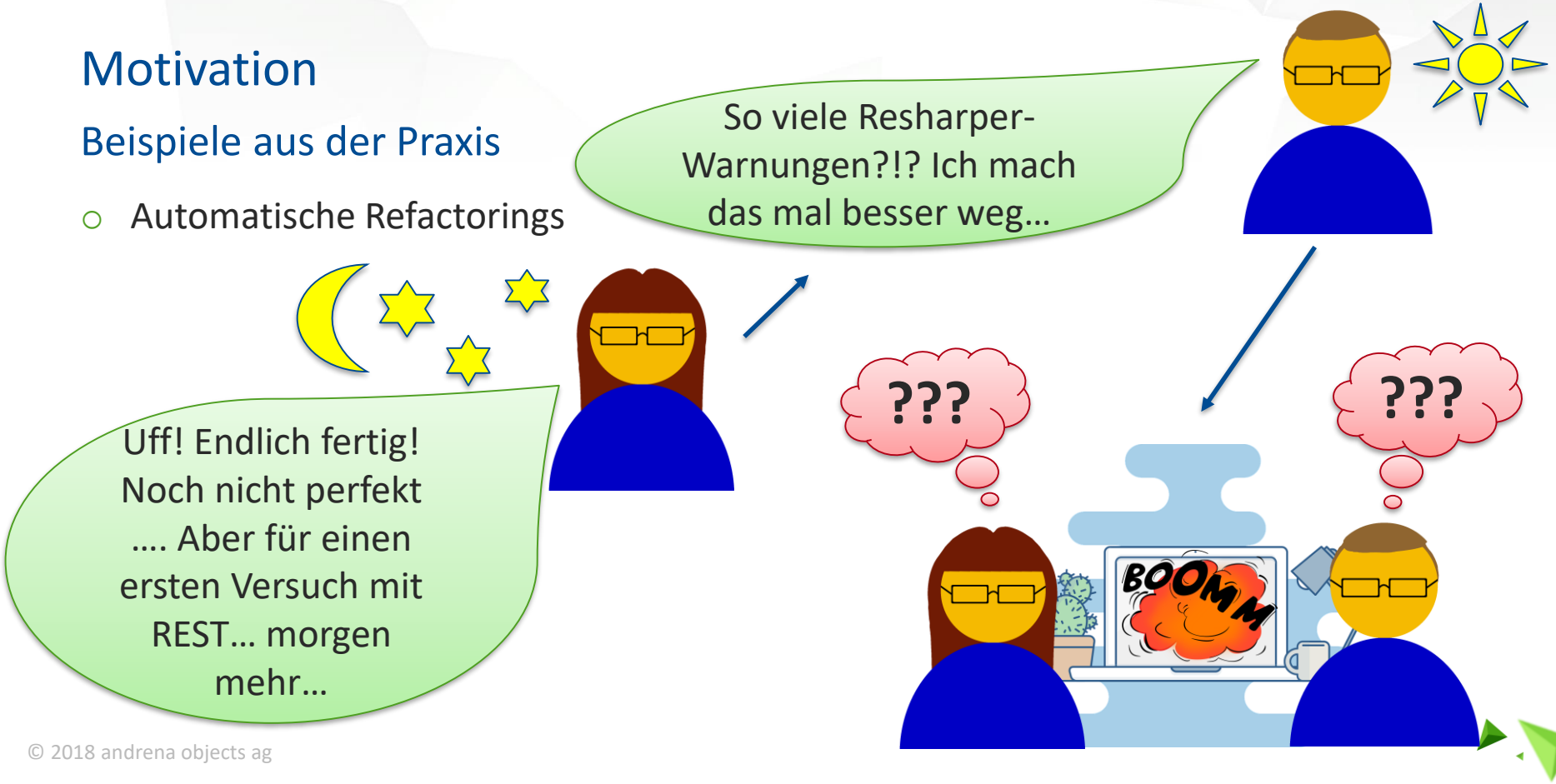
Was kann schiefgehen?



Motivation

Beispiele aus der Praxis

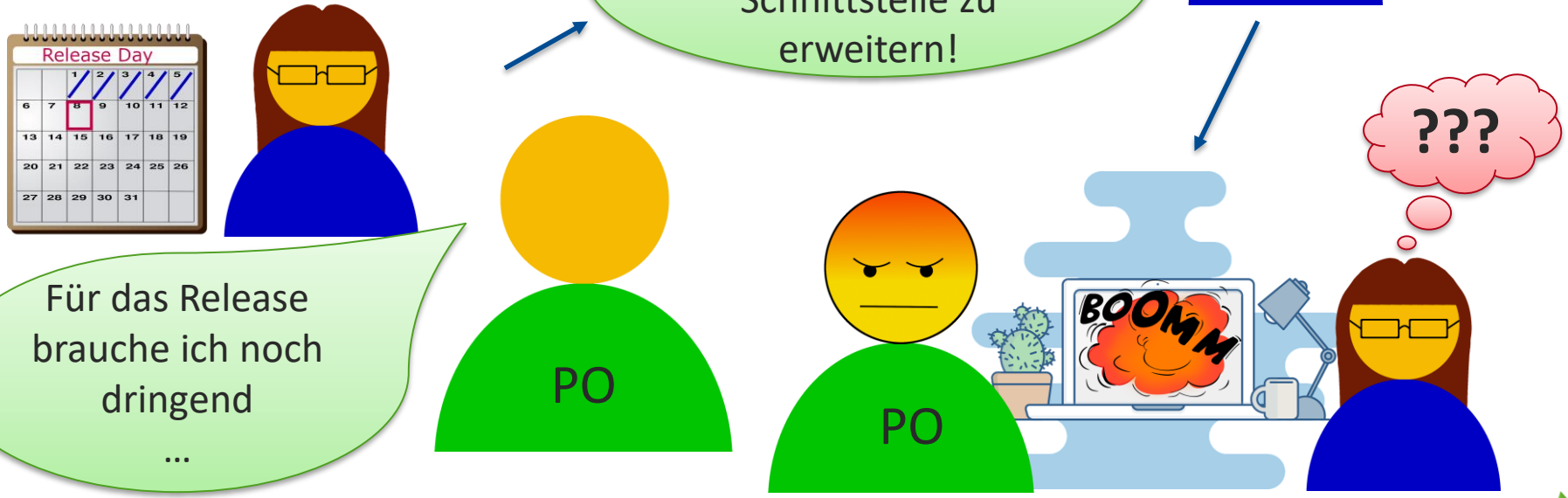
- Automatische Refactorings



Motivation

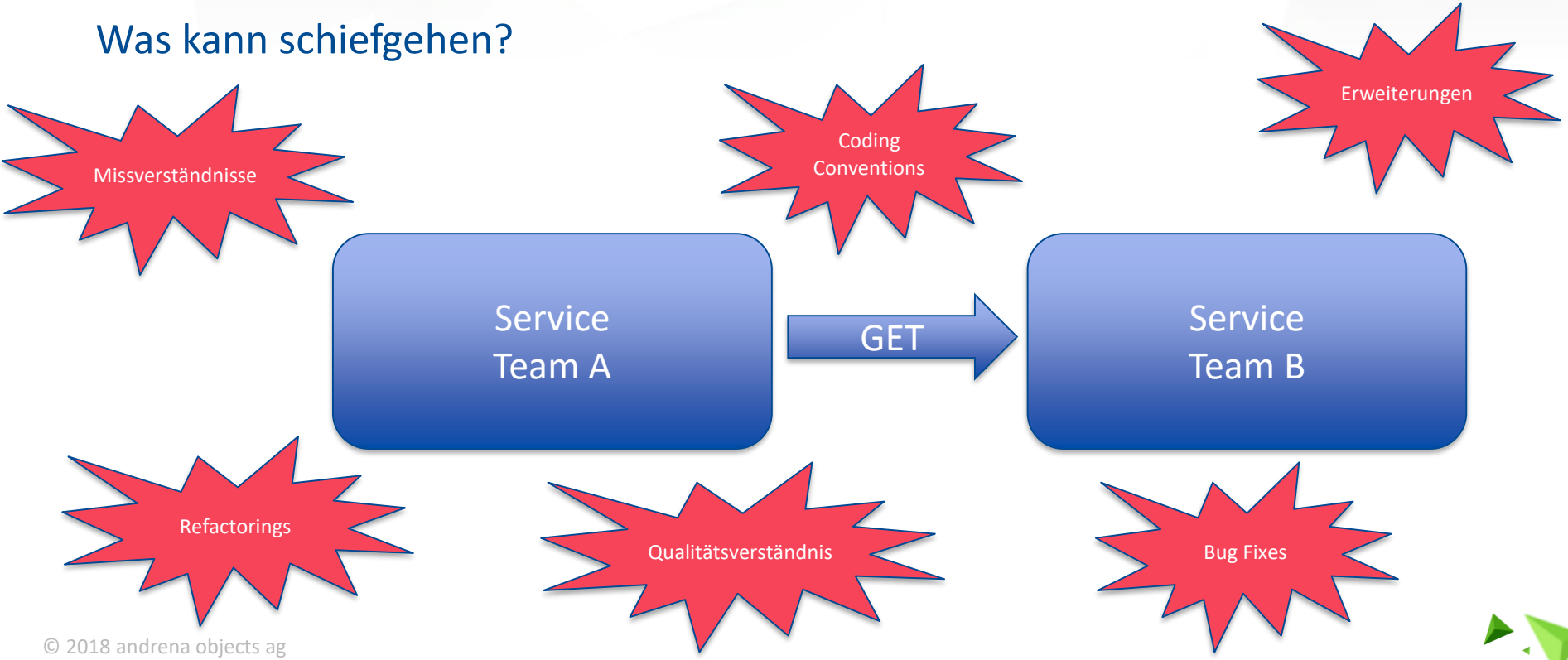
Beispiele aus der Praxis

- Funktionale Anpassungen



Motivation - Schnittstellen zwischen Teams

Was kann schiefgehen?



Motivation

Wie können Schnittstellen stabil gehalten werden?

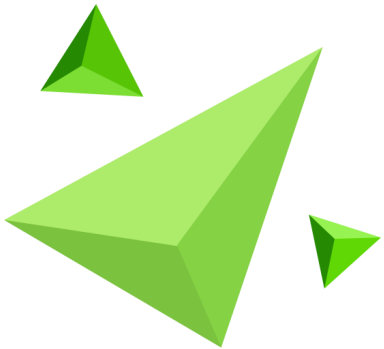
- Mündlich
- Schriftlich
- Automatisiert?

→ Consumer Driven Contract Testing!

andrena
OBJECTS

Consumer Driven Contract Testing

Was ist das?



Consumer Driven Contract Testing

Allgemein

- Konzept besteht schon lange
- im Microservice-Umfeld an Präsenz gewonnen
- Vertrag zwischen Konsument und Produzent einer Schnittstelle
- Produzent führt Tests des Konsumenten aus

Consumer Driven Contract Testing

Pact - Überblick

- Tool für Contract Testing
- Technologieunabhängig
- Bibliotheken für verschiedene Sprachen (z.B. C#, Javascript, Java...)
- Vertrag wird dynamisch durch Unittests erstellt

Consumer Driven Contract Testing

Pact - Terminologie

- Service Consumer
- Service Provider
- Mock Service Provider
- Interaction
- Pact File
- Pact Verification
- Provider State
- Pact Specification

Setzt einen HTTP-Request ab

Consumer Driven Contract Testing

Pact - Terminologie

- Service Consumer
- Service Provider
- Mock Service Provider
- Interaction
- Pact File
- Pact Verification
- Provider State
- Pact Specification

Consumer Driven Contract Testing

Pact - Terminologie

- Service Consumer
- Service Provider
- Mock Service Provider
- Interaction
- Pact File
- Pact Verification
- Provider State
- Pact Specification

Dient als Mock im Consumer Test

Consumer Driven Contract Testing

Pact - Terminologie

- Service Consumer
- Service Provider
- Mock Service Provider
- Interaction
- Pact File
- Pact Verification
- Provider State
- Pact Specification

Kombination aus Anfrage und Antwort

Consumer Driven Contract Testing

Pact - Terminologie

- Service Consumer
- Service Provider
- Mock Service Provider
- Interaction
- Pact File
- Pact Verification
- Provider State
- Pact Specification

JSON-Datei, die serialisierte Interactions enthält

Consumer Driven Contract Testing

Pact - Terminologie

- Service Consumer
- Service Provider
- Mock Service Provider
- Interaction
- Pact File
- Pact Verification
- Provider State
- Pact Specification

Vergleich Pact File mit tatsächlicher Antwort

Consumer Driven Contract Testing

Pact - Terminologie

- Service Consumer
- Service Provider
- Mock Service Provider
- Interaction
- Pact File
- Pact Verification
- Provider State
- Pact Specification

Zustand, der vom Consumer definiert und vom Provider bereitgestellt wird

Consumer Driven Contract Testing

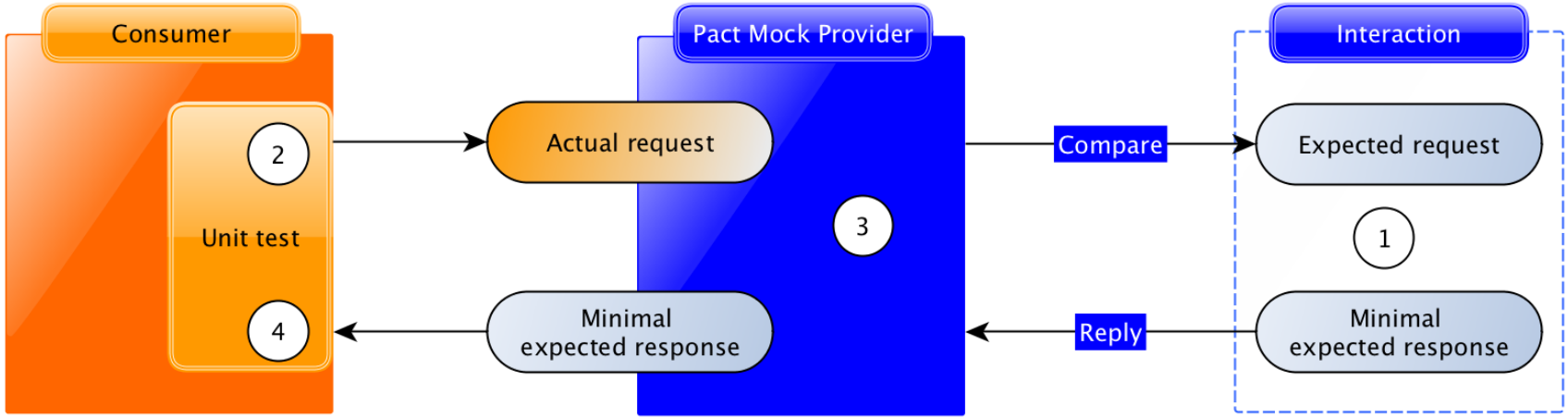
Pact - Terminologie

- Service Consumer
- Service Provider
- Mock Service Provider
- Interaction
- Pact File
- Pact Verification
- Provider State
- Pact Specification

Beschreibt die plattformunabhängige Struktur der JSON-Datei

Consumer Driven Contract Testing

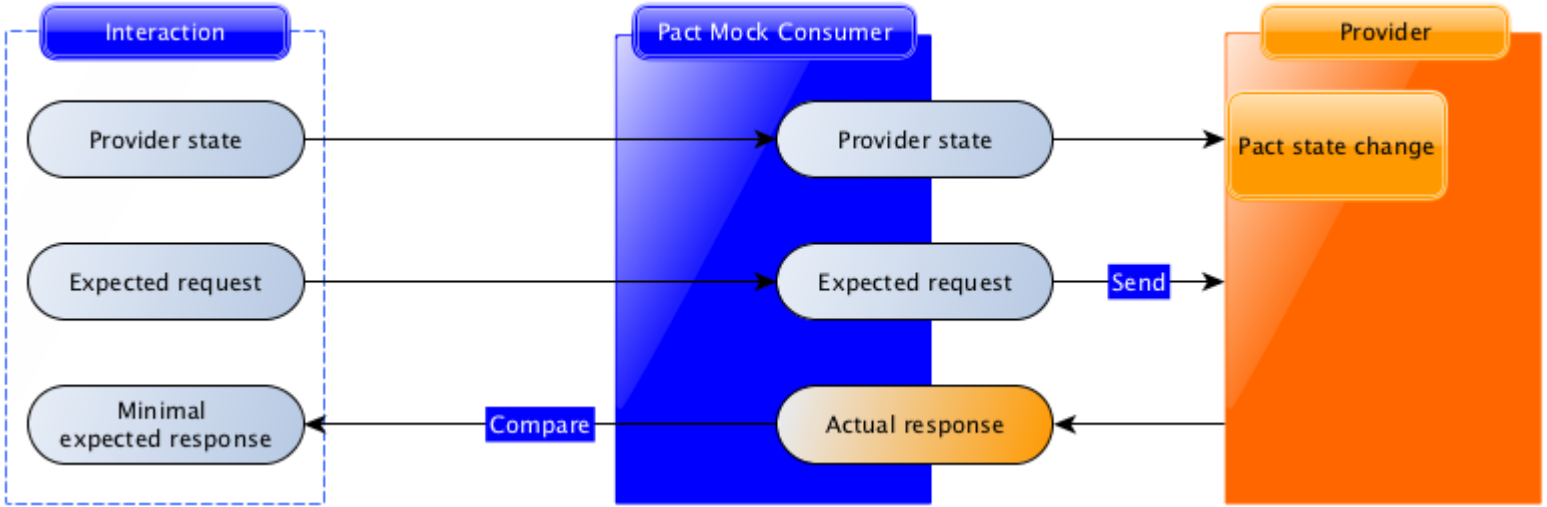
Pact - Test bei Konsument



© 2015 Pact Foundation

Consumer Driven Contract Testing

Pact - Test bei Produzent



Consumer Driven Contract Testing

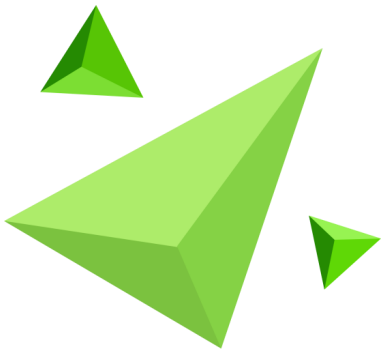
Pact Broker

Pacts

Consumer	Provider	Latest pact published	Last verified
Foo	Animals	2 minutes ago	2 days ago
Foo	Bar	7 days ago	15 days ago ⚠
Foo	Hello World App	1 day ago	
Foo	Wiffles	less than a minute ago	7 days ago
Some other app	A service	26 days ago	less than a minute ago
The Android App	The back end	less than a minute ago	

© 2013 Bethany Skurrie

Einführung im Projekt

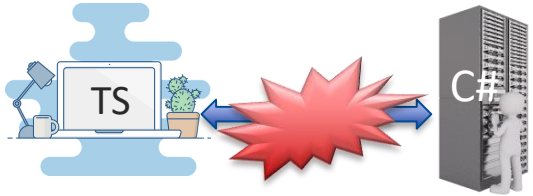


Einführung im Projekt

Untersuchung - Einsatzgebiete:

Im Team:

- Server-Client-Schnittstellen (Provider in C#, Consumer in Typescript)



- Zwischen Microservices (Provider und Consumer in C#)

Einführung im Projekt

Untersuchung - Einsatzgebiete:

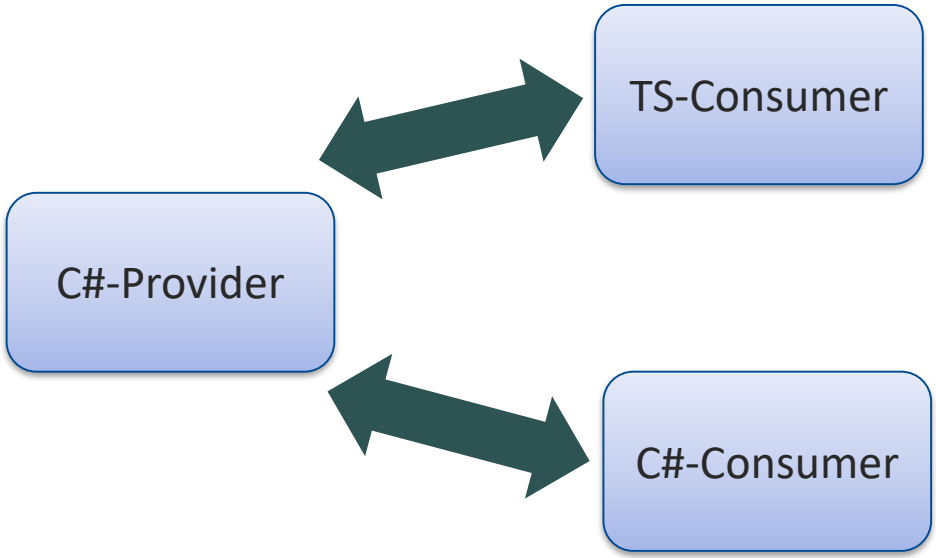
Zwischen unterschiedlichen Teams:

- Grenzen zwischen Microservices (Provider und Consumer in C#)

- Server-Client-Schnittstelle (Provider in C#, Consumer in Typescript)

Einführung im Projekt

Untersuchung - Einsatzgebiete:



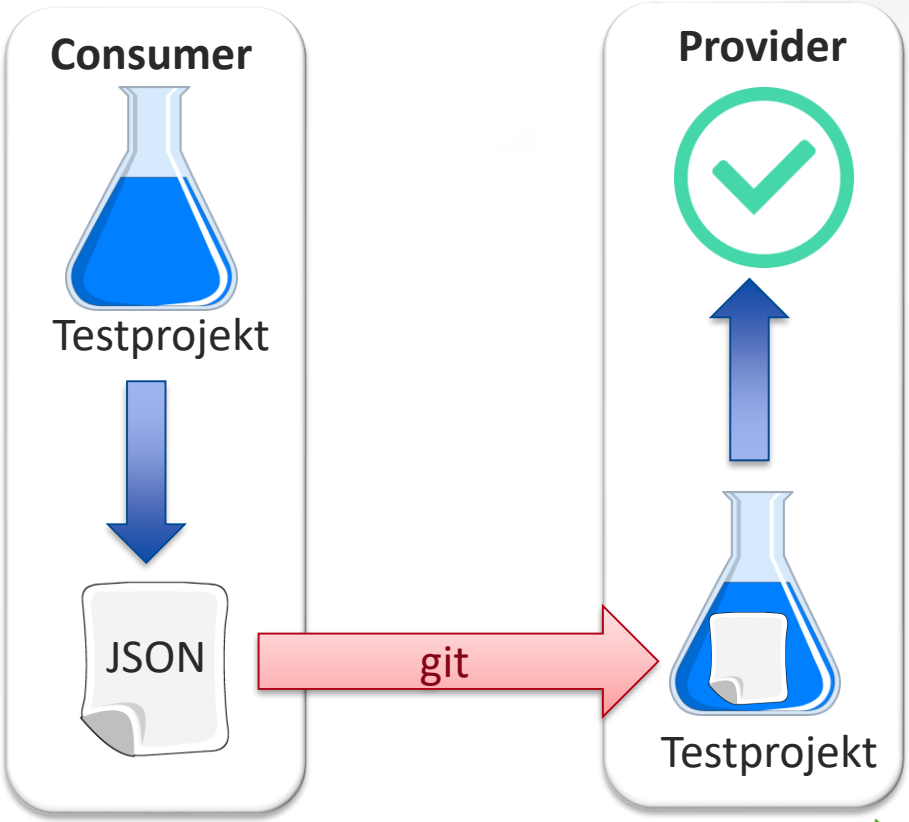
Einführung im Projekt

Untersuchung - Einsatzgebiete:

Zur Zeit noch nicht viele Teams an den untersuchten Schnittstellen beteiligt

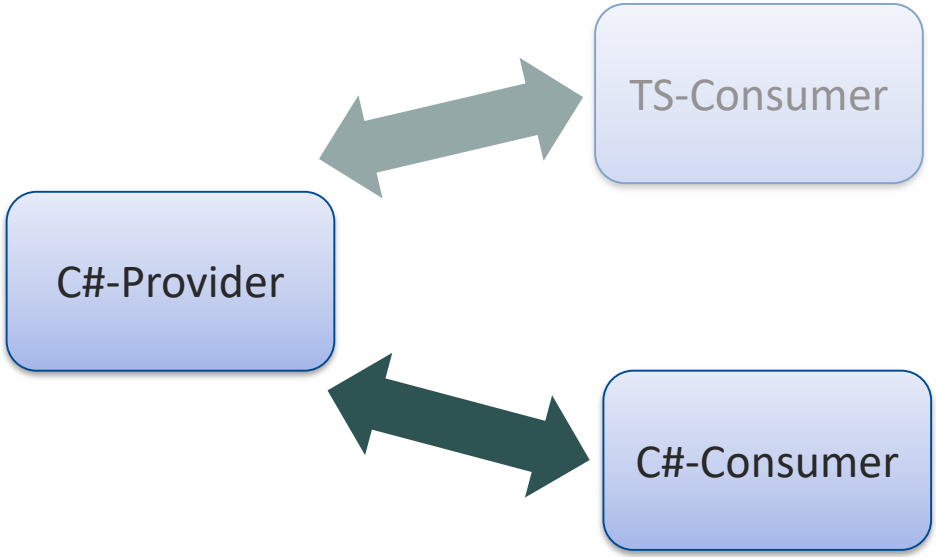
⇒ Noch keine Notwendigkeit für einen Pact-Broker.

⇒ Anstelle dessen: Austausch der **json**-Datei via **git** (so auch automatisiert)



Einführung im Projekt

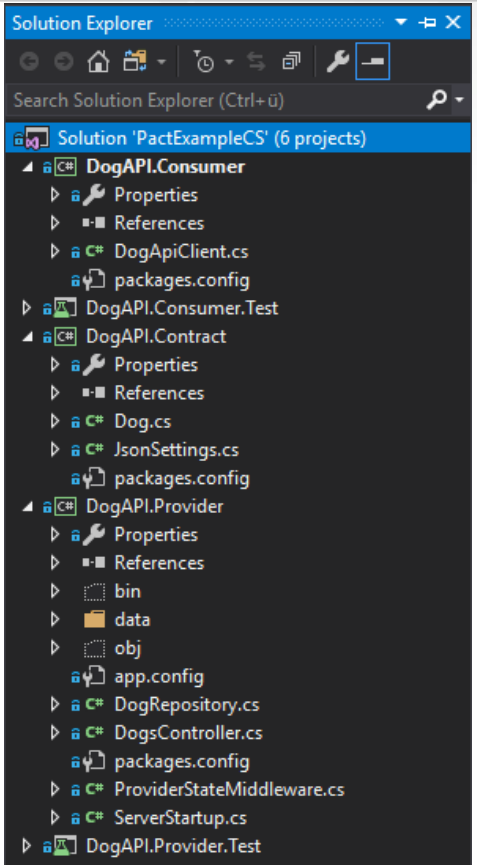
Untersuchung - Einsatzgebiete:



Einführung im Projekt

Untersuchung - Testprojekt in C#

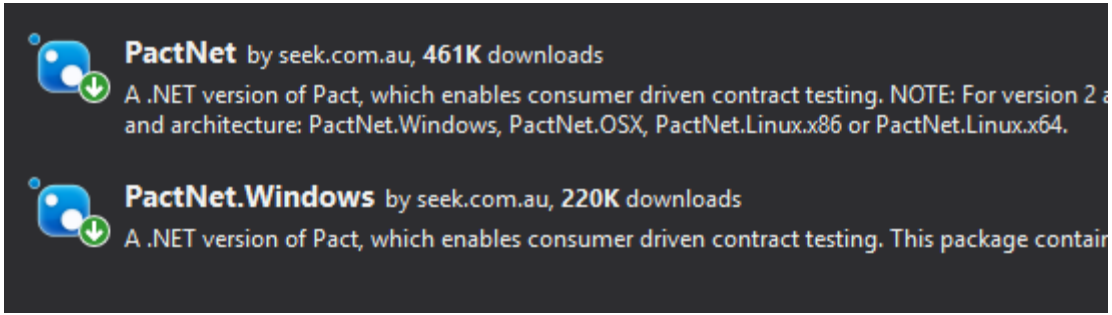
- DogAPI.Contract: Enthält Dog-Klasse und Json-Settings
- DogAPI.Provider: Server, der Liste von Hunden einliest und über eine REST-Schnittstelle zurückgibt (Owin)
- DogAPI.Consumer: Client, der Abfragen an die REST-Schnittstelle macht



Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von PactNet: Es müssen die nuget-Pakete PactNet und PactNet.Windows (auch unterschiedliche Linux oder OSX möglich) eingebunden werden



Einführung im Projekt

Untersuchung - Einsatzgebiete:

C#-Consumer

Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Consumer:
  - 1. Schritt: Instanziieren eines PactBuilders:

```
public DogApiProviderMockSetup()
{
      var currentDirectory = GetCurrentDirectory();
      PactBuilder = new PactBuilder(new PactConfig {
            SpecificationVersion = "2.0.0",
            PactDir = currentDirectory + @"\..\..\..\pacts",
            LogDir = currentDirectory + @"\..\..\..\logs" });
      PactBuilder.ServiceConsumer("Dog API")
                  .HasPactWith("C# Client");

      MockProviderService = PactBuilder.MockService(MockServerPort);
 }
```


Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Consumer:
  - 2. Schritt: Service Mock hoch- und herunterfahren

```
[OneTimeSetUp]
public void Setup()
{
      . . .
       _mockProviderService = new DogApiProviderMockSetup().MockProviderService;
       _mockProviderService.ClearInteractions(); //stelle sicher, dass nur die später konfigurierten
                                                 //Interactions berücksichtigt werden

}
[OneTimeTearDown]
public void TearDown()
{
        _dogApiMockSetup.ShutdownServerAndSaveToFile();
}
```

```
public void
ShutdownServerAndSaveToFile()
{
      PactBuilder.Build();
}
```

Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Consumer:
  - 3. Schritt: Definition einer Interaction

```
_mockProviderService.Given("There is a dog with id '1'")
      .UponReceiving("A GET request to retrieve the dog")
       .With(new ProviderServiceRequest{
           Method = HttpVerb.Get,
           Path = "/dogs/1",
           Headers = new Dictionary<string, object>{{ "Accept", "application/json" }}
        })
       .WillRespondWith(new ProviderServiceResponse{
           Status = 200,
           Body = Match.Type(testDog.ConvertToRestResponseBodyItem()),
           Headers = new Dictionary<string, object>{{
                      "Content-Type", "application/json; charset=utf-8" }}
       });
```


Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Consumer:
  - 4. Schritt: Test der Client-Methode.

```
var consumer = new DogApiClient($"http://localhost:{MockServerPort}");

var result = consumer.GetDog(testDog.Id);

Assert.That(result.Id, Is.EqualTo(testDog.Id));
Assert.That(result.Name, Is.EqualTo(testDog.Name));
Assert.That(result.Race, Is.EqualTo(testDog.Race));

_mockProviderService.VerifyInteractions();
```


Einführung im Projekt

```
MockProviderService = PactBuilder.MockService(MockServerPort);
```

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Consumer:
  - 4. Schritt: Test der Client-Methode

```
var consumer = new DogApiClient($"http://localhost:{MockServerPort}");
var result = consumer.GetDog(testDog.Id);

Assert.That(result.Id, Is.EqualTo(testDog.Id));
Assert.That(result.Name, Is.EqualTo(testDog.Name));
Assert.That(result.Race, Is.EqualTo(testDog.Race));

_mockProviderService.VerifyInteractions();
```


Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Consumer:
  - 4. Schritt: Test der Client-Methode

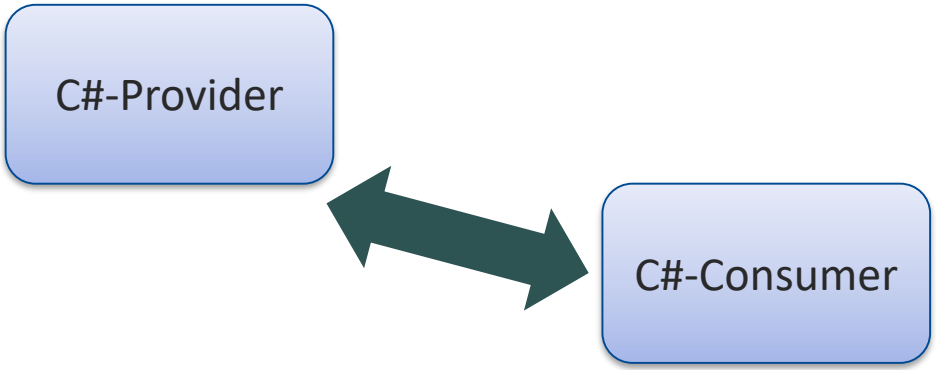
```
var consumer = new DogApiClient($"http://localhost:{MockServerPort}");
var result = consumer.GetDog(testDog.Id);

Assert.That(result.Id, Is.EqualTo(testDog.Id));
Assert.That(result.Name, Is.EqualTo(testDog.Name));
Assert.That(result.Race, Is.EqualTo(testDog.Race));

_mockProviderService.VerifyInteractions();
```


Einführung im Projekt

Untersuchung - Einsatzgebiete:



Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Provider:
  - 1. Schritt: Instanziieren eines PactVerifiers:

```
                                                    PactBuilder.ServiceConsumer("C# Consumer")
                                                                .HasPactWith("Dog API");
using (WebApp.Start<ServerStartup>(serviceUri)){
      IPactVerifier pactVerifier = new PactVerifier(config);
      pactVerifier.ProviderState($"{serviceUri}/provider-states")
                  .ServiceProvider("Dog API", $"http://localhost:{MockServerPort}")
                  .HonoursPactWith("C# Consumer")
                  .PactUri(DatabaseFileReader.GetCurrentDirectory()
                                  + @"\..\..\..\pacts\consumer-dog_api.json");
            …
}
```


Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Provider:
  - 1. Schritt: Instanziieren eines PactVerifiers:

```
                                        MockProviderService = PactBuilder.MockService(MockServerPort);

using (WebApp.Start<ServerStartup>(serviceUri)){
      IPactVerifier pactVerifier = new PactVerifier(config);
      pactVerifier.ProviderState($"{serviceUri}/provider-states")
                  .ServiceProvider("Dog API", $'http://localhost:{MockServerPort}')
                  .HonoursPactWith("C# Consumer")
                  .PactUri(DatabaseFileReader.GetCurrentDirectory()
                                  + @"\..\..\..\pacts\consumer-dog_api.json");
            …
}
```


Einführung im Projekt

Untersuchung - Testprojekt in C#

- Einbindung von Pact.Net im Provider:
  - 1. Schritt: Instanziieren eines PactVerifiers:

```
using (WebApp.Start<ServerStartup>(serviceUri)){
      IPactVerifier pactVerifier = new PactVerifier(config);
      pactVerifier.ProviderState($"{serviceUri}/provider-states")
                  .ServiceProvider("Dog API", $"http://localhost:{MockServerPort}")
                  .HonoursPactWith("C# Consumer")
                  .PactUri(DatabaseFileReader.GetCurrentDirectory()
                                    + @"\..\..\..\pacts\consumer-dog_api.json")
            …
}
```


Einführung im Projekt

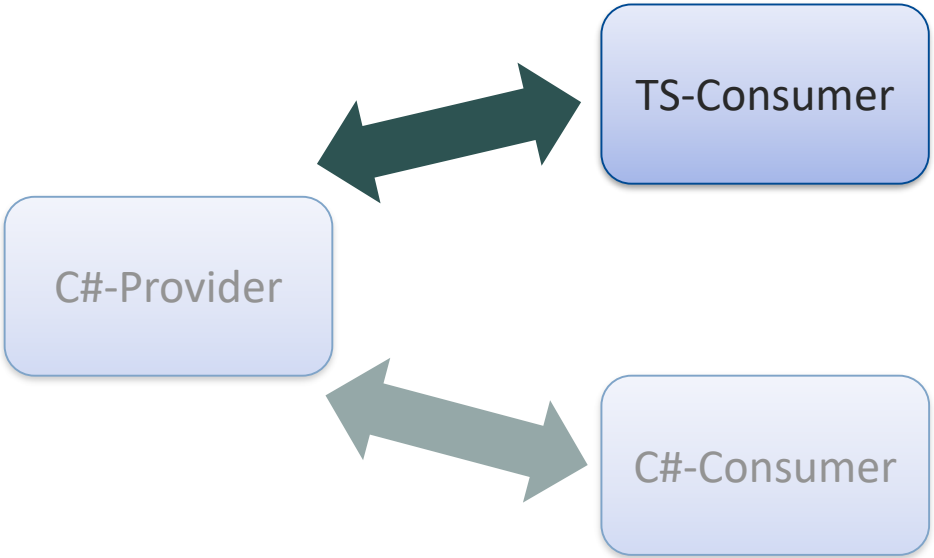
Untersuchung -Testprojekt in C#

- Einbindung von Pact.Net im Provider:
  - 2. Schritt: Verifizieren des Contracts:

```
using (WebApp.Start<ServerStartup>(serviceUri)){
        …
        pactVerifier.Verify();
}
```


Einführung im Projekt

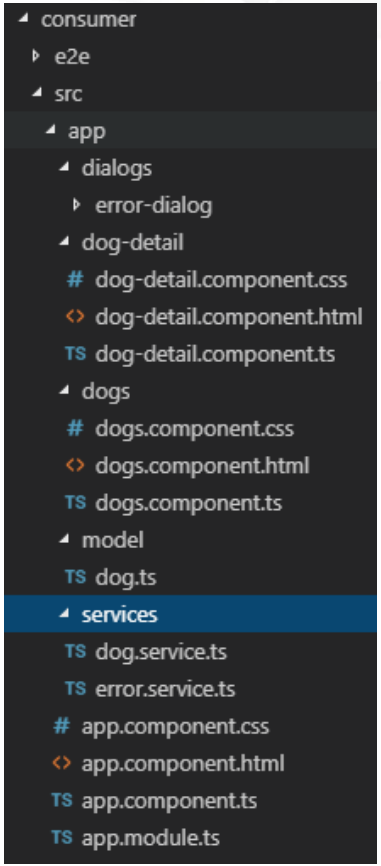
Untersuchung - Einsatzgebiete:



Einführung im Projekt

Untersuchung - Testprojekt in Typescript

- Consumer: Master-Detail-App, die eine Liste von Hunden und Details zu einem ausgewählten Hund anzeigt (Angular).



Einführung im Projekt

Untersuchung - Testprojekt in Typescript

- Einbindung von Pact (Testrunner Karma):
  - 1. Schritt: NPM-Pakete einbinden.

```
"devDependencies": {
    …
    "@pact-foundation/karma-pact": "^2.2.0",
    "@pact-foundation/pact-node": "^8.1.2",
    "@pact-foundation/pact-web": "^8.2.0",
    …
    "typescript": "~3.2.2"
  }
```

Achtung: Unterschiede bei Testrunnern (Hier: Karma)

Einführung im Projekt

Untersuchung - Testprojekt in Typescript

- Einbindung von Pact (Testrunner Karma):
  - 2. Schritt: Instanziieren eines Pact-Provider-Mocks

```
beforeAll((done) => {
      this.provider = new PactWeb(pactConfig);
      setTimeout(done, 2000);
      this.provider.removeInteractions();
 });
…
afterAll((done) => {
    this.provider.finalize()
         .then(() => done())
         .catch((error) => done.fail(error));
});
```


Einführung im Projekt

Untersuchung - Testprojekt in Typescript

- Einbindung von Pact (Testrunner Karma):
  - 3. Definition einer Interaction:

```
public addInteraction = (interaction: InteractionObject, done)
=> {
      this.provider.addInteraction(interaction)
                   .then(() => done())
                   .catch((error) => done.fail(error));
}

 public create = (): InteractionObject => {
       …
 }
```

Einführung im Projekt

Untersuchung - Testprojekt in Typescript

- Einbindung von Pact (Testrunner Karma):
  - 3. Definition einer Interaction:

```
public create = (): InteractionObject => {
      return {
                  state: ”State name”,
                  uponReceiving: ”State name”,
                  withRequest: this.request,
                  willRespondWith: this.response,
                }
}
```


Einführung im Projekt

Untersuchung - Testprojekt in Typescript

- Einbindung von Pact (Testrunner Karma):
  - 3. Definition einer Interaction:

```
public create = (): InteractionObject => {
      return {
                  state: ”State name”,
                  uponReceiving: ”State name”,
                  withRequest: this.request,
                  willRespondWith: this.response,
                }
}
```

```
this.request = {
      method: "GET",
       path: "/dogs",
       headers: this.requestHeaders
}
```

Einführung im Projekt

Untersuchung - Testprojekt in Typescript

- Einbindung von Pact (Testrunner Karma):
  - 3. Definition einer Interaction:

```
public create = (): InteractionObject => {
      return {
                  state: ”State name”,
                  uponReceiving: ”State name”,
                  withRequest: this.request,
                  willRespondWith: this.response
                }
}
```

```
this.resonse = {
      status: 200,
      headers: this.headers,
      body: this.listOfDogs
}
```


Einführung im Projekt

Untersuchung - Testprojekt in Typescript

- Einbindung von Pact (Testrunner Karma):
  - 4. Test der Client-Methode:

```
let dogService = TestBed.get(DogService);

const requestResultVerifier = (dogs) => {
      expect(dogs).toEqual(testDogs);
      this.provider.verify();
      done();
}

dogService.getDogList().subscribe(requestResultVerifier);
```


Einführung im Projekt

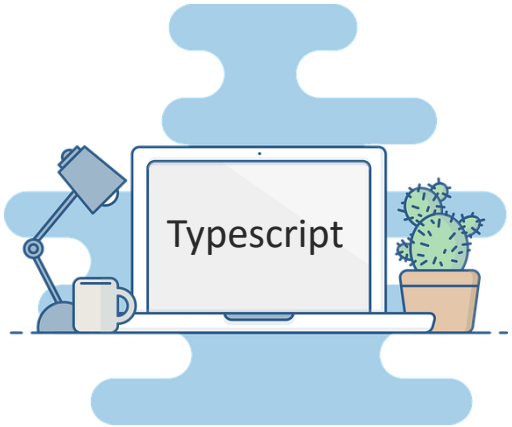
Untersuchung – Verbinden von Typescript und C#

- Zuätzlicher Testcase in C# Provider: Vertrag des Typescript-Clients einbinden.

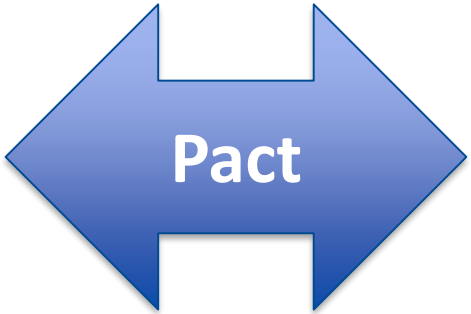
```
public void DogsApiWorksWithTsConsumerTests()
{
      …
       using (WebApp.Start<ServerStartup>(serviceUri))
       {
           new PactVerifier(config).ProviderState($"{serviceUri}/provider-states")
                    .ServiceProvider("Dog API", serviceUri)
                    .HonoursPactWith(„Typescript Consumer")
                    .PactUri(DatabaseFileReader.GetCurrentDirectory()
                           + @"\contracts\typescriptClient.json")
                    .Verify();
        }
}
```

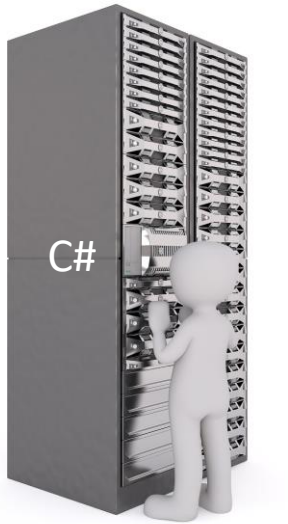
- Automatisierte Aktualisierung der Datei: Über git.

Beispiele für konkrete Problemlösungen



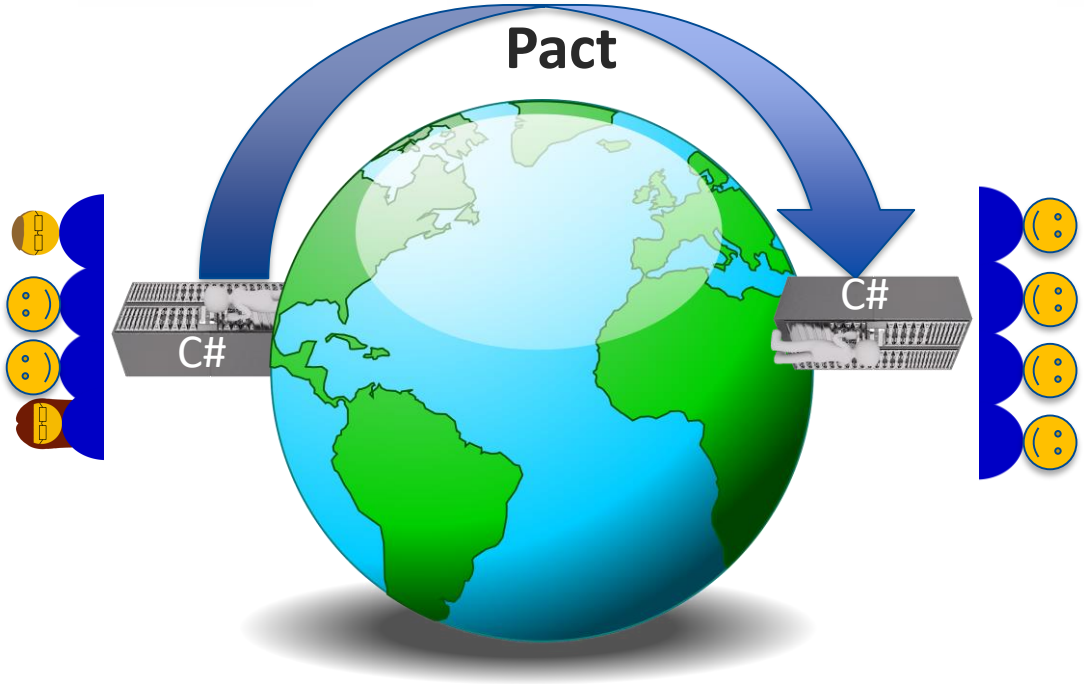
Client





Server

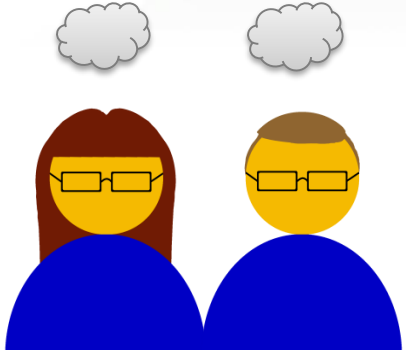
Beispiele für konkrete Problemlösungen



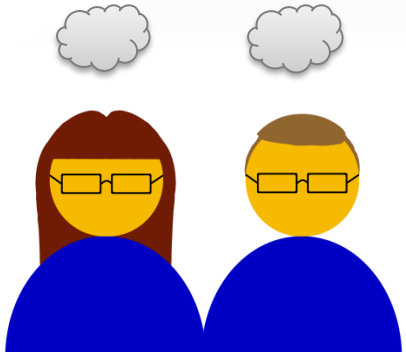
Beispiele für konkrete Problemlösungen

Schwierigkeiten:

- Akzeptanz für Mehraufwand schaffen
- Mehrere Teams involviert
- Austausch der Verträge



Beispiele für konkrete Problemlösungen



Schwierigkeiten:

- C#: Ruby-Prozess wird manchmal nicht beendet => nächster Testdurchlauf wird rot
- Typescript: Bei komplexerer Netzwerkstruktur update der npm-Packages schwer
- Typescript: z.T. keine Typisierung in verfügbaren Beispielen => Aufräumarbeiten für Verständnis notwendig
- Typescript: Unterschiedliche Abhängigkeiten für unterschiedliche Testrunner (keine dediziert für unseren -> keine Lösung 😢) => Forschungsarbeit notwendig

Fazit

- Gewisse Einstiegshürde
- Firewallkonfigurationen in Unternehmen können hinderlich sein
- CDC hat sehr viel Potential

- Wir werden Pact weiter nutzen!

andrena
OBJECTS

Bitte geben Sie uns jetzt Ihr Feedback!

Mit Pact REST-Schnittstellen innerhalb und außerhalb des Teams definieren und stabilisieren
Eva Ziebarth, Marvin Kranz

Nächste Vorträge in diesem Raum

13:30 Event Sourcing - Wahrscheinlich machen Sie es falsch, *David Schmitz*
14:30 Verbesserte Observability unter Verwendung offener, OpenCensus-basierter Application-Monitoring-Lösungen, *Dr. Alexander Wert*
15:45 Size does matter! Why and how you should use JVM-Microframeworks, *Christian Schwörer*

Referenzen

https://docs.pact.io/
https://github.com/pact-foundation/pact-net
https://github.com/pact-foundation/pact_broker
https://martinfowler.com/articles/consumerDrivenContracts.html