Karlsruhe Institute of Technology

# A Test Concept for the Development of Microservice-based Applications

## ICSEA 2021
## 03-07 October 2021 in Barcelona, Spain

COOPERATION & MANAGEMENT (C&M, PROF. ABECK), INSTITUTE OF TELEMATICS, DEPARTMENT OF INFORMATICS

**Michael Schneider**[1]
Stephanie Zieschinski[1]
Hristo Klechorov[1]
Lukas Brosch[1]
Patrick Schorsten[1]
Sebastian Abeck[1]
Christof Urbaczek[2]

**Contact: michael.schneider@kit.edu**

[1] Research Group Cooperation & Management, Karlsruhe Institute of Technology, Germany
[2] xdi360 GmbH, Leopoldstraße 252b, 80807 München, Germany

# About Me

(1) Before studies
    (1) Completed training as an electronics technician for devices and systems

(2) Master degree at the Karlsruhe Institute of Technology (KIT) in the field of computer science

(3) Doctoral student at KIT with the following topics
    (1) Systematic development of microservices-based applications
    (2) Internet of Things
    (3) Testing of microservice-based systems

Contact: michael.schneider@kit.edu

# Motivation

(1) Microservice-based applications are composed of several distributed microservices
  - (1) Can be developed separately by different development teams
  - (2) Testing the whole application becomes far more complex

(2) Requires a systematic development approach
  - (1) Testing an application itself has to consider the whole test pyramid
  - (2) Needs to be integrated into the microservice-based development process

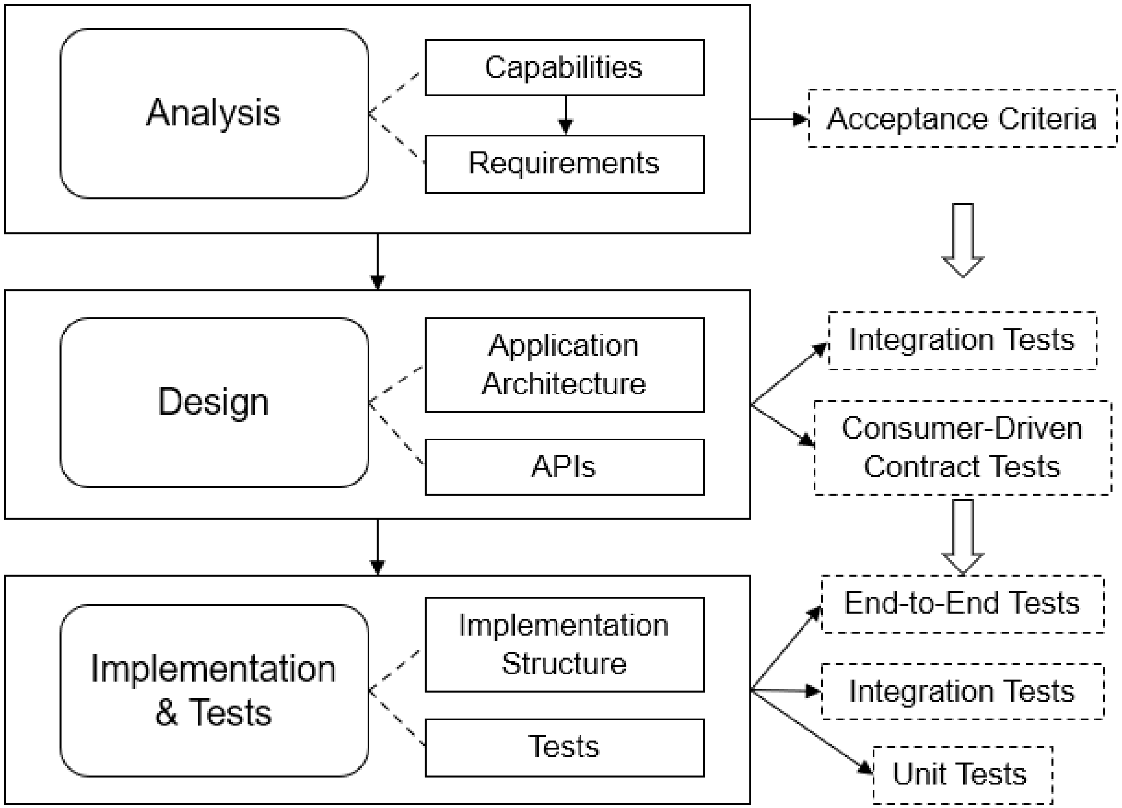
(3) Several problems arise without a process
  - (1) Testing is handled differently for each developed application
  - (2) Systematic approach is missing
  - (3) Developers require assistance and guidelines for well-tested microservice applications

# Development Process and Test Artifacts

(1) Tests must be considered in each phase

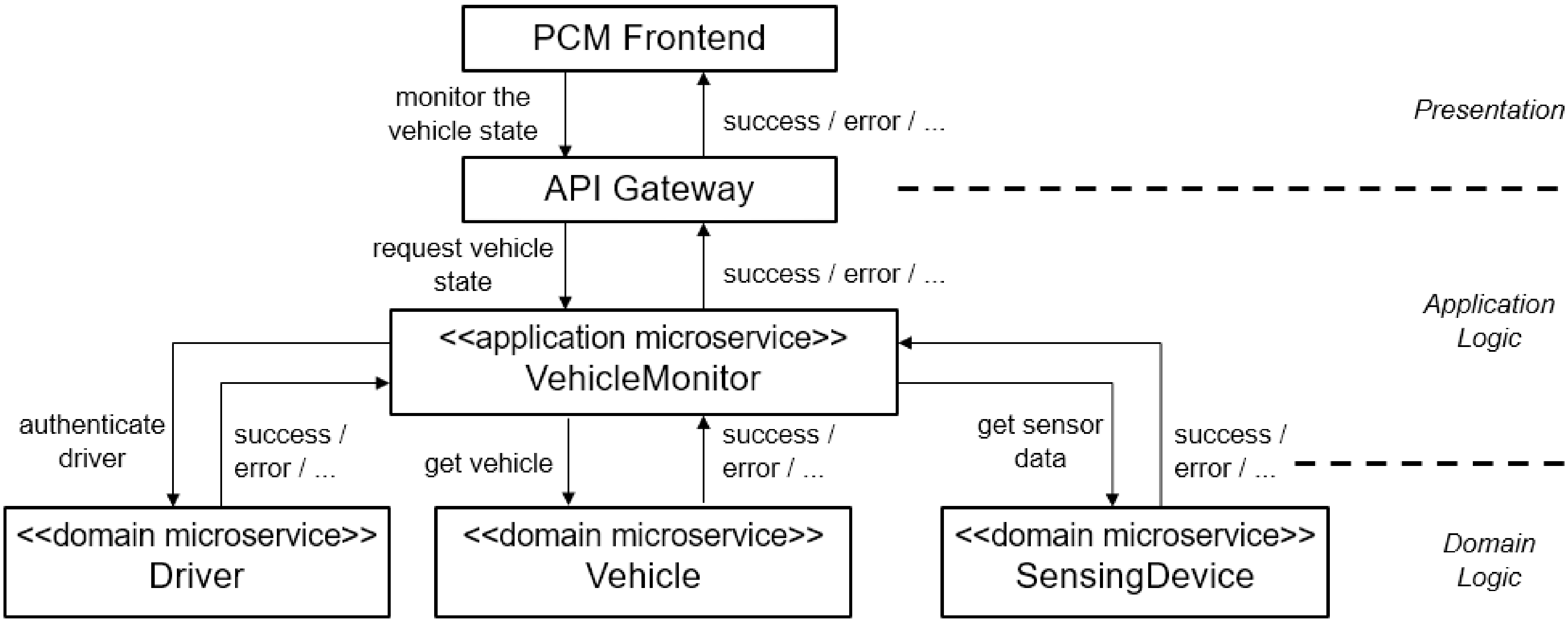
(2) The process needs to support the testing by providing the necessary artifacts

(3) It should be made clear what needs to be tested



# Example: Predictive Car Maintenance Application Architecture Overview

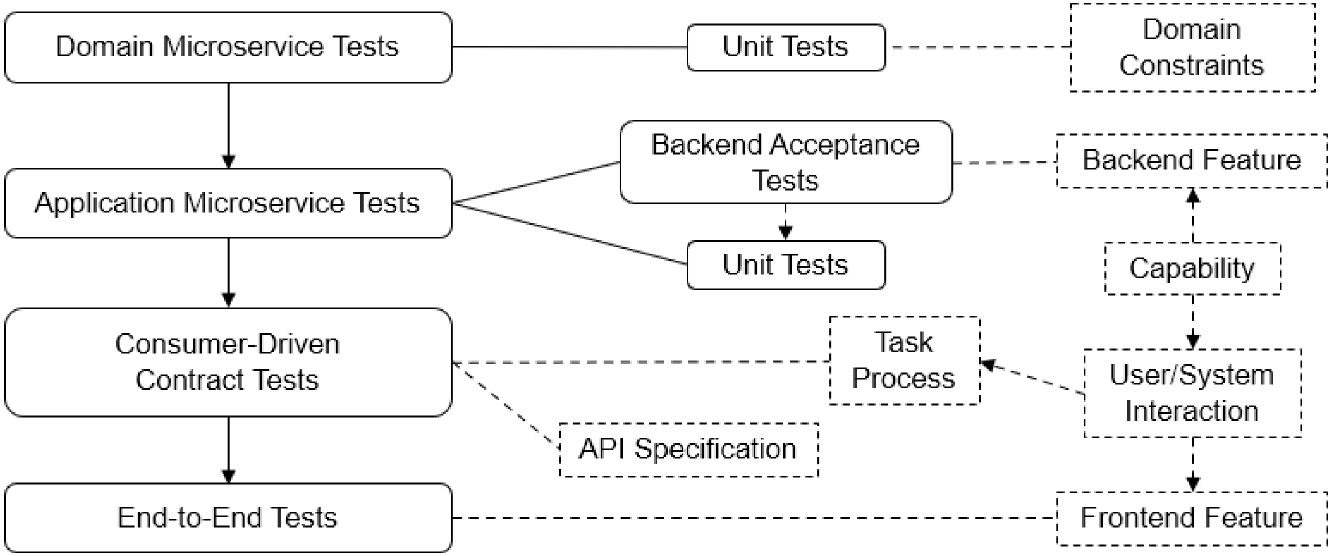
- Application consists of multiple microservices
  - Differentiation of domain and application microservices



# Overview of the Test Concept

(1) Development of tests follows a logical order
  (1) Starting with the domain microservices to the application microservices
  (2) Testing the APIs
  (3) Testing the whole microservice-based system with end-to-end tests

(2) Tests utilize the development artifacts





Cooperation & Management (C&M, Prof. Abeck)
Institute of Telematics, KIT Department of Informatics

# Excerpt of Domain Constraint Testing

Karlsruhe Institute of Technology

(1) Constraints are derived from the domain knowledge and express part of the domain logic which needs to be implemented

(2) Test cases are derived from the constraints

  (1) Valid and invalid cases need to be tested

***Constraints***

```
context VehicleComponent::
  getObservationsFromTimePeriod(start:Date, end:
  Date):
pre: start.before(end)
post: forAll(o:Observation | (start.after(start)
  or start.equals(start))
and (end.before(end) or end.equals(end)))
```

implemented

```
List<Observation> result = new ArrayList<>();
  for (Observation o : this.observations) {
      ZonedDateTime t = o.getTimeOfMeasurement();
      if((t.isAfter(start) || t.equals(start)) &&
          (t.isBefore(end) || t.equals(end))) {
          result.add(o);
      }
}
```

***Implementation***

tested by

```
List<Observation> result =
validTestComponent.
   getObservationsFromTimePeriod(start,end);
assertEquals(result, expectedResult);
```

***Test***

***Argument Provider***

```
OutputProvider op = new OutputProvider();
ZonedDateTime start = ZonedDateTime.now().
  minusMonths(1);
ZonedDateTime end = ZonedDateTime.now();
return Stream.of(
// Test Case 1: Last month
Arguments.of(start, end, op.getOutput(1))
);
```

initializes arguments

# Backend Features

(1) Application microservice is developed to support user/system interactions

(2) BDD outside-in approach is adopted for the development of acceptance tests

  (1) Specification of acceptance criteria

  (2) Transferred to backend acceptance tests

  (3) Additional unit tests to further test the implementation

(3) Tests the functionality independent from the frontend

(4) Faster execution of the tests

```
Scenario: Monitor Component State (Success)
Given the component with uuid "123..." exists
When the state of a component with uuid "123..."
    is requested
Then latest sensor information about the component
    is fetched
```

***Backend Scenario***

```
@When("When the state of a component with
    uuid <string> is requested")
public void request_component_info(String id
    ) throws Throwable {
List componentInfo = operations.
    getComponentInfo(id);
}
```

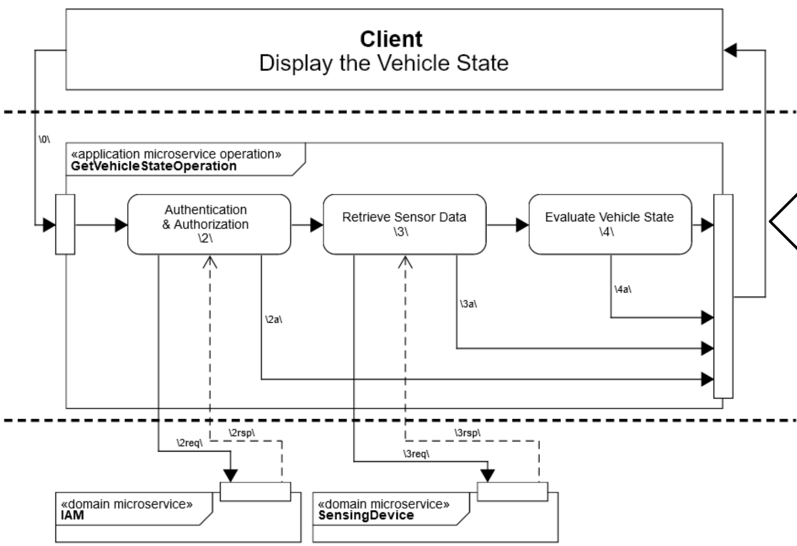
***Backend Acceptance Test Step Definition***

# Consumer-Driven Acceptance Tests

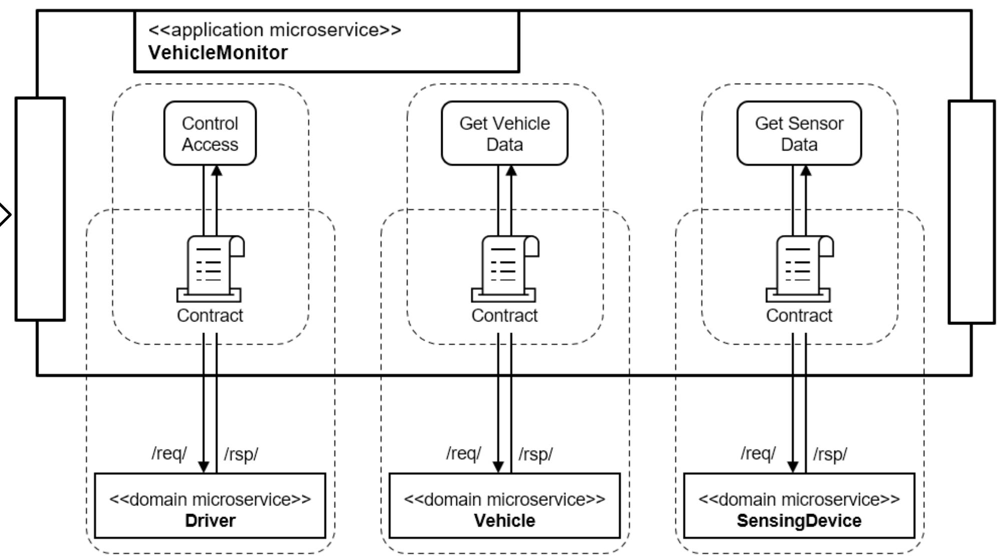
(1) Can decrease the number of integration tests

(2) Contracts document the communication between two services

- (1) Microservice under test is the consumer
- (2) Contracts are derived from the task process




*Task Process*




*Consumer-Driven Contracts*

# End-to-End Tests

(1) Whole application is tested using end-to-end tests
  - (1) User/system interactions form the basis for the derivation of the Gherkin scenarios

(2) Separate repository is used for end-to-end test
  - (1) Application is treated as a black box

(3) End-to-end tests should be robust against changes
  - (1) Guidelines for writing end-to-end tests

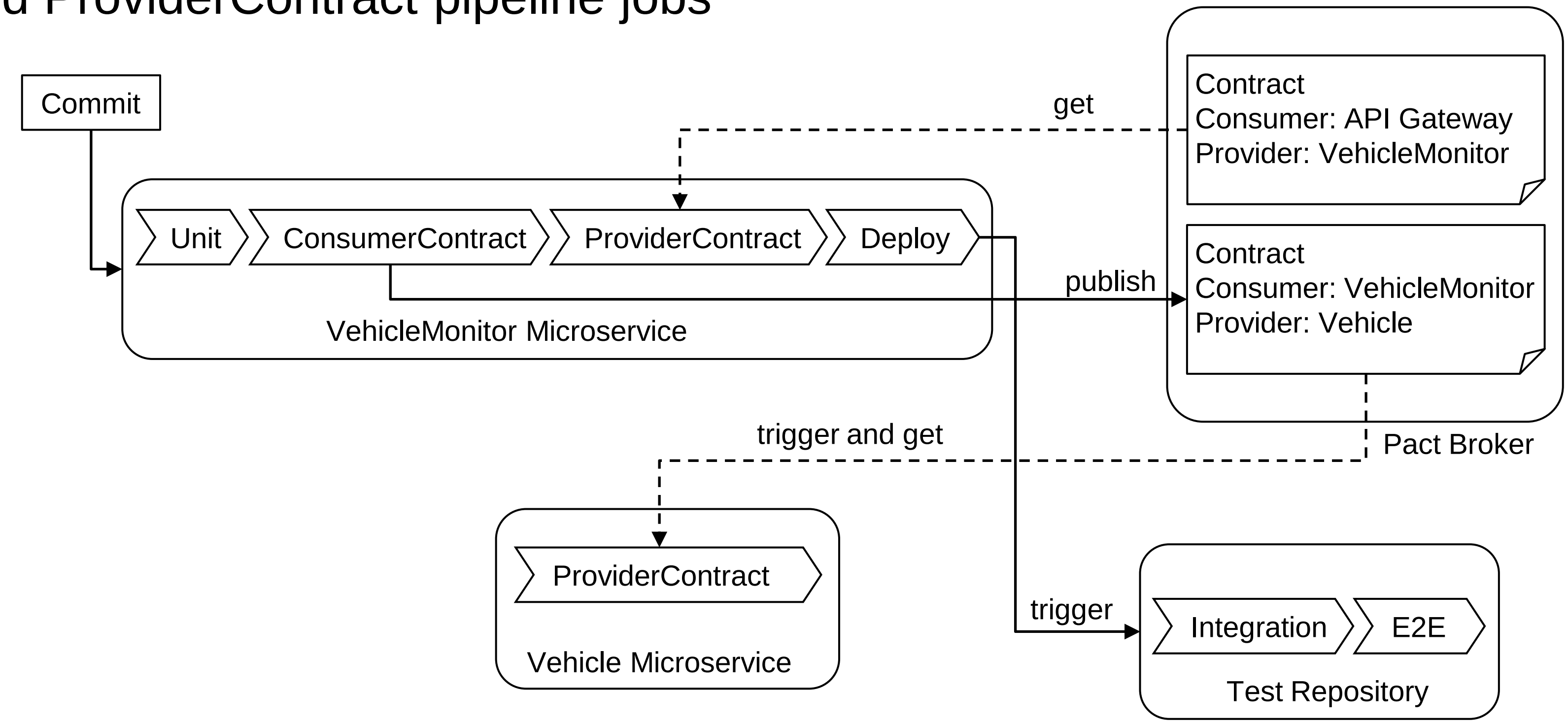
(4) Microservices need to be available
  - (1) Test instance required which contains all services and databases

```
1. Scenario: Monitor the Vehicle State (Success)
2. Given I am logged in as a vehicle owner
3. And the vehicle state overview is displayed
4. When I open the vehicle state overview for the motor
5. Then I see the detailed summary of the motor
```

# Pipeline and Cross-Triggering

(1) Pipeline considers all test types
  (1) Change in the microservice VehicleMonitor requires to trigger different tests

(2) Consumer-Driven Contract tests are split into separate ConsumerContract and ProviderContract pipeline jobs






Cooperation & Management (C&M, Prof. Abeck)
Institute of Telematics, KIT Department of Informatics

# Results, Conclusion, and Outlook

(1) A test concept for the development of microservice-based applications was introduced

(2) Test concept distinguishes between domain and application microservices
   - (1) Domain microservice requires testing the domain and its constraints
   - (2) Application microservice requires testing of the behavior

(3) Shifting the tests of lower layers reduces the execution time of tests

(4) CI/CD pipeline is used for executing the test on different stages

(5) Usage and refinement of the test concept by the development of further application

# Thank you for your Attention

Questions?