Property Based Testing with ScalaCheck

Enhance Your Unit Testing

Property-based testing on the Java Platform

ScalaCheck

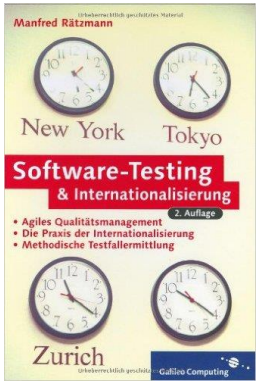
The Definitive Guide

artima

Rickard Nilsson

Manfred Rätzmann

“Nitpicker, mingle-minded, process freak, everyday philosopher, dude. Currently Head of QA at E-Post Development GmbH. Motivated by curiosity.“ @RaezzM



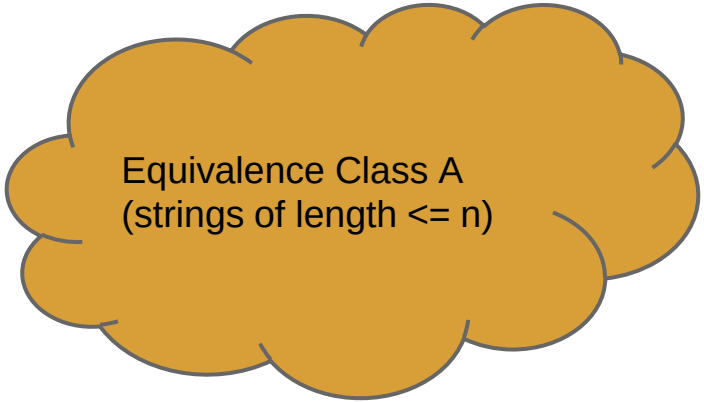
What‘s the difference between Classical Unit Testing and Property Based Testing ?

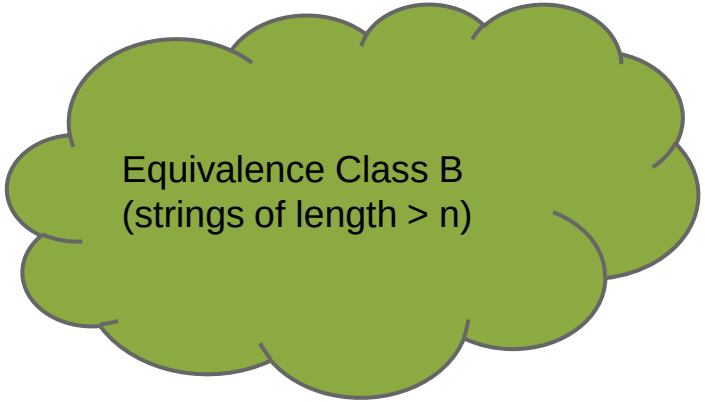
Classical Unit Testing

checks examples of the desired behaviour
derived from equivalence class partitioning

Class

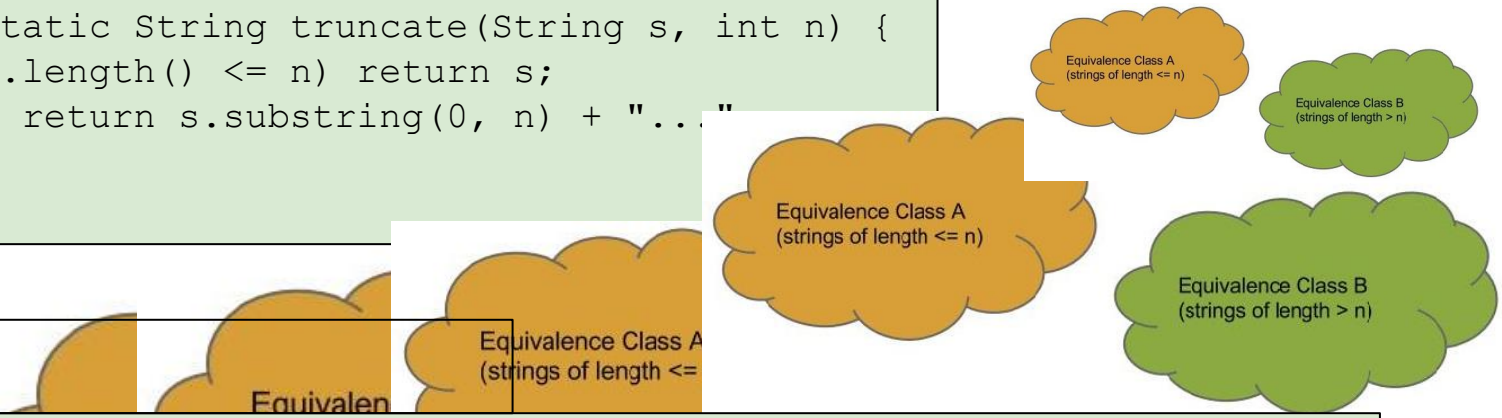
```
public class StringUtils {
    public static String truncate(String s, int n) {
        if(s.length() <= n) return s;
        else return s.substring(0, n) + "...";
    }
}
```





Class

```
public class StringUtils {
    public static String truncate(String s, int n) {
        if(s.length() <= n) return s;
        else return s.substring(0, n) + "...
    }
}
```



Test

```
public class StringUtilsTests {
    @Test public void aStringToShort_shouldBeReturnedCompletely() {
        String s = StringUtils.truncate("abc", 5);
        assertEquals("abc", s);
    }
    @Test public void aLongString_shouldBeTruncated() {
        String s = StringUtils.truncate("Hello World", 8);
        assertEquals("Hello Wo...", s);
    }
}
```

Property Based Testing

checks conformance with rules that define the desired behaviour

Example: A String Truncation Function

Property Based Testing checks for rule conformance using random data

natural language

Strings that exceed a given length n will be truncated to n chars plus “...”.

Example: A String Truncation Function

Property Based Testing checks for rule conformance using random data

Pseudocode

```
for all strings s and positive integers n :
if length of s <= n then return s
else return left n chars of s + "..."
```

Example: A String Truncation Function

Property Based Testing checks for rule conformance using random data

ScalaCheck

```
property("truncation") =
    forAll { (s: String, n: Int) =>
      val t = StringUtils.truncate(s, n)
      (s.length <= n && t == s) ||
        (s.length > n && t == s.take(n) + "...")
    }
```

Example: A String Truncation Function

Property Based Testing checks for rule conformance using random data

ScalaCheck

```
property("truncation") =
    forAll { (s: String, n: Int) =>
      val t = StringUtils.truncate(s, n)
      (s.length <= n && t == s) ||
        (s.length > n && t == s.take(n) + "...")
    }
```

Random Data Generation

Call function under test

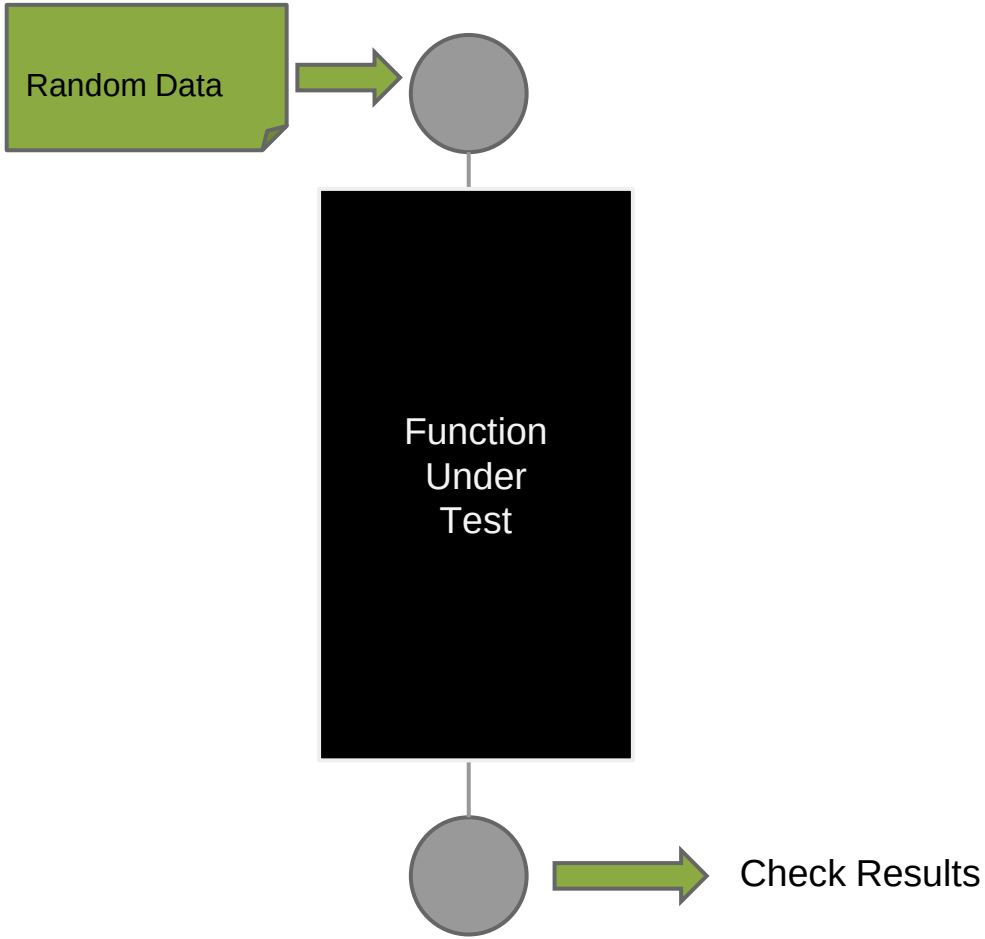
Check Property Rule

How to design suitable properties?

Uncomplete Properties

Ask what should never happen or
what must always hold true!

Sanity check



Checklist

- ❏ … must always …
- ❏ … must always …
- ❏ … should never …
- ❏ … should never …
- ❏ … should never …

….

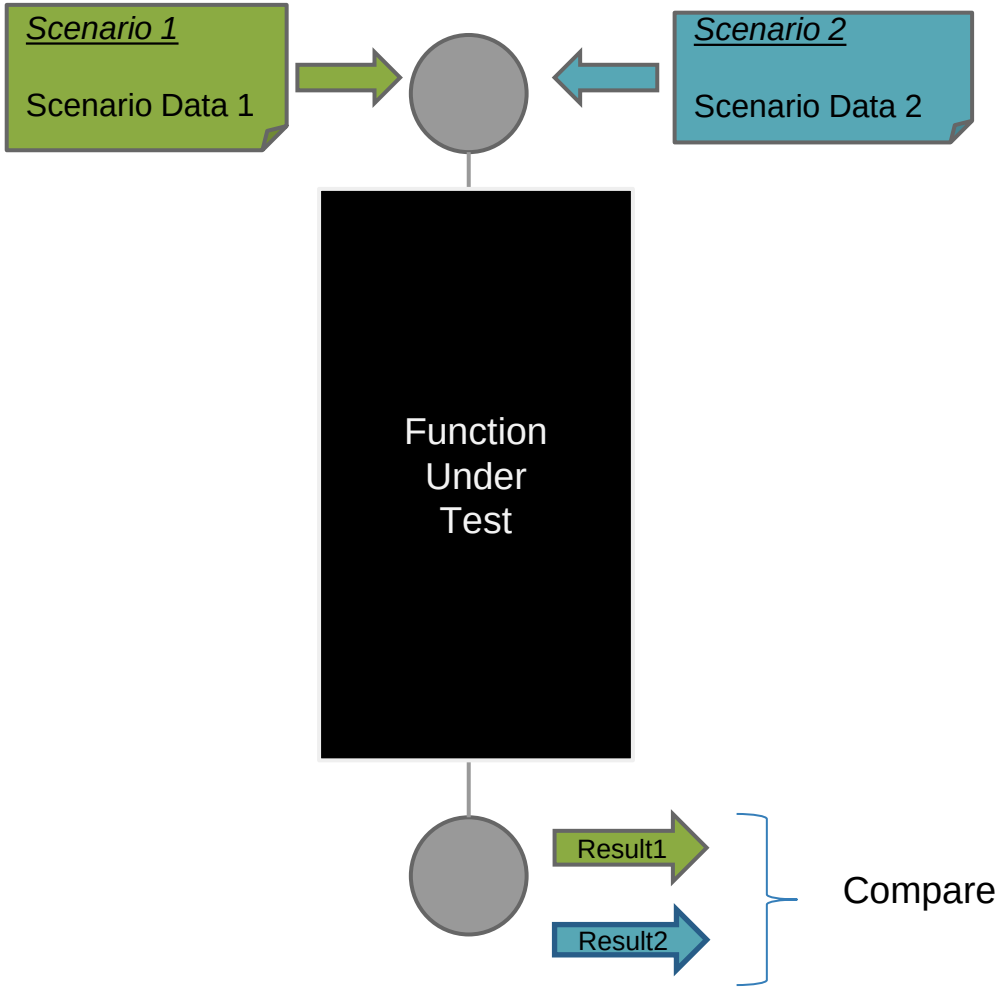
Sanity check

```
property("actually compressed") =
  forAll {s: String =>
    val result = myCompressor(s)
    result.length <= s.length
}
property("is monotonic") =
  forAll { (s1: String, s2: String) =>
    val r1 = myCompressor(s1)
    val r2 = myCompressor(s2)
  if (s1.length > s2.length) r1.length >= r2.length
  else r1.length <= r2.length
}
```

Relation Properties

Compare two results of a given function relating to its behavioural rules.

Relation properties



Compare Result1 related to Result2

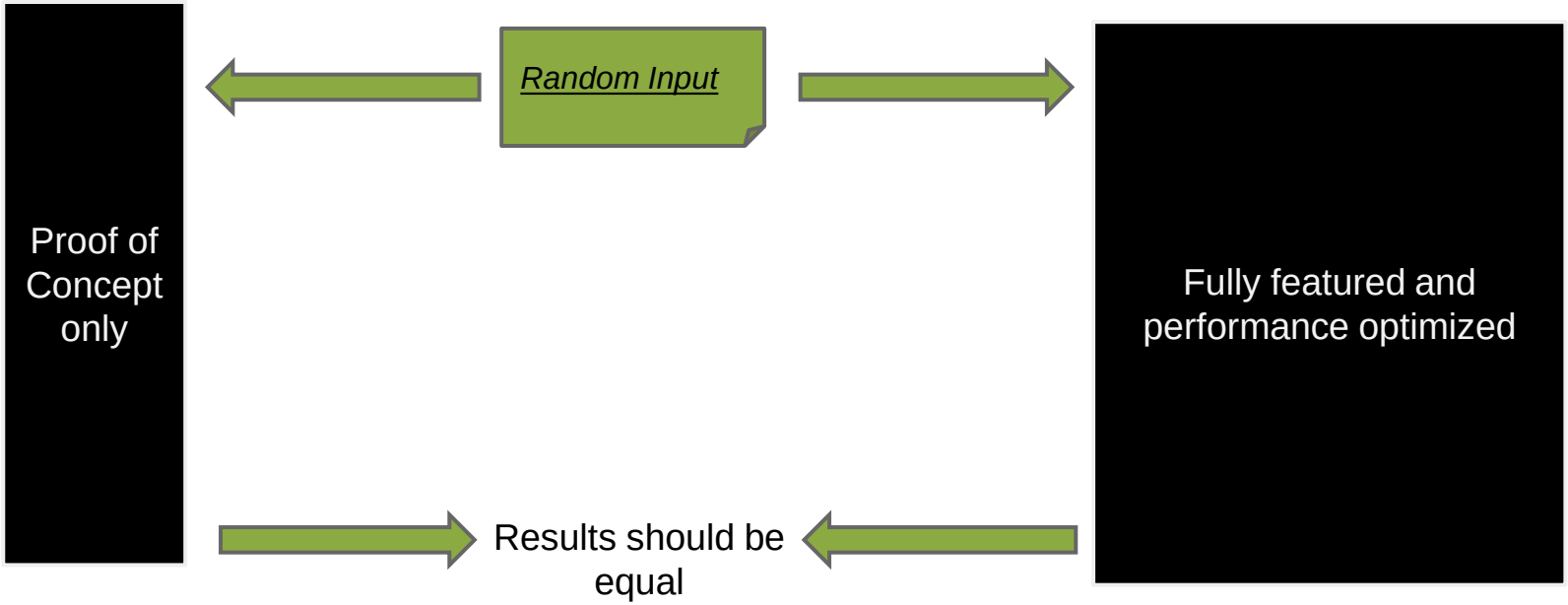
Relation properties

```
val propRankTweet = Prop.forAll(genTweet, genTweet) {
(tweet1,tweet2) =>
val score1 = rankTweet(tweet1)
val score2 = rankTweet(tweet2)
if (tweet1.length <= tweet2.length) score1 >= score2
else score1 < score2
}
```

Reference Implementation

Compare your implementation
to a known working one.

Reference implementation



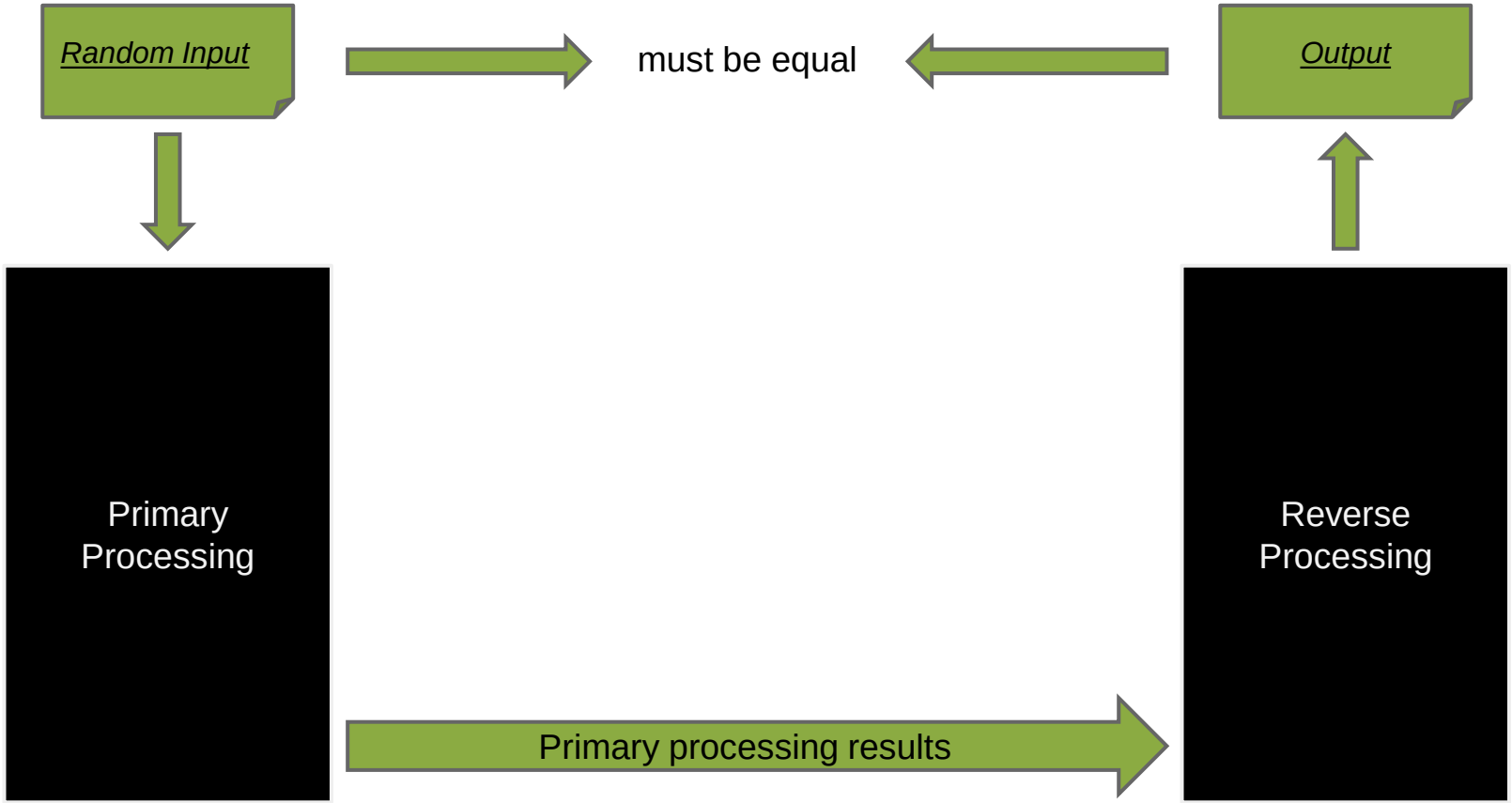
Reference implementation

```
val propCheckImpl = Prop.forAll(genInput) {
  input =>
    runPOC(input) == runImpl(input)
}
```

Round-trip Properties

Compare your input data
to an inverted result of the primary processing.

Round-trip properties



Round-trip properties

```
val propDecodeEncode = Prop.forAll{
  s: String =>
    decodeString(encodeString(s)) == s
}
```

Contract Properties

Make sure that a contract holds.

The contract of the `equals` method

- It is **reflexive**:
  for any non-null value x, the expression `x.equals(x)` *should return* `true`

- It is **symmetric**:
  for any non-null values x and y, `x.equals(y)` *should return* `true` *if and only if* `y.equals(x)` *returns* `true`

- It is **transitive**:
  for any non-null values x,y and z, if `x.equals(y)` *returns* `true` *and* `y.equals(z)` *returns* `true`, *then* `x.equals(z)` *should return* `true`

- It is **consistent**:
  for any non-null values x and y multiple invocations of `x.equals(y)` *should consistently return* `true` *or consistently return* `false`*, provided no information used in equals comparisons on the object is modified*

- *For any non-null value x,* `x.equals(null)` *should return* `false`

Demonstration

Enhance your Unit Testing with Property Based Testing for

- ✓ Sanity checks
- ✓ Checking relations between results
- ✓ Testing against a reference implementation
- ✓ Ensuring round-trip capabilities
- ✓ Testing that a contract holds
- ✓ Behavioural simulations for robustness tests
- ✓ Anything that does not fit well in equivalence partitioning

Further readings

Home

https://www.scalacheck.org/

API

https://scalacheck.googlecode.com/svn/artifacts/1.9/doc/api/org/scalacheck/package.html

User Guide

https://github.com/rickynils/scalacheck/wiki/User-Guide

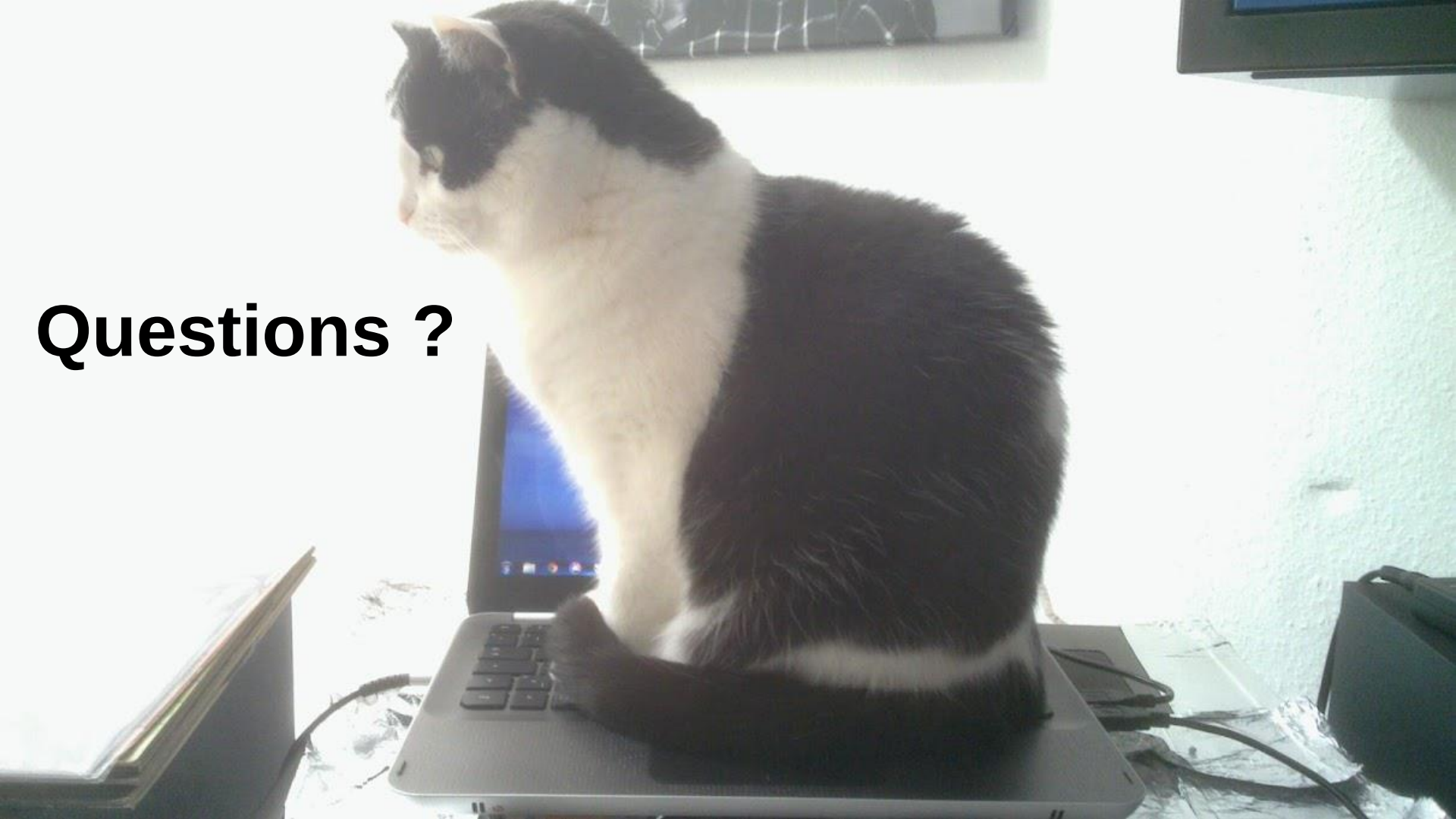
Integration with ScalaTest

http://www.scalatest.org/user_guide/property_based_testing

Integration with specs²

https://etorreborre.github.io/specs2/guide/SPECS2-3.6.4/org.specs2.guide.UseScalaCheck.html

Questions ?



TESTING BEYOND EQU

at 16:30–17:00 by Manfred Rätzman

Main Statement:

> Property based testing is a powe
> and integration testing. With Scal

Sometimes example based testing
does not sufficiently fit your testing
have well known and structured inp
challenges are sanity-checks or rob
implementations, or hunting faults
situations property based testing ca
Property based testing supports th
specifications describe the behavio

RATE THIS SESSION

CONSENSUS TALKS VI

by Lim Sim and Manfred Rätzmann

CONTENT NOT RATED

CLARITY OF SPEECH NOT RATED

NEW INSIGHTS NOT RATED