Property-Based Testing mit Java

JUG Switzerland

Zürich, 4. Dezember 2019

@johanneslink

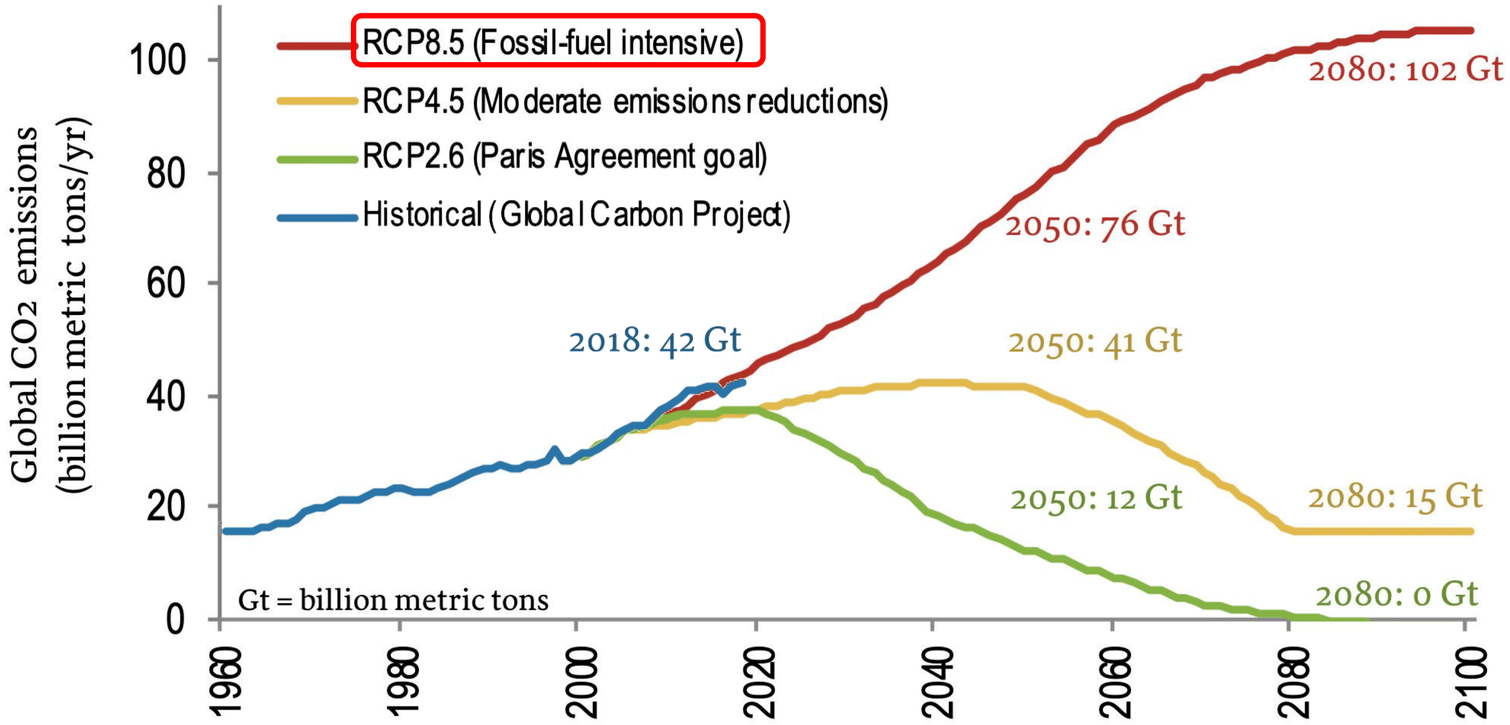
johanneslink.net

Softwareentwicklung ist völlig ohne Bedeutung…

… wenn es morgen **unsere** Welt in dieser Form **nicht** **mehr geben wird**

CLIMATE FACTS NOW

CO_2 Emissionen

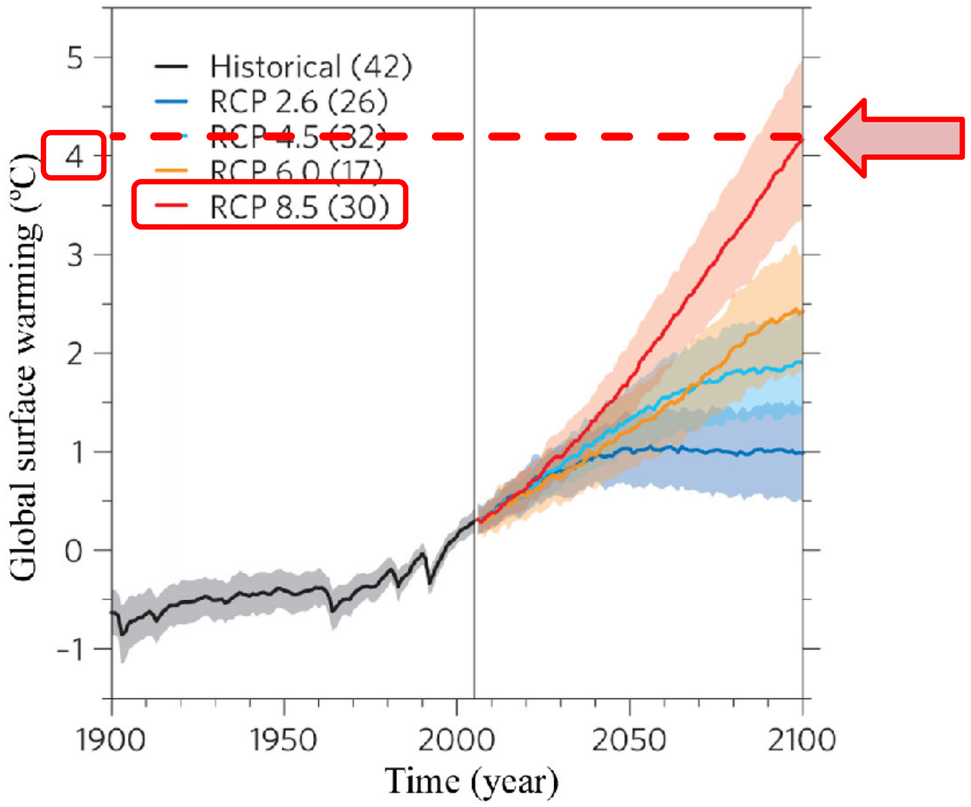


(R. Kopp, Rutgers Climate Institute; aktualisiert von R. Kopp nach Kopp et al. 2014)

Quellen und weitere Details: www.ClimateFactsNow.org

CLIMATE FACTS NOW

Temperatur



Knutti and Sedlácek (2013)

Quellen und weitere Details: www.ClimateFactsNow.org

Wir sind auf dem Weg zum Worst-Case-Szenario!

Weniger als 2 Grad Erderwärmung sind vermutlich schon nicht mehr machbar.
4 Grad noch in diesem Jahrhundert sind wahrscheinlich.

- Massive Hitzewellen und Trockenzeiten
- Intensive Überflutungen
- Zerstörung großer Ökosysteme (Korallenriffe, Feuchtgebiete, Wälder)
- Weltweite Ernteeinbrüche
- Öffentliche Unruhen und politische Instabilisierung
- Kriege um Wasser, Land und Nahrungsmittel
- Bis 2050 ca. 400 Mio Flüchtlinge
- und und und und

28. NOVEMBER 2019 DIE ZEIT Nº 49

GLÜHENDE LANDSCHAF

...ts
...knapp,
...markt.

...HLENBERG,

...n sollen es werden, wofür die Stadt ... Dollar aufbringen muss. Zur Fi... ...rden Schuldverschreibungen aus... Rating Agentur S&P bewertete sie ... dramatischen Prognosen für den ...eresspiegels. Die Stadt, so die Be... Agentur, habe schon sehr viel für ... ihren Schutz gegen den Klima...

...angekommen, zeigt Berthier das ...m Morgen gemacht hat, ihrer Che... ...Für die sind solche Bilder in... ...»Ist das eine städtische oder ... von der Schule?« fragt sie. »Städ... ...wortet Berthier. »Wir sind schon da... *...eine Erhöhung vorzunehmen.«*

...43, die stellvertretende Resilienz... ...on Miami Beach, sieht ein bisschen ...ge Hillary Clinton, und sie strahlt ...die Botschaft aus: »Wir haben das ...hre Abteilung berät den Bürger... ...rhindern, dass Miami Beach in ...hren untergeht. Aus diesem ... sie überzeugt, kann auch ein ...odell werden.

...g tagt Knowles mit Kollegen ...meinden. Hauptthema ist die ...ung. Die Flut schwemmt Öl ...e Straßen. Wie kann man die ...echenschaft ziehen? Lässt sich ... Intelligenz (KI) nutzen? ... Kollegen sehen die Region ...ue Silicon Valley für Resilienz, ... Einsatz von Big Data und KI ...der Erderwärmung. Man müsse ...Küstenstädte klarmachen, »dass ...em Klimawandel viele ökono... ...ungen«, sagt Knowles' Kollegin ...arten Miami.

... die Millionäre nicht nur als ...en. Die Immobiliensteuer ist ...die einzige Einnahme... ...in Florida. Ziehen die ...ben die Städte kein Geld mehr, ...wassers... Genauer ge...

neues Silicon Valley der Klima-Resilienz aufbauen zu wollen, während die eigene Regierung das Pariser Abkommen aufkündigt und in Erdöl und Kohle investieren will, ist also erst mal vernunftwidrig. Trotzdem ist Anpassung nicht gleich Verdrängung. Mit besserer Infrastruktur und Schutzmaßnahmen können sich manche Städte und Staaten wappnen, sich Handlungsspielraum erkaufen und damit auch beeinflussen, ob ihre Gesellschaft im absoluten Notfall zur Kampfzone wird – oder zu einer Solidargemeinschaft.

Kapstadt, Südafrika

Day Zero, Tag null: Aus dem Hahn kommt kein Tropfen Wasser. Vom Duschen kann man nur träumen, Hände bloß noch chemisch desinfizieren, die Toiletten nicht mehr spülen. Alle Einwohner müssen mit leeren Flaschen und Eimern an öffentlichen Stationen für ihr Trinkwasser anstehen. 25 Liter pro Person und Tag. Keinen Tropfen mehr.

Im Januar 2018 stand Kapstadt kurz vor dem Tag null. Eine Jahrhundertdürre, verschärft durch den Klimawandel, hatte die Staubecken der Stadt fast leerlaufen lassen. Parks und Felder ...

afrikanische Musiker, Songs zu komponieren, die nicht länger als zwei Minuten dauern. ...

ZEIT-GRAFIK/Quelle: CONNECTOGRAPHY von Parag Khanna (2016)

shower songs auf Lager. Auch solche, die nur eine Minute dauern.

Der Klimawandel verschärft nicht nur extreme Wetterlagen, er bestraft schlechtes Regieren. Gemeint sind die Versäumnisse fast aller Staaten beim Klimaschutz. Gemeint sind ganz profane Misswirtschaft und Fehlplanung: marode Kanäle, ...

denkilometern heranrast. Binnen kurzer Zeit ist alles mit einer dicken Schicht aus Sand bedeckt.

Solche Wüstenstürme hat es immer gegeben, aber mit der Erderwärmung werden sie häufiger und verheerender. Der Sudan gilt als eine »Ground Zero«-Zone des Klimawandels. Die Temperatur steigt hier wie in weiten Teilen Afrikas schneller als im globalen Durchschnitt – nach aktuellen Prognosen um bis zu drei Grad bis 2050. Sommertage mit über 50 Grad häufen sich in Khartum. Die Regenzeiten sind inzwischen so unberechenbar, dass die Bauern kaum wissen, wann sie säen sollen. Weil sich die Wüste ausdehnt, wirbeln die Habubs mehr Sand auf, zerstören mehr Ackerflächen und Weideland.

Nisreen Al Sayeem, 24, Physikerin, klingt erstaunlich fröhlich, wenn sie die Lage ihrer Heimat schildert. ...

Die Welt, vier Grad wärmer

Diese Karte zeigt, wie die Erde aussehen könnte, wenn die globale Durchschnittstemperatur sich um vier Grad gegenüber dem ...

Landes bei Kämpfen um ... Wasser ums Leben.

Die Physikerin Al Saye... weil im Sudan bislang so g... passung an die Klimakrise u... sei, könnten schon kleine... zählt auf: Neue Verfahren... Wasser, widerstandsfähigere... dezentrale Versorgung mi... Wiederaufforstung von W... brauchen wird Geld von ... staaten. Wir schaffen das ... das nicht Not- oder Entw... Gerechtigkeit. Wer Schad... ihn aufkommen.

Aber wichtiger als Gel... halbwegs funktionierend... wesen. Eine Öffentlichk... fen den möglichen K... Ausmaß der Klimakris... zu verstehen, was p... richten können – un...

Zeeland, Nieder...

Das Meer ist ruhi... *Ruten ausgeworfe... Nicht vom Sonne... riesigen Wand aus S... delta der Schelde du... gigantische Betonpfe... Stahltore aneinander, ... sen werden, um die süd... zu schützen. Das Ooster... nument der Ingenieursk...*

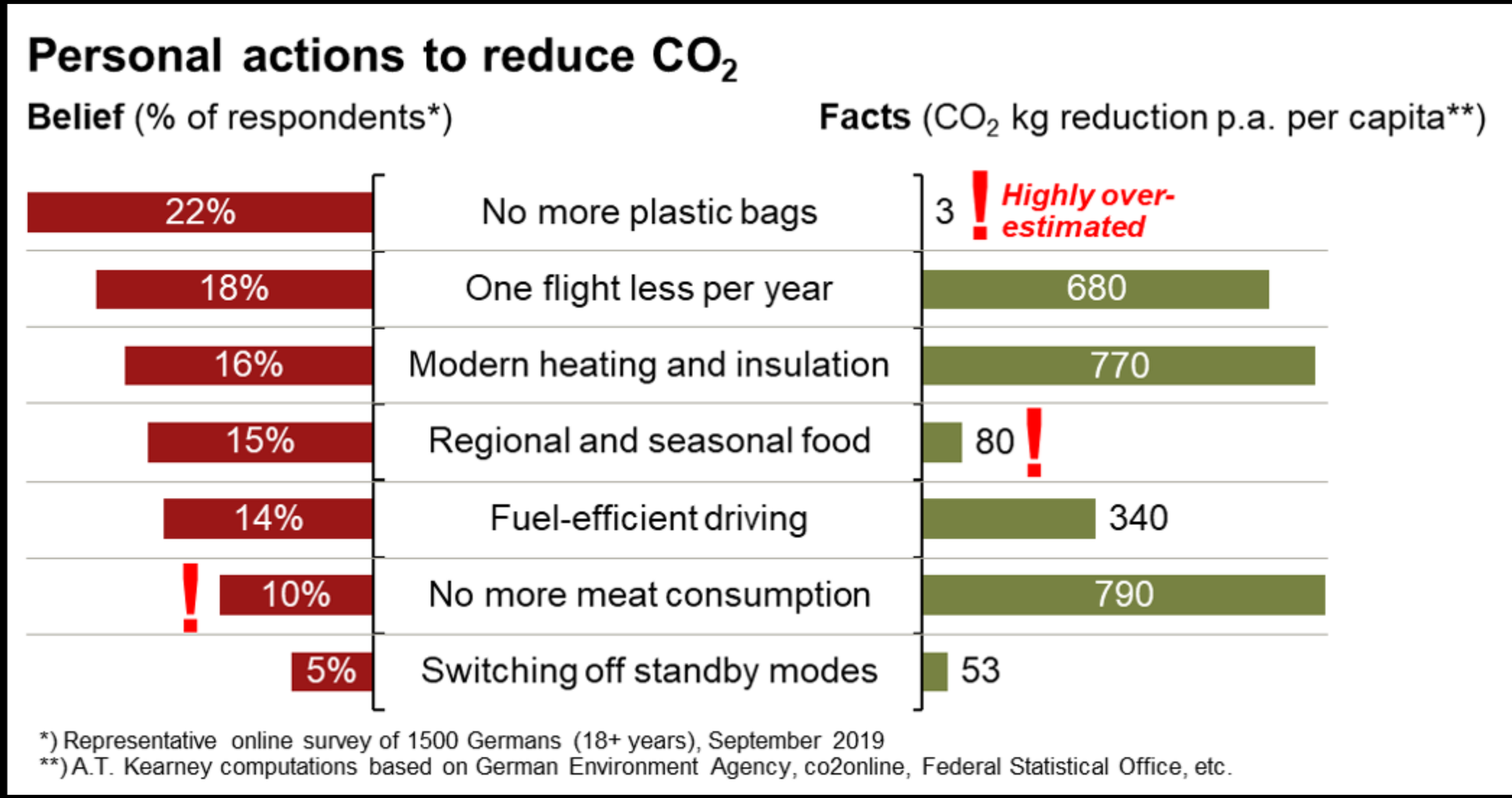
des niederländischen Urgl... herrschen zu können.

Das neue Silicon Valley ... längst. Nicht in Florida, ... Nordsee. Schwimmende ... schinen zur Stabilisierung ... plätze, die sich in Flutbeck... Geht es um den steigenden... kaum eine Idee, die die N... hatten. Einige stammen v... Die 44-jährige Umweltw... für das Technologie-Instit...

Unsere Intuitionen aus der Softwareentwicklung taugen nicht bei der Eingrenzung der Klimakrise!

- Klimakrise Pro/Contra ist nicht Emacs vs VIM
- Klimapanik ist keine willkürliche Deadline eines Projektleiters
- Technologie allein wird das Problem nicht lösen
- Der "gesunde Mittelweg" hilft nicht, wenn ein Extrempunkt Überleben heißt

Was können wir tun?



Was müssen wir tun?

- Die Fakten und wissenschaftlichen Erkenntnisse immer wieder aussprechen
  - Insbesondere bei öffentlichen Auftritten
- Die Politik(er/innen) unter Druck setzen
  - Wählen
  - Briefe, Anrufe, Emails, Petitionen, Volksbegehren
  - Demos, Streiks, Blockaden

ScientistsForFuture

FridaysForFuture

ParentsForFuture

EXtinctionRebellion

The Show Must Go On...

Property-Based Testing mit Java

Beispiel-basierte Tests

Ein *Beispiel* zeigt, dass unser Code
bei ganz konkreten Eingaben
ein ganz konkretes Ergebnis liefert.

```
@Example
void reverseList() {
   List<Integer> aList = Arrays.asList(1, 2, 3);
   Collections.reverse(aList);
   assertThat(aList).containsExactly(3, 2, 1);
}
```

Funktioniert *reverse()* nur für die getesteten Beispiele?

Wie repräsentativ sind unsere Tests?

Wie viele Beispiele benötigen wir um ausreichendes Vertrauen zu schaffen?

```
@Example void emptyList() {
    List<Integer> aList = Collections.emptyList();
    assertThat(Collections.reverse(aList)).isEmpty();
}

@Example void oneElement() {
    List<Integer> aList = Collections.singletonList(1);
    assertThat(Collections.reverse(aList)).containsExactly(1);
}

@Example void manyElements() {
    List<Integer> aList = asList(1, 2, 3, 4, 5, 6);
    assertThat(Collections.reverse(aList)).containsExactly(6, 5, 4, 3, 2, 1);
}

@Example void duplicateElements() {
    List<Integer> aList = asList(1, 2, 2, 4, 6, 6);
    assertThat(Collections.reverse(aList)).containsExactly(6, 6, 4, 2, 2, 1);
}
```

Properties

Eine *Property* zeigt, dass unser Code
für eine Klasse von Eingaben (Vorbedingung)
bestimmte allgemeine Eigenschaften (Invarianten)
erfüllt.

```
Collections.reverse(List aList):
    // Vorbedingungen?
    // Nachbedingungen und Invarianten?
```

```
Collections.reverse(List aList):
   // Vorbedingungen?
   // Nachbedingungen und Invarianten?
```

Vorbedinungen

- Beliebige Liste - aber nicht null

Invarianten

- Länge der Liste bleibt gleich
- Alle Elemente bleiben erhalten
- Nach reverse ist das erste Elemente das letzte
- 2 x reverse erzeugt wieder das Original

Eine Property als Java Code

```
boolean theSizeRemainsTheSame(List<Integer> original) {
   List<Integer> reversed = reverse(original);
   return original.size() == reversed.size();
}


private <T> List<T> reverse(List<T> original) {
   List<T> clone = new ArrayList<>(original);
   Collections.reverse(clone);
   return clone;
}
```

Jqwik

```
@Property
boolean theSizeRemainsTheSame(@ForAll List<Integer> original) {
   List<Integer> reversed = reverse(original);
   return original.size() == reversed.size();
}
```

```java
@Property
void allElementsStay(@ForAll List<Integer> original) {
    List<Integer> reversed = reverse(original);
    Assertions.assertThat(original).allMatch(
        element -> reversed.contains(element)
    );
}
```

```java
@Property
boolean reverseMakesFirstElementLast(@ForAll List<Integer> original) {
    Assume.that(original.size() > 0);
    Integer lastReversed = reverse(original).get(original.size() - 1);
    return original.get(0).equals(lastReversed);
}
```

```java
@Property
boolean reverseTwiceIsOriginal(@ForAll List<Integer> original) {
    return reverse(reverse(original)).equals(original);
}
```

```haskell
prop_reversed :: [Int] -> Bool
prop_reversed xs =
    reverse (reverse xs) == xs
```

Haskell!

Demo

- Reverse List
- Length of String
- Absolute value of Integer
- Sum of two integers
- Einbindung in Gradle und IntelliJ

Was ist jqwik?

https://jqwik.net

- Eine Test-Engine für die JUnit5–Plattform
- Ein Generator für Testfälle mit
  - zufälligen und typischen Eingabewerten
  - manchmal sogar die vollständige Menge aller möglichen Eingabekombinationen
- Aktuelle Version: 1.2.1

Was ist jqwik nicht?

- Es ist kein vollständig randomisiertes Testwerkzeug, das man ohne Nachdenken auf sein Programm loslässt.
- Properties werden nicht bewiesen, sondern widerlegt (aka falsifiziert)

```java
@Property
void squareOfRootIsOriginalValue(@ForAll double aNumber) {
    double sqrt = Math.sqrt(aNumber);
    Assertions.assertThat(sqrt * sqrt).isCloseTo(aNumber, withPercentage(1));
}
```

```
java.lang.AssertionError:
  Expecting:
    <NaN>
  to be close to:
    <-1.0>
  by less than 1% but difference was NaN%.
  (a difference of exactly 1% being considered valid)
```

Beschränkung generierter Werte

Häufig gilt eine Property nur für eine
beschränkte Untermenge eines Typs

```java
@Property
void squareOfRootIsOriginalValue(
    @ForAll @DoubleRange(min=0) double aNumber
) {
    double sqrt = Math.sqrt(aNumber);
    Assertions.assertThat(sqrt * sqrt).isCloseTo(aNumber, withPercentage(1));
}
```

```
tries = 1000,
checks = 1000,
seed = 7890962728489990406
```

```java
@Property
void squareOfRootIsOriginalValue(
    @ForAll @Positive double aNumber
) {
    double sqrt = Math.sqrt(aNumber);
    Assertions.assertThat(sqrt * sqrt).isCloseTo(aNumber, withPercentage(1));
}
```

```
tries = 1000,
checks = 1000,
seed = 7890962728489990406
```

```java
@Property
void squareOfRootIsOriginalValue(
    @ForAll("positiveDoubles") double aNumber
) {
    double sqrt = Math.sqrt(aNumber);
    Assertions.assertThat(sqrt * sqrt).isCloseTo(aNumber, withPercentage(1));
}

@Provide
Arbitrary<Double> positiveDoubles() {
    return Arbitraries.doubles().between(0, Double.MAX_VALUE);
}
```

```
tries = 1000,
checks = 1000,
seed = 7890962728489990406
```

Arbitrary:
"Monaden-ähnliche" Factory von Generatoren
für zufällige Werte

```
public interface Arbitrary<T> {
    RandomGenerator<T> generator(int genSize);

    default Arbitrary<T> filter(final Predicate<T> filterPredicate) {…}
    default <U> Arbitrary<U> map(final Function<T, U> mapper) {…}
    …
}
```

```
public interface RandomGenerator<T> {
    Shrinkable<T> next(Random random);
}
```

```java
@Property
void squareOfRootIsOriginalValue(@ForAll double aNumber) {
    Assume.that(aNumber > 0);

    double sqrt = Math.sqrt(aNumber);
    Assertions.assertThat(sqrt * sqrt).isCloseTo(aNumber, withPercentage(1));
}
```

```
tries = 1000,
checks = 489,
seed = -1808546598028468149
```

```
static <E> List<E> brokenReverse(List<E> aList) {
    if (aList.size() < 4) {
        aList = new ArrayList<>(aList);
        reverse(aList);
    }
    return aList;
}

@Property(shrinking = ShrinkingMode.OFF)
boolean reverseShouldSwapFirstAndLast(@ForAll List<Integer> aList) {
    Assume.that(!aList.isEmpty());
    List<Integer> reversed = brokenReverse(aList);
    return aList.get(0) == reversed.get(aList.size() - 1);
}
```

```
org.opentest4j.AssertionFailedError:
  Property [reverseShouldSwapFirstAndLast] falsified with sample
     [[0, -2147483648, 2147483647, -7997, 7997, -3223, -6474, 1915, -7151,
     3102, 4362, 714, 3053, 1919, -445, 7498, -2424, 3016, -5127, -7401,
     -7946, -3801, -305]]
```

```java
static <E> List<E> brokenReverse(List<E> aList) {
    if (aList.size() < 4) {
        aList = new ArrayList<>(aList);
        reverse(aList);
    }
    return aList;
}

@Property(shrinking = ShrinkingMode.OFF)
boolean reverseShouldSwapFirstAndLast(@ForAll List<Integer> aList) {
    Assume.that(!aList.isEmpty());
    List<Integer> reversed = brokenReverse(aList);
    return aList.get(0) == reversed.get(aList.size() - 1);
}
```

```
org.opentest4j.AssertionFailedError:
  Property [reverseShouldSwapFirstAndLast] falsified with sample
     [[0, 0, 0, -1]]
```

The Importance of Being Shrunk

- "Schrumpfen" einer falsifizierten Property: Finde das einfachste Eingabe-Beispiel, das immer noch fehlschlägt.
- Manchmal gibt es das einfachste Beispiel nicht, oder die Suche danach würde sehr lange dauern.
- Benutze Heuristiken um Werte zu schrumpfen, z.B.
  - Versuche Zahlen-Werte näher bei Null
  - Verkleinere Listen, Mengen, Arrays

Type-based vs Integrated Shrinking

- Type-Based Shrinking: Nur der Typ von Werten dient als Constraint für die Schrumpfversuche
  - Problem: Schrumpfen kann zu Ergebnissen führen, die eigentlich bei der Generierung ausgeschlossen wurden
- Integrated Shrinking: Alle Schritte und Bedingungen der Generierung werden beim Schrumpfen berücksichtig
- jqwik implementiert integriertes Schrumpfen

```java
@Property
boolean shrinkingCanBeComplicated(
    @ForAll("first") String first,
    @ForAll("second") String second
) {
    String aString = first + second;
    return aString.length() > 5 || aString.length() < 4;
}

@Provide
Arbitrary<String> first() {
    return Arbitraries.strings() //
            .withCharRange('a', 'z') //
            .ofMinLength(1).ofMaxLength(10) //
            .filter(string -> string.endsWith("h"));
}

@Provide
Arbitrary<String> second() {
    return Arbitraries.strings() //
            .withCharRange('0', '9') //
            .ofMinLength(0).ofMaxLength(10) //
            .filter(string -> string.length() >= 1);
}
```

Werte generieren

Arbitraries sind der Anfang von allem…

```
Arbitraries
  .strings()
  .integers()
  .floats()
  …
  .of(…) // values, enums
  .frequency(…) // add weights
  .constant(…)
  .oneOf(…)
  …
```

Generatoren Konfigurieren

Fluent Interfaces

Arbitraries können konfiguriert werden

```
@Provide
StringArbitrary fluentString() {
   return Arbitraries.strings()
                     .alpha()
                     .numeric()
                     .withChars('?', '!', '.')
                     .ofMinLength(2)
                     .ofMaxLength(10);
}
```

Generierte Werte verändern

- Manchmal möchte man generierte Werte filtern
- Manchmal möchte man generierte Werte abbilden
- Manchmal möchte man generierte Werte miteinander kombinieren

Werte filtern

```
@Property
boolean evenNumbersAreEven(@ForAll("evenUpTo10000") int anInt) {
    return anInt % 2 == 0;
}

@Provide
Arbitrary<Integer> evenUpTo10000() {
    return Arbitraries.integers()
                .between(0, 10000)
                .filter(i -> i % 2 == 0);
}
```

Werte abbilden

```
@Provide
Arbitrary<Integer> evenUpTo10000() {
    return Arbitraries.integers()
                .between(0, 5000)
                .map(i -> i * 2);
}
```

```
@Provide
Arbitrary<Integer> evenUpTo10000() {
    return Arbitraries.integers()
                .between(0, 10000)
                .filter(i -> i % 2 == 0);
}
```

Werte kombinieren

```
public class Person {
    public Person(String firstName, String lastName) {…}
    public String fullName() {return firstName + " " + lastName;}
}
```

```
@Provide
Arbitrary<Person> validPerson() {
    Arbitrary<Character> initialChar = Arbitraries.chars().between('A', 'Z');
    Arbitrary<String> firstName = Arbitraries.strings()… ;
    Arbitrary<String> lastName = Arbitraries.strings()… ;
    return Combinators.combine(initialChar, firstName, lastName)
            .as((initial, first, last) -> new Person(initial + first, last));
}
```

Exhaustive Value Generation

```
@Property(generation = GenerationMode.EXHAUSTIVE)
void allChessSquares(
      @ForAll @CharRange(from = 'a', to = 'h') char column,
      @ForAll @CharRange(from = '1', to = '8')char row
) {
   String square = column + "" + row;
   System.out.println(square);
}
```

Demo

- `ExhaustiveGenerationExamples`

How to Specify it

Patterns and Strategies of PBT

- Fuzzying
- Postconditions
- Metamorphic Properties
- Black-box Testing
- Inductive Testing
- Test Oracle
- Stateful Testing
- Model-based Properties

Fuzzing:
Your Code Should not Explode

- Generiere alle denkbaren Arten von Inputs und teste, dass der Basis-Kontrakt einer Funktion immer erfüllt wird, z.B.:
  - keine Exceptions,
  - keine Nulls als Rückgabe
  - Rückgabe im erlaubten Wertebereich
  - Laufzeit unter einer bestimmten Grenze
- Besonders wertvoll bei Integrierten Tests

Postconditions

- Häufig können wir Nachbedingung für die Ausführung von Operationen benennen
- Beispiel: Einfügen in Binary Search Tree
  - *"Nach dem Einfügen von Key X mit Value Y in irgendeinen beliebigen Binärbaum, sollten wir bei Suche nach X den Wert Y finden"*

```
@Property
boolean inserted_value_can_be_found(
        @ForAll Integer key, @ForAll Integer value,
        @ForAll("trees") BST<Integer, Integer> bst
) {
    Optional<Integer> found = bst.insert(key, value).find(key);
    return found.equals(Optional.of(value));
}
```

Invarianten sind auch Nachbedingungen…

Manche Dinge ändern sich nie…

- Die Größe einer Liste nach dem Mapping
- Die Inhalte einer Liste nach dem Sortieren
- Die gesamte Geldmenge nach einer Überweisung

Metamorphic Properties

"... even if the expected result of a function call [...] may be difficult to predict, we may still be able to express an expected relationship between this result, and the result of a related call."

John Hughes in *How to Specify It*

https://johanneslink.net/how-to-specify-it

Metamorphic Property: Inject Data

- Wenn ich f(x) kenne, dann kann ich f(x + a) daraus ableiten
  - reverse(L) = L' & reverse(M) = M'
    => reverse(L + M) = M' + L'

```
@Property
boolean reverseConcatenatedLists(
        @ForAll List<Integer> first,
        @ForAll List<Integer> second
) {
    List<Integer> firstReversed = reverse(first);
    List<Integer> secondReversed = reverse(second);

    List<Integer> reversed = reverse(concat(first, second));

    return reversed.equals(concat(secondReversed, firstReversed));
}
```

Metamorphic Property: Inverse Functions

- Funktion + inverse Funktion ergibt die ursprüngliche Eingabe
  - Encode / Decode

```java
class InverseFunctions {

    @Property
    void encodeAndDecodeAreInverse(
            @ForAll @StringLength(min = 1, max = 20) String toEncode,
            @ForAll("charset") String charset
    ) throws UnsupportedEncodingException {
        String encoded = URLEncoder.encode(toEncode, charset);
        assertThat(URLDecoder.decode(encoded, charset)).isEqualTo(toEncode);
    }

    @Provide
    Arbitrary<String> charset() {
        Set<String> charsets = Charset.availableCharsets().keySet();
        return Arbitraries.of(charsets.toArray(new String[charsets.size()]));
    }
}
```

```
sample = ["€", "Big5"]
original-sample = ["鯒阧퇼ꞵ̧찝蚨ݣ҈넫ᅥꝸ۴虌", "x-IBM1098"]

  org.opentest4j.AssertionFailedError:
  Expecting:
   <"?">
  to be equal to:
   <"€">
  but was not.
```

Häufige "Metamorphic Properties"

- Idempotenz
  - Mehrfache Anwendung einer Operation verändert nichts
- Kommutativität
  - Veränderung der Reihenfolge von Operationen ist egal
- Symmetrie
  - Veränderung von Ausgangsdaten verändert das Ergebnis immer in gleicher Weise
- Robustheit
  - Einfügen von "Rauschen" oder Fehler in Datenströme lässt die Verarbeitungsergebnisse unverändert

Black-box Testing

Hard to compute, easy to verify

- Primzahlermittlung
- Pfad durch ein Labyrinth

Inductive Testing: Solving a smaller problem first

- Manchmal können wir die fachliche Spezifikation durch eine Menge von Properties vollständig beschreiben
- Beispiel: Eine Liste ist sortiert, wenn
  - Das erste Element kleiner als das zweite ist
  - Alles nach dem ersten Element auch sortiert ist

```java
@Property
boolean sortingAListWorks(@ForAll List<Integer> unsorted) {
   return isSorted(sort(unsorted));
}

private boolean isSorted(List<Integer> sorted) {
   if (sorted.size() <= 1) return true;
   return sorted.get(0) <= sorted.get(1)
           && isSorted(sorted.subList(1, sorted.size()));
}
```

Test Oracle: Verifizieren ein Ergebnis gegen alternative Implementierung

- Einfach, aber nicht-performant
- Parallel versus Single-Threaded
- Selbst-gemacht versus kommerziell
- Alt (vor dem Refactoring) versus Neu

Stateful Testing

Bei einem zustandsbehafteten Objekt...

- Welche Aktionen sind möglich?
- Welche Invarianten gelten immer?
- Wie wird der Zustand verändert?

Lass den Computer viele zufällig gewählte Aktionen ausprobieren...

```
public class MyStringStack {
    public void push(String element) {…}
    public String pop() {…}
    public void clear() {…}
    public boolean isEmpty() {…}
    public int size() {…}
    public String top() {…}
}
```

```java
public interface Action<M> {
    default boolean precondition(M model) {return true;}
    M run(M model);
}
```

```java
class PopAction implements Action<MyStringStack> {
    @Override
    public boolean precondition(MyStringStack model) {
        return !model.isEmpty();
    }
    @Override
    public MyStringStack run(MyStringStack model) {
        int sizeBefore = model.size();
        String topBefore = model.top();

        String popped = model.pop();
        Assertions.assertThat(popped).isEqualTo(topBefore);
        Assertions.assertThat(model.size()).isEqualTo(sizeBefore - 1);
        return model;
    }
}
```

```
static Arbitrary<Action<MyStringStack>> actions() {
    return Arbitraries.oneOf(push(), clear(), pop());
}
private static Arbitrary<Action<MyStringStack>> push() {
    return Arbitraries.strings().alpha().ofLength(5).map(PushAction::new);
}
private static Arbitrary<Action<MyStringStack>> clear() {…}
private static Arbitrary<Action<MyStringStack>> pop() {…}
```

```
class MyStackProperties {

    @Property
    void checkMyStackMachine(@ForAll ActionSequence<MyStringStack> sequence) {
        sequence.run(new MyStringStack());
    }

    @Provide
    Arbitrary<ActionSequence<MyStringStack>> sequences() {
        return Arbitraries.sequences(MyStringStackActions.actions());
    }
}
```

Demo

- MyStackProperties

Model-based Properties

- Wir benutzen ein vereinfachtes Modell der zu testenden Logik, und vergleichen das Verhalten des Modells mit dem Verhalten unserer Software
  - z.B. Benutze Key-Value-Map als Modell für Datenpersistenz
- Stateful und model-based werden häufig kombiniert

Lessons Learned

- Beispiel-basierte Tests…
  - sind oft bessere Treiber für das funktionale Verhalten
  - helfen beim Verstehen der Fachlichkeit
- Interaktion mit der externen Welt machen Properties langsam
- Randomisierte Tests werden häufiger nicht-deterministisch
- Investiere in domänen-spezifische Generatoren!

Alternative PBT-Tools für Java

- JUnit-Quickcheck:
  Enge Integration mit JUnit 4
- QuickTheories: Arbeitet mit beliebigen Test-Libraries zusammen
- Vavr: Die funktionale Java-Bibliothek hat auch ein eigenes PBT-Modul

jqwik auf Github:

http://github.com/jlink/jqwik

Mitstreiter gesucht!

Code:

http://github.com/jlink/property-based-testing

Slides:

http://johanneslink.net/downloads/PropertyTesting-JUGCH.pdf

Blog Series:

http://blog.johanneslink.net/2018/03/24/property-based-testing-in-java-introduction/