(Automated) Software Testing (Automation)

Maurício Aniche

M.FinavaroAniche@tudelft.nl

Roman Numerals

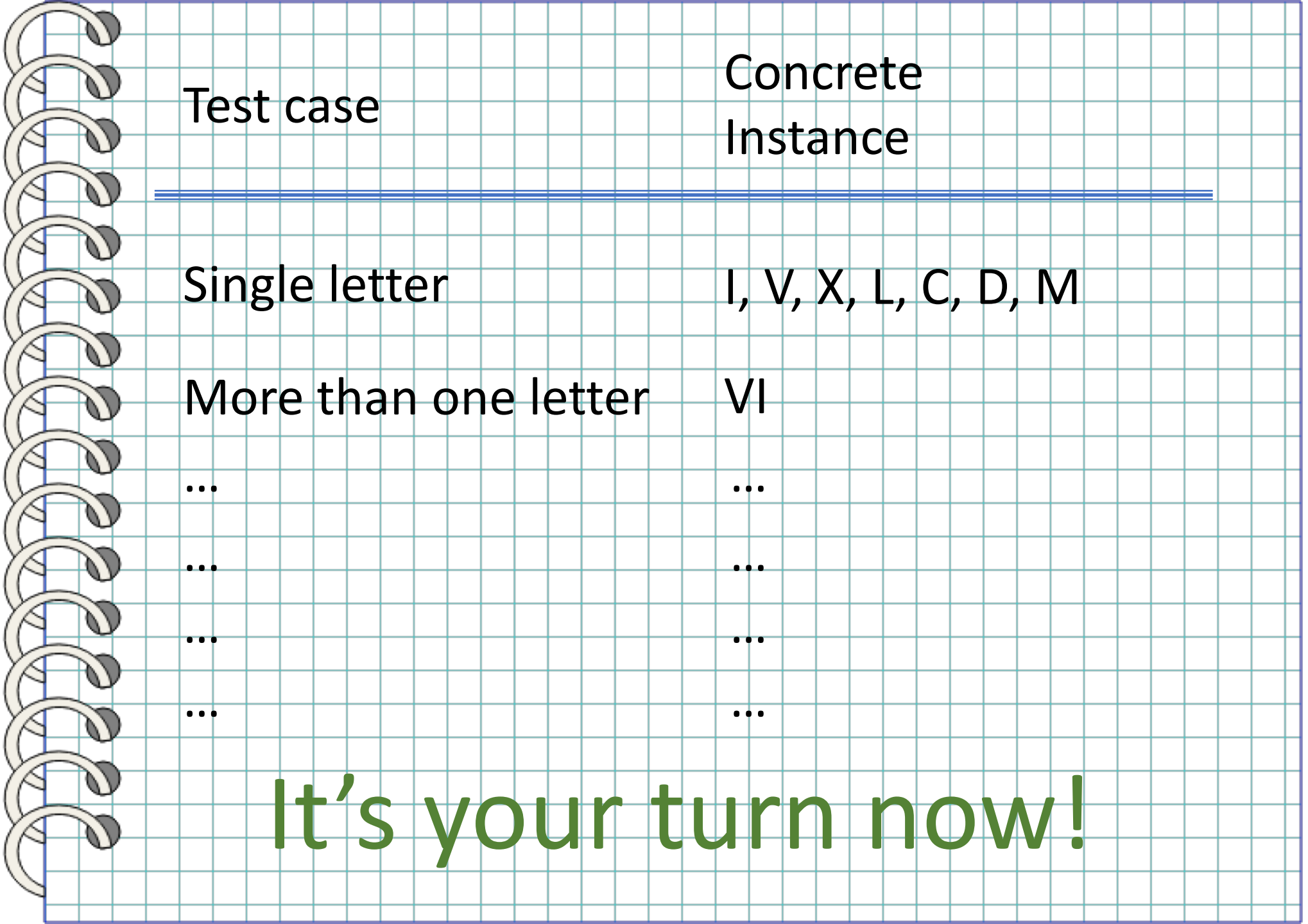
- Given a string in a roman numeral format, the program needs to convert it to an integer.
- I=1, V=5, X=10, L=50, C=100, D=500, M=1000.
- Combine numerals to make numbers: II=2, VII=7, XVI=16.
- Subtractive notation: I, II, III, IV=4, V, VI, VII, VIII, IX=9, X, …

photo by Bence Boros

```java
public class RomanNumeral {

  private static Map<Character, Integer> map;

  static {
    map = new HashMap<Character, Integer>() {{
      put('I', 1);
      put('V', 5);
      put('X', 10);
      put('L', 50);
      put('C', 100);
      put('D', 500);
      put('M', 1000);
    }};
  }
```

```java
public int romanToInt(String s) {
    int convertedNumber = 0;
    for(int i = 0; i < s.length(); i++) {
      int currentNumber = map.get(s.charAt(i));
      int next = i+1 < s.length() ?
          map.get(s.charAt(i+1)) : 0;

      if(currentNumber > next)
        convertedNumber += currentNumber;
      else
        convertedNumber -= currentNumber;
    }

    return convertedNumber;

  }
}
```

~~Source code in: http://bit.ly/sqt-roman-1~~



Test case	Concrete Instance
Single letter	I, V, X, L, C, D, M
More than one letter	VI
…	…
…	…
…	…
…	…

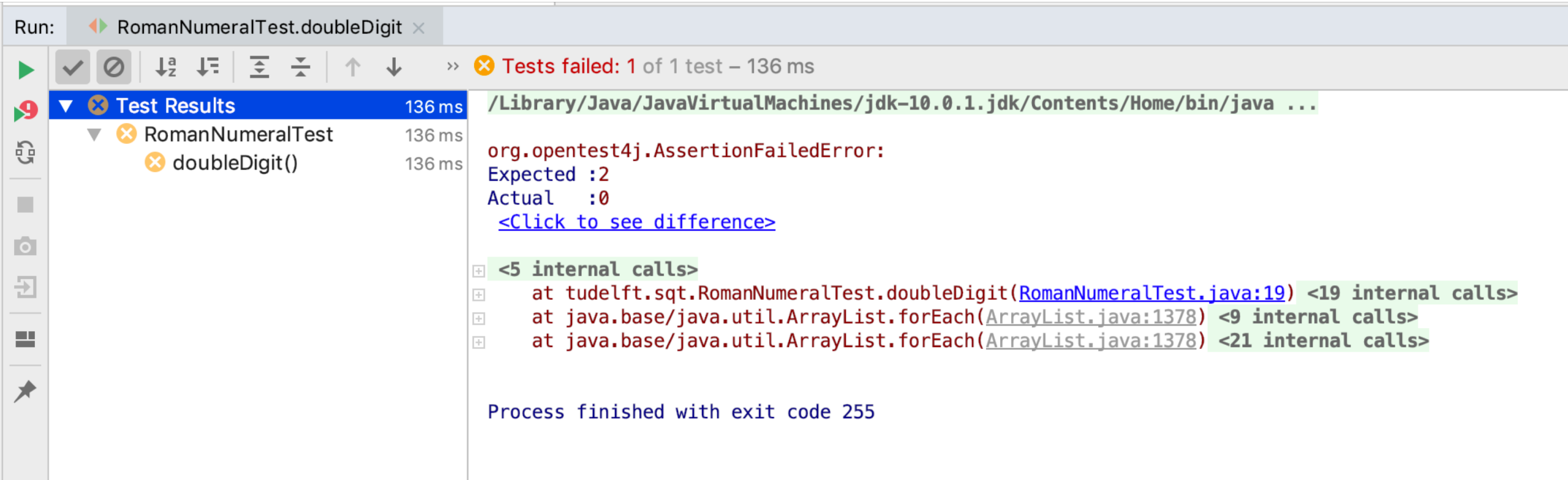
It’s your turn now!

Test case	Concrete Instance
Single letter	I, V, X, L, C, D, M
Many letters in order	VI, XV
Invalid letter	Y
Valid and invalid letter	VIIY
Not valid	IIII, VV
NULL	<null>
…	…

How did you do it? Did you follow any procedure?

Go to **http://bit.ly/sqt-roman-exercise**

```java
@Test
public void bug() {
    int result = new RomanNumeral().romanToInt("II");
    Assertions.assertEquals(2, result);
}
```



```
public int romanToInt(String s) {

    int convertedNumber = 0;
    for(int i = 0; i < s.length(); i++) {
        int currentNumber = map.get(s.charAt(i));
        int next = i+1 < s.length() ?
            map.get(s.charAt(i+1)) : 0;

        if(currentNumber > next)
            convertedNumber += currentNumber;
        else
            convertedNumber -= currentNumber;
    }

    return convertedNumber;

}
```

```java
public int romanToInt(String s) {

    int convertedNumber = 0;
    for(int i = 0; i < s.length(); i++) {
        int currentNumber = map.get(s.charAt(i));
        int next = i+1 < s.length() ?
            map.get(s.charAt(i+1)) : 0;

        if(currentNumber >= next)
            convertedNumber += currentNumber;
        else
            convertedNumber -= currentNumber;
    }

    return convertedNumber;

}
```

Curiosity

"The absence of zero and irrational numbers, impractical and inaccurate fractions, and difficulties with multiplication and division ***prevented the Romans*** *and the Europeans who later used the system* ***from making advances*** *in number theory and geometry as the Greeks had done in the Pythagorean and Euclidean schools."*

https://www.encyclopedia.com/science/encyclopedias-almanacs-transcripts-and-maps/roman-numerals-their-origins-impact-and-limitations

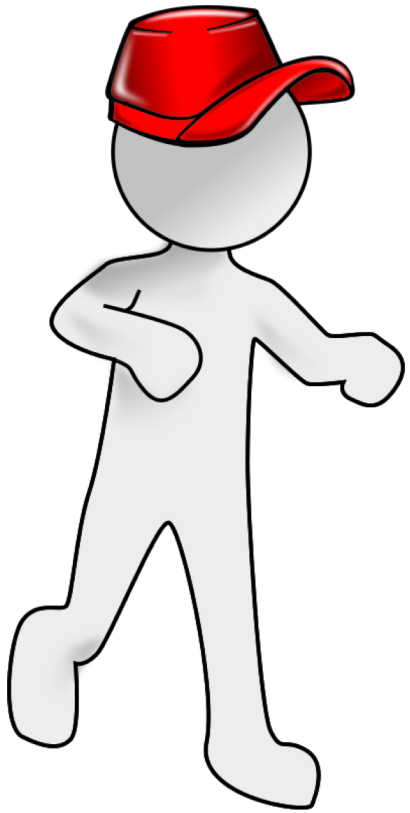
A little story

- First job as a developer in 2004
- First important project in 2006
- First important bug: 2006

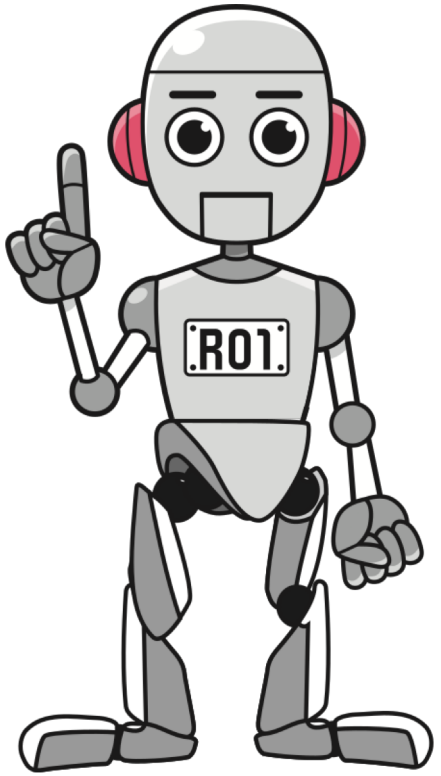
- Tests are important!

Photo by Michael Mims
https://unsplash.com/photos/0ZL0O-eDOpU

TEST ANALYSIS & TEST DESIGN

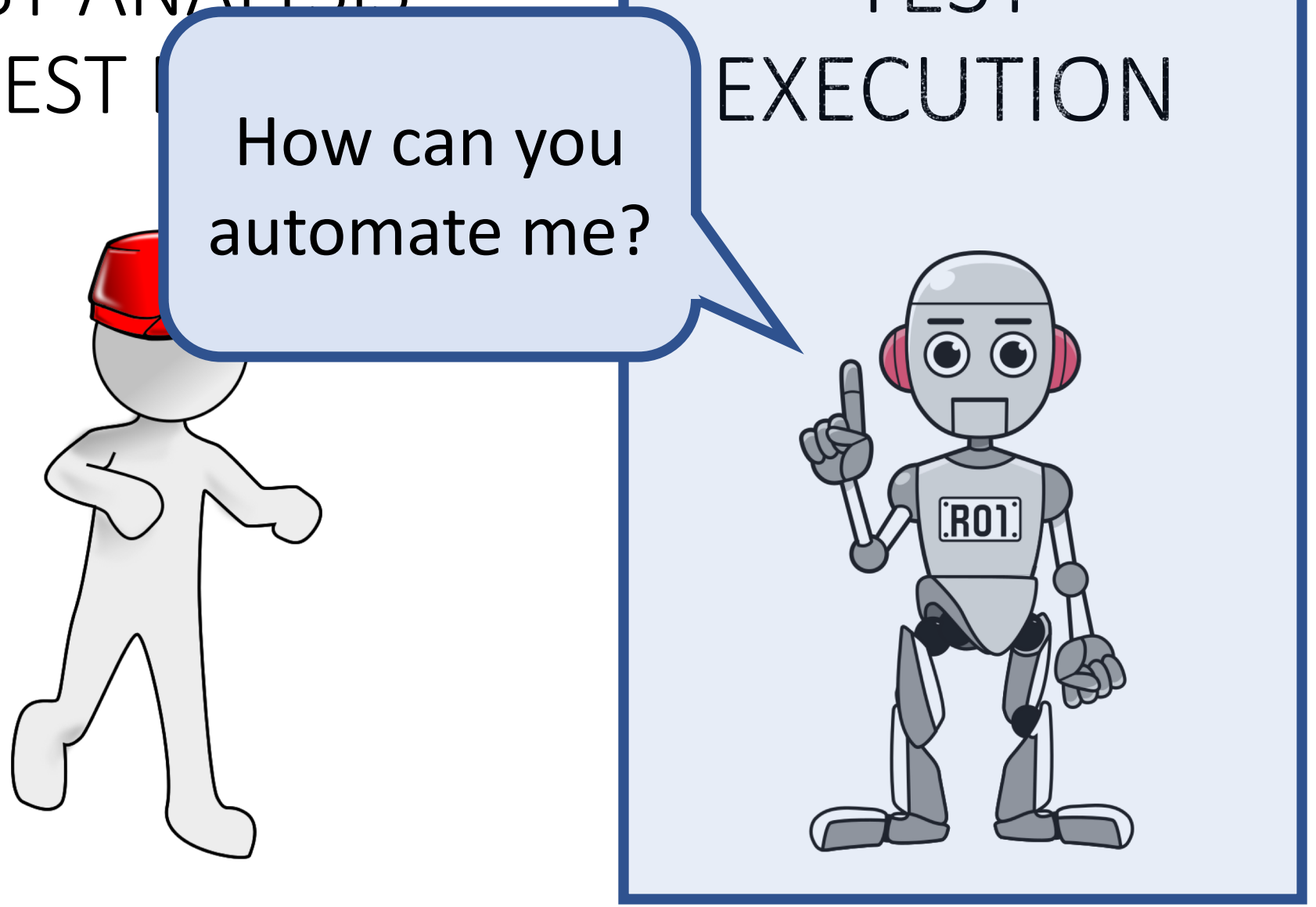


TEST EXECUTION



TEST ANALYSIS
& TEST

TEST
EXECUTION

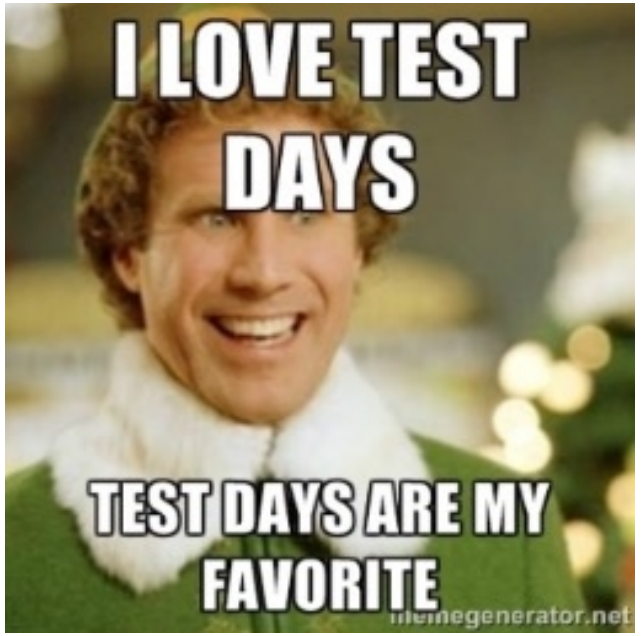


```java
@Test
void singleDigit() {
  Assertions.assertEquals(1, new RomanNumeral().romanToInt("I"));
  Assertions.assertEquals(5, new RomanNumeral().romanToInt("V"));
  Assertions.assertEquals(10, new RomanNumeral().romanToInt("X"));
  Assertions.assertEquals(50, new RomanNumeral().romanToInt("L"));
  Assertions.assertEquals(100, new RomanNumeral().romanToInt("C"));
  Assertions.assertEquals(500, new RomanNumeral().romanToInt("D"));
  Assertions.assertEquals(1000, new RomanNumeral().romanToInt("M"));
}

@Test
void repetition() {
  Assertions.assertEquals(2, new RomanNumeral().romanToInt("II"));
  Assertions.assertEquals(20, new RomanNumeral().romanToInt("XX"));
}

@Test
void manyLettersInOrder() {
  Assertions.assertEquals(1000, new RomanNumeral().romanToInt("VI"));
  Assertions.assertEquals(1000, new RomanNumeral().romanToInt("XV"));
}

...
```

All tests in
http://bit.ly/sqt-roman-2

What are the advantages?

- Too slow → Too Fast
- Too expensive → Machine is cheap
- Not easy to reproduce → Reproducible
- Susceptible to failures → No failures
- … boring! → Very very cool!



- But there's a learning curve (as with any technique).

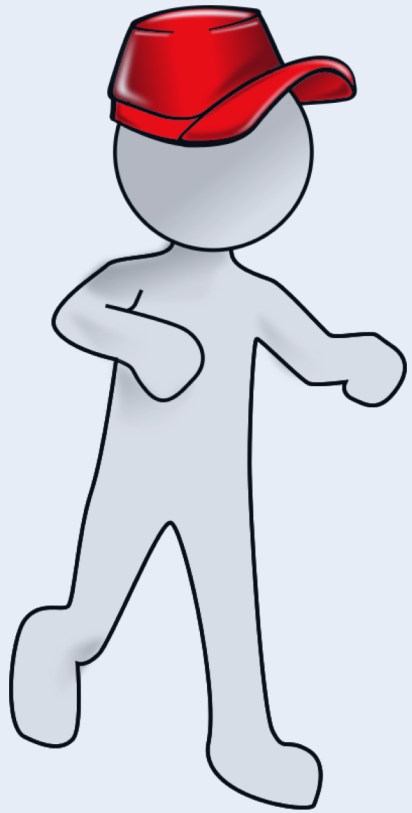
”But if you write 100 lines of production code, now you’ll write only 50, as the other 50 are testing. Therefore, you are less productive.”
– says a bad manager.

Not true.

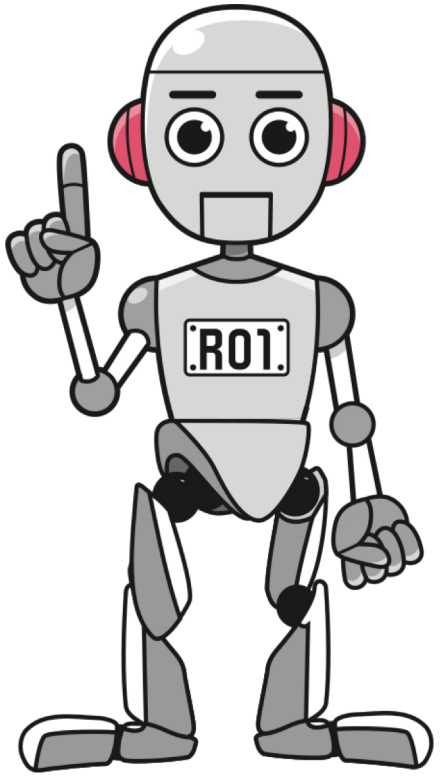
- You spend a lot of time in executing manual tests.
  - Now, you will spend it only once: to write the test.
- Teams with automated test suites spend less time debugging.

George, B., Williams, L., An Initial Investigation of TDD in Industry. ACM Symposium on Applied Computing. Melbourne, Florida, USA, 2003.
Janzen, D., Software Architecture Improvement through Test-Driven Development. Conference on Object Oriented Programming Systems Languages and Applications, ACM, 2005

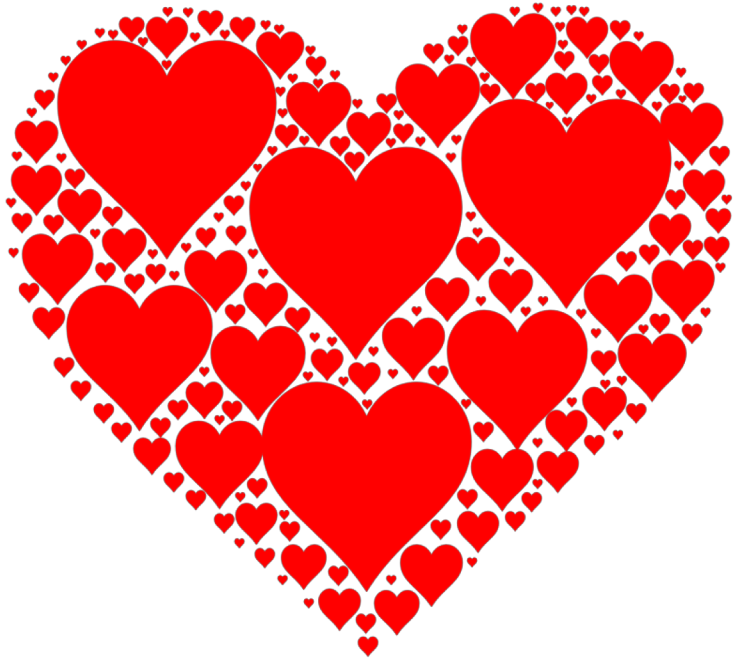
TEST ANALYSIS & TEST DESIGN



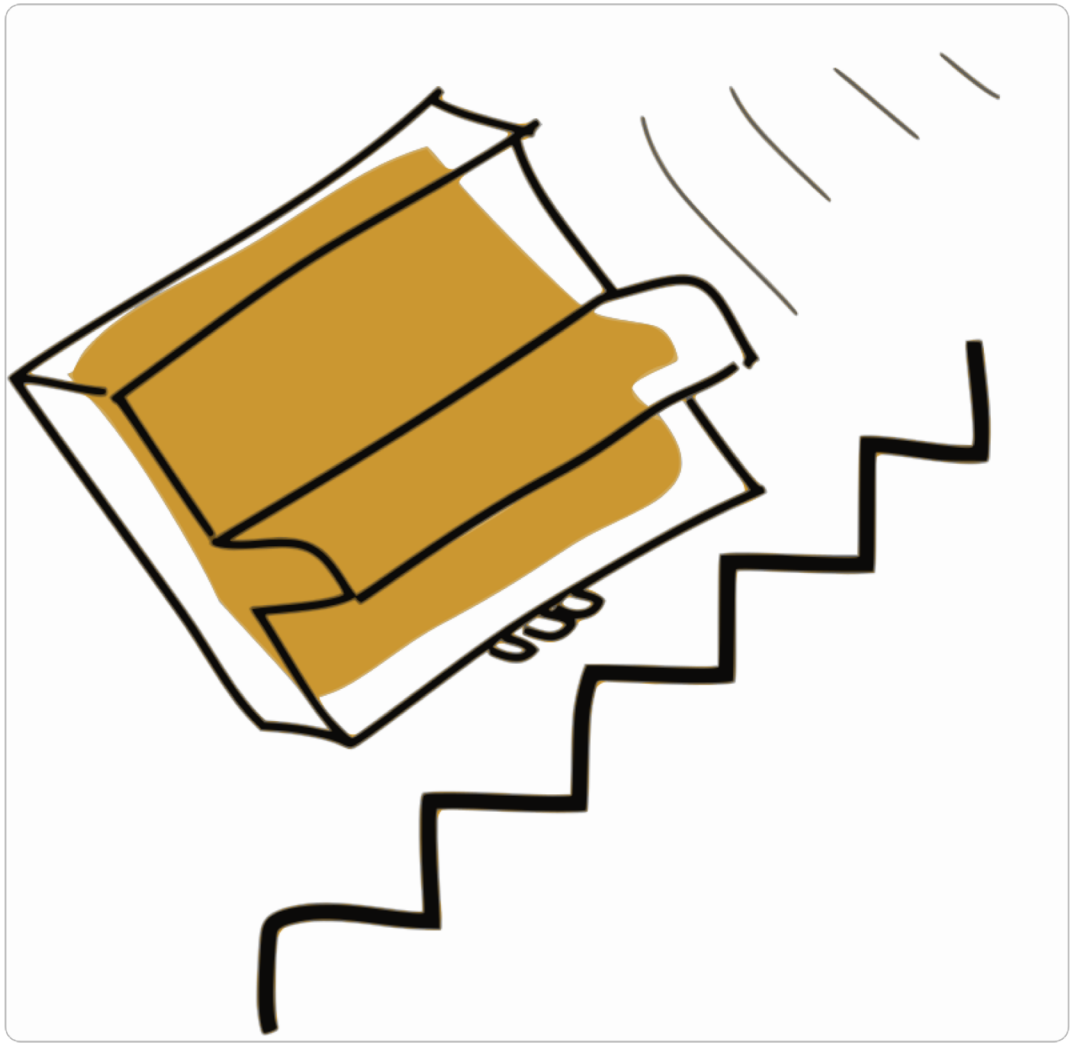
TEST EXECUTION



I told you to use
your hearts
when designing
the tests!



What’s the problem with that?

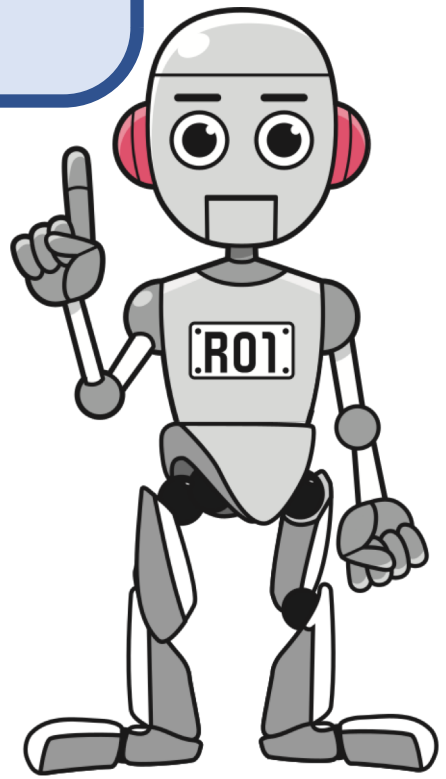


A systematic approach would be better!

TEST ANALYSIS
& TEST DESIG

TEST
CUTION

How can you automate me?



The Oracle Problem in Software Testing: A Survey

Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo

Abstract—Testing involves examining the behaviour of a system in order to discover potential faults. Given an input for a system, the challenge of distinguishing the corresponding desired, correct behaviour from potentially incorrect behavior is called the "test oracle
Withou
has int
testing
aware
forms
compre
testing

Index

1 INTRO

MUCH
much
to make te
this end, w
guishes be
the System

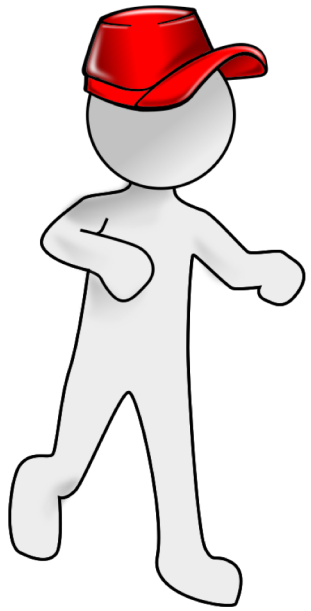
Howeve
the problem
icantly less attention, and remains comparatively less well-solved. This current open problem represents a significant bottleneck that inhibits greater test automation and uptake

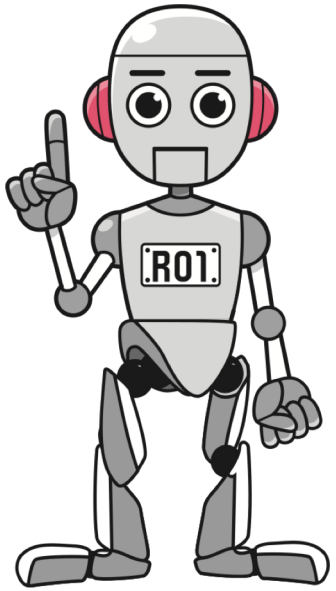
Where no full specification of the properties of the SUT exists, one may hope to construct a partial test oracle that can answer questions for some inputs. Such partial test

The literature on test oracles has introduced techniques for oracle automation, including modelling, specifications, contract-driven development and metamorphic testing. **When none of these is completely adequate, the final source of test oracle information remains the human**, who may be aware of informal specifications, expectations, norms and domain specific information that provide informal oracle guidance.

“Testing is different from writing tests. Developers write tests as a a way to give them space to think and confidence for refactoring. Testing focuses on finding bugs. Both should be done.”

https://medium.com/@mauricioaniche/testing-vs-writing-tests-d817bffea6bc





Find systematic and/or automated ways to design and execute tests!