

Unit testing vs Property-based testing

...

Kasper Liu
Gunnar Applelid

Do you know about these subjects?

...

Online poll in chat...

Why does testing benefit DevOps?

"Software testing is an **investigation** conducted to **provide stakeholders** with information about the **quality** of the **software**" - Wikipedia

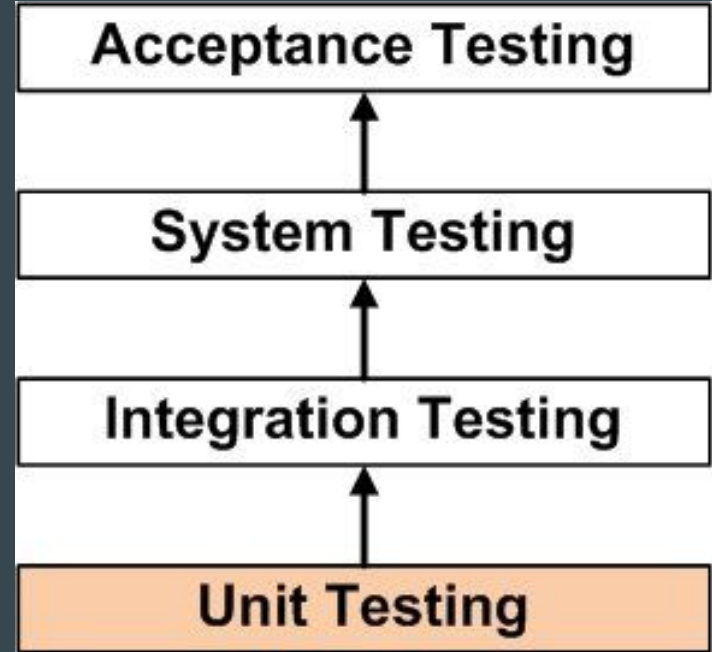
- Find bugs
- Code works after making changes
- Get more trust in our code...

**ANYTHING
THAT CAN GO WRONG
WILL GO WRONG**

Murphy's Law

Unit testing

- Unit is the smallest part of software
- Few inputs and outputs
- Method (object oriented programming)



Benefits

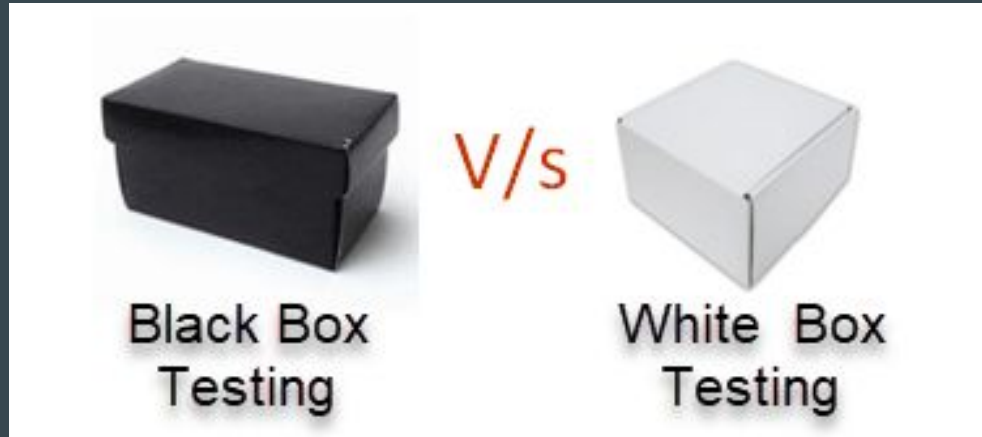
- Faster development - No need to start GUI
- Easier debugging
- Write the test first - Think about usage early
- Documentation

Not fun to write unit tests?



It's like doing the dishes....

White Box Testing



- Black Box Testing - We have **no** knowledge of internal structure
- White Box Testing - We **have** knowledge of internal structure
- **Unit Testing uses White Box Testing**

Common Tools

Unit Testing

- .NET xUnit.net
- Java **JUnit**
- PHP PHPUnit
- Ruby Test Unit
- Python PyUnit
- C Check
- C++ CppUnit

Property Testing

FsCheck
QuickTheories
PhpQuickCheck
Rantly
FactCheck
theft
RapidCheck

Unit testing - Given, When, Then

Given - some state

When - I perform an action

Then - I expect a certain result

Given, When, Then - JUnit

```
@Test
public void testMultiply(){
    UnitTest1 test = new UnitTest1();

    //Given
    int validResult = 10;
    //When
    int result = test.multiplyNumber( a: 5, b: 2);
    //Then
    assertEquals(validResult,result);
}
```

▼ ✓ UnitTest1Test	2 ms
✓ testMultiply	2 ms

Property Testing

Generative testing

- QuickCheck
- Quicktheories

Data \rightarrow Properties

VS

Properties \rightarrow Data

Property Testing

- **For any** input value (x,y,z, \dots)
- **Such that** precondition (x,y,z, \dots) is satisfied
- **Property** (x,y,z, \dots) must be true

Example multiplying integers - QuickTheories

- For any input value i
- Such that i is an integer
- $i * i \geq 0$ must be true

@Test

```
public void squaringAnIntegersAlwaysGivesAPositiveInteger(){  
    qt()  
        .forAll(integers().all())  
        .check((i) -> i*i >= 0);  
}
```

```
java.lang.AssertionError: Property falsified after 5 example(s)
```

```
Smallest found falsifying value(s) :-
```

```
46341
```

```
Other found falsifying value(s) :-
```

```
46344
```

```
46346
```

```
46360
```

```
46371
```

```
46739
```

```
46819
```

```
47684
```

```
48921
```

```
49282
```

```
49471
```

```
Seed was 662286519945976
```

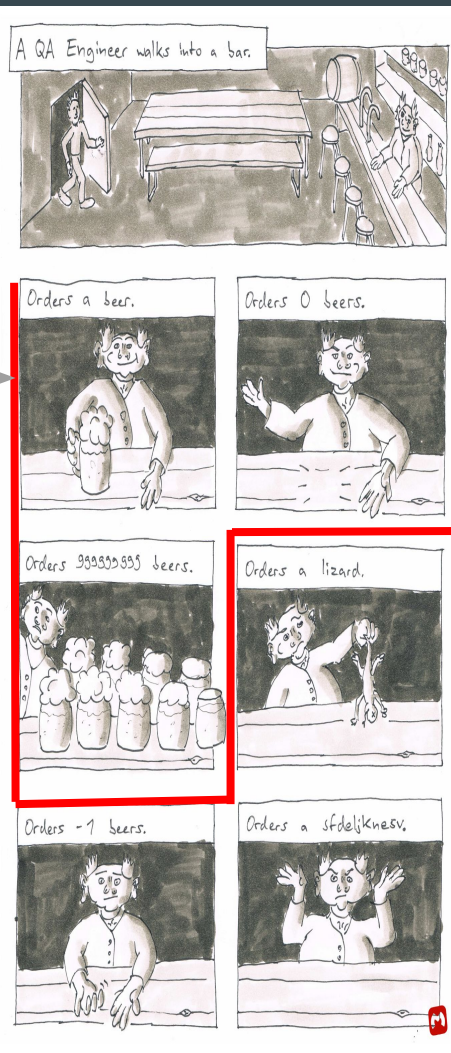
Benefits

- Covers a larger scope
- Shrink the input in case of failure
- Reproducible and replayable

Conclusion

- Edge cases
- Complementary
- Defined values vs Non defined values.

Property-based testing



Unit testing



Important take away message

- Use test method depending on your needs

YOU DECIDE!



References

- Software Testing
https://en.wikipedia.org/wiki/Software_testing
- Unit Testing
<http://softwaretestingfundamentals.com/unit-testing/>
https://en.wikipedia.org/wiki/Unit_testing
- White box Testing
<http://softwaretestingfundamentals.com/white-box-testing/>
- Property based testing
<https://www.leadingagile.com/2018/04/step-by-step-toward-property-based-testing/>
<https://bit.ly/2I1vpPt>
- QuickTheories
<https://github.com/quicktheories/QuickTheories>