



software testing

SANAE
DO LIVE with us

BEEЯ
28.- 29.9.22' **EX**

Testing data – golden vs. generic expected

Michał Pilarski

September 28, 2022

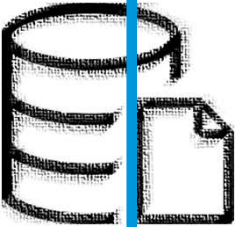
TECH6A

Data Lifecycle



collecting

storing



analyzing



transforming



sharing

Testing Data - definitions

Requirements (Reqs) - logic

convert from ... to...
reformat to ...
calculate based on ...
etc.

Test Set

input set, according to **requirements**
covers all possibilities (in perfect world)

Actual

result collected from dev system based on **reqs** and performed on **test set**

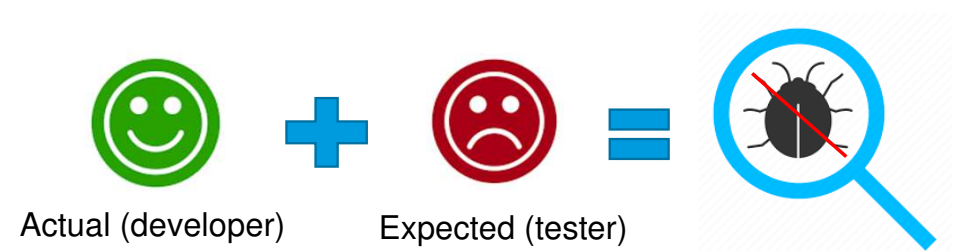
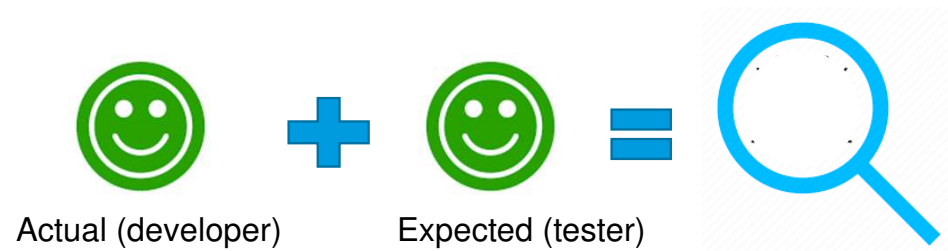
Expected

result prepared by tester based on **reqs** and linked with **test set**

Actual vs. Expected

“... expected is usually compared with actual result, and if the actual result differs from the expected one, the difference is documented and called a bug”.

Reqs and Actual vs. Expected



Testing Data – can we do it manually?

Reqs

```
source = 'LETTER'
target = 'CODE'

if input = 'A':
    expected = '1'
elif input = 'B':
    expected = '2'
elif input = 'C':
    expected = '3'
elif input = 'D':
    expected = '4'
else:
    expected = 'NULL'
```



Actual →

CODE
1
2
3
4
NULL



source = 'LETTER'
target = 'CODE'

LETTER
A
B
C
D
E

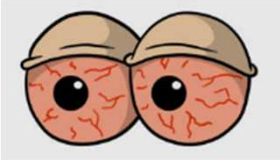
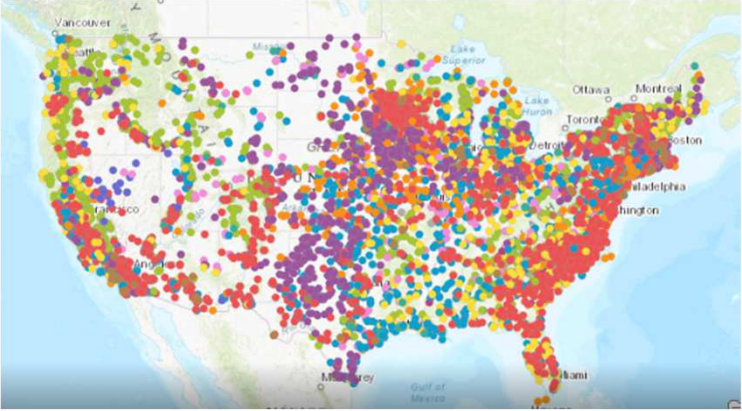
```
if input = 'A':
    expected = '1'
elif input = 'B':
    expected = '2'
elif input = 'C':
    expected = '3'
elif input = 'D':
    expected = '4'
else:
    expected = 'NULL'
```



Test Set

LETTER
A
B
C
D
E

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="xyaxis">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="xaxis"/>
        <xs:element ref="yaxis"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="xaxis">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="property"/>
        <xs:element ref="value"/>
        <xs:element name="unit"/>
      </xs:sequence>
      <xs:restriction base="xs:string">
        <xs:enumeration value="m"/>
        <xs:enumeration value="in"/>
      </xs:restriction>
    </xs:complexType>
  </xs:element>
  <xs:element name="yaxis">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="property"/>
        <xs:element ref="value"/>
        <xs:element ref="unit"/>
      </xs:sequence>
      <xs:restriction base="xs:string">
        <xs:enumeration value="m"/>
        <xs:enumeration value="in"/>
      </xs:restriction>
    </xs:complexType>
  </xs:element>
  <xs:attribute name="axisType" use="required" type="xs:NName"/>
</xs:element>
  <xs:element name="property">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="axisType" use="required" type="xs:NName"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="value" type="xs:decimal"/>
  <xs:element name="unit" type="xs:string"/>
</xs:schema>
```



Testing Data (Automatically) – golden expected

Reqs

```
source = 'LETTER'  
target = 'CODE'
```

```
if input = 'A':  
    expected = '1'  
elif input = 'B':  
    expected = '2'  
elif input = 'C':  
    expected = '3'  
elif input = 'D':  
    expected = '4'  
else:  
    expected = 'NULL'
```

Test Set

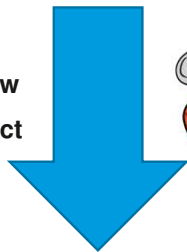
LETTER
A
B
C
D
E



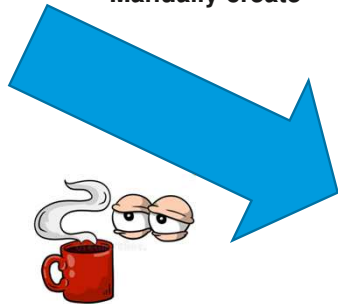
Actual

CODE
1
2
3
4
NULL

Copy
Review
Correct

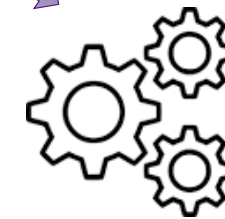


Manually create

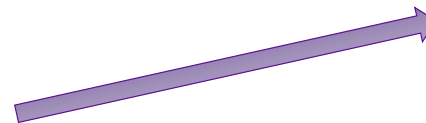
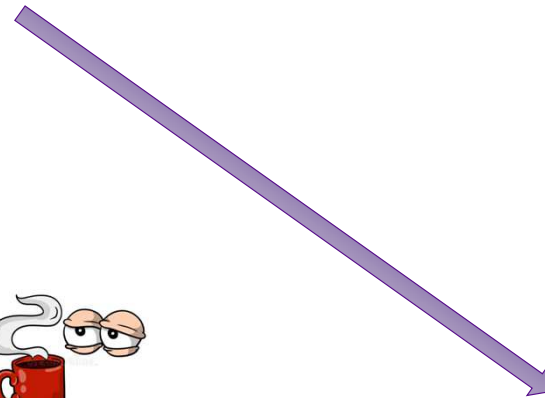


Expected

CODE
1
2
3
4
NULL



Test Assertion



Testing Data (Automatically) – golden expected - maintaining

Reqs

```
source = 'LETTER'  
target = 'CODE'
```

```
if input = 'A':  
    expected = '1'  
elif input = 'B':  
    expected = '2'  
elif input = 'C':  
    expected = '3'  
elif input = 'D':  
    expected = '4'  
else:  
    expected = 'NULL'
```

```
elif input = 'Z':  
    expected = '0'
```



Actual

CODE
1
2
3
4
NULL
0

Copy
Review
Correct

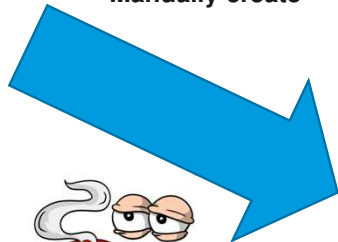


Manually create

Test Set

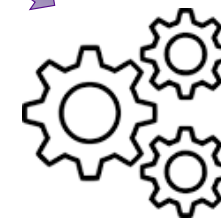
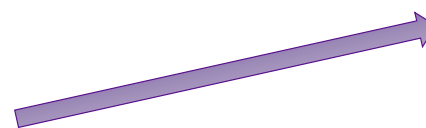
LETTER
A
B
C
D
E

Z

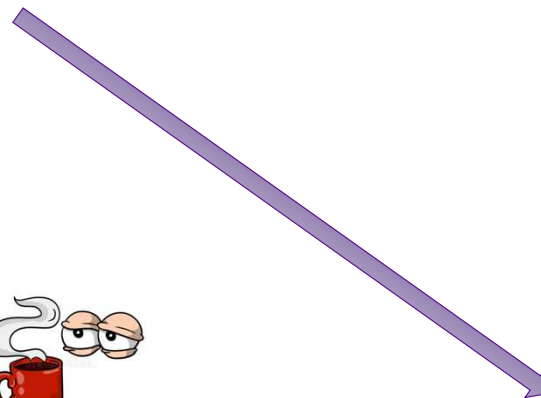


Expected

CODE
1
2
3
4
NULL
0



Test Assertion



Testing Data (Automatically) – generic expected

Reqs

```
source = 'LETTER'  
target = 'CODE'
```

```
if input = 'A':  
    expected = '1'  
elif input = 'B':  
    expected = '2'  
elif input = 'C':  
    expected = '3'  
elif input = 'D':  
    expected = '4'  
else:  
    expected = 'NULL'
```



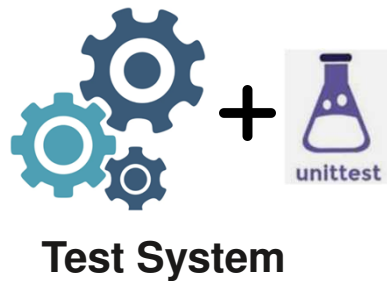
Actual

CODE
1
2
3
4
NULL



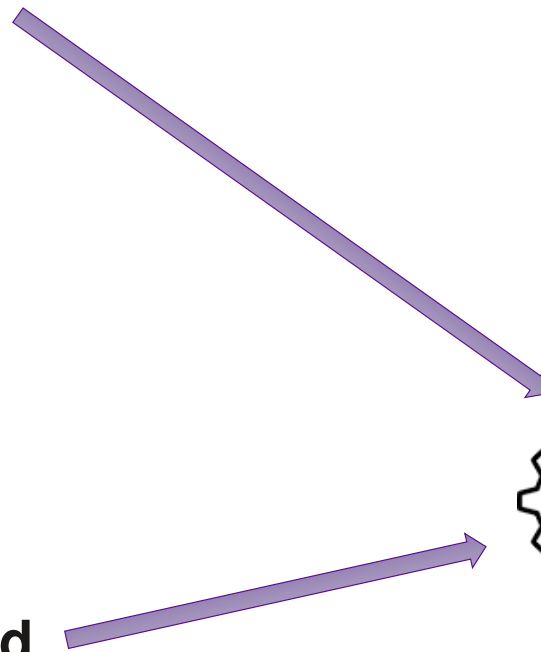
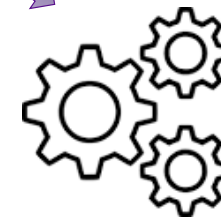
Test Set

LETTER
A
B
C
D
E



Expected

CODE
1
2
3
4
NULL



Testing Data (Automatically) – generic expected - maintaining

Reqs

```
source = 'LETTER'  
target = 'CODE'
```

```
if input = 'A':  
    expected = '1'  
elif input = 'B':  
    expected = '2'  
elif input = 'C':  
    expected = '3'  
elif input = 'D':  
    expected = '4'  
else:  
    expected = 'NULL'
```



```
elif input = 'Z':  
    expected = '0'
```

Develop



Test Set

LETTER
A
B
C
D
E

Z

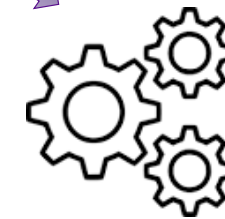


Actual

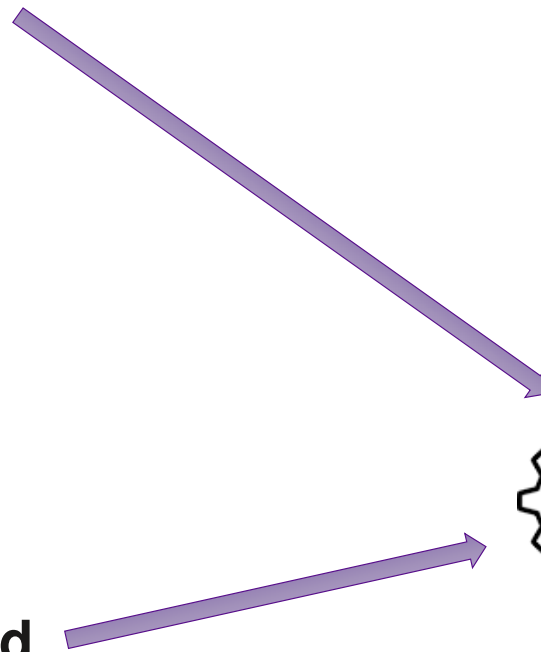
CODE
1
2
3
4
NULL
0

Expected

CODE
1
2
3
4
NULL
0



Test Assertion



Testing Data (Automatically) – golden vs. generic expected

Golden Expected

Prepared expected data set via:

1. **Develop** system
(review&correction needed)
2. Manually (:/)

- + easy and fast to prepare
- hard to maintain
- dependent on dev sys
- not fully reliable

Generic Expected

Prepared expected data set via **test** system

- time consuming to prepare
- + easy to maintain
- + totally independent
- + fully reliable (unit tests)



Thank you!