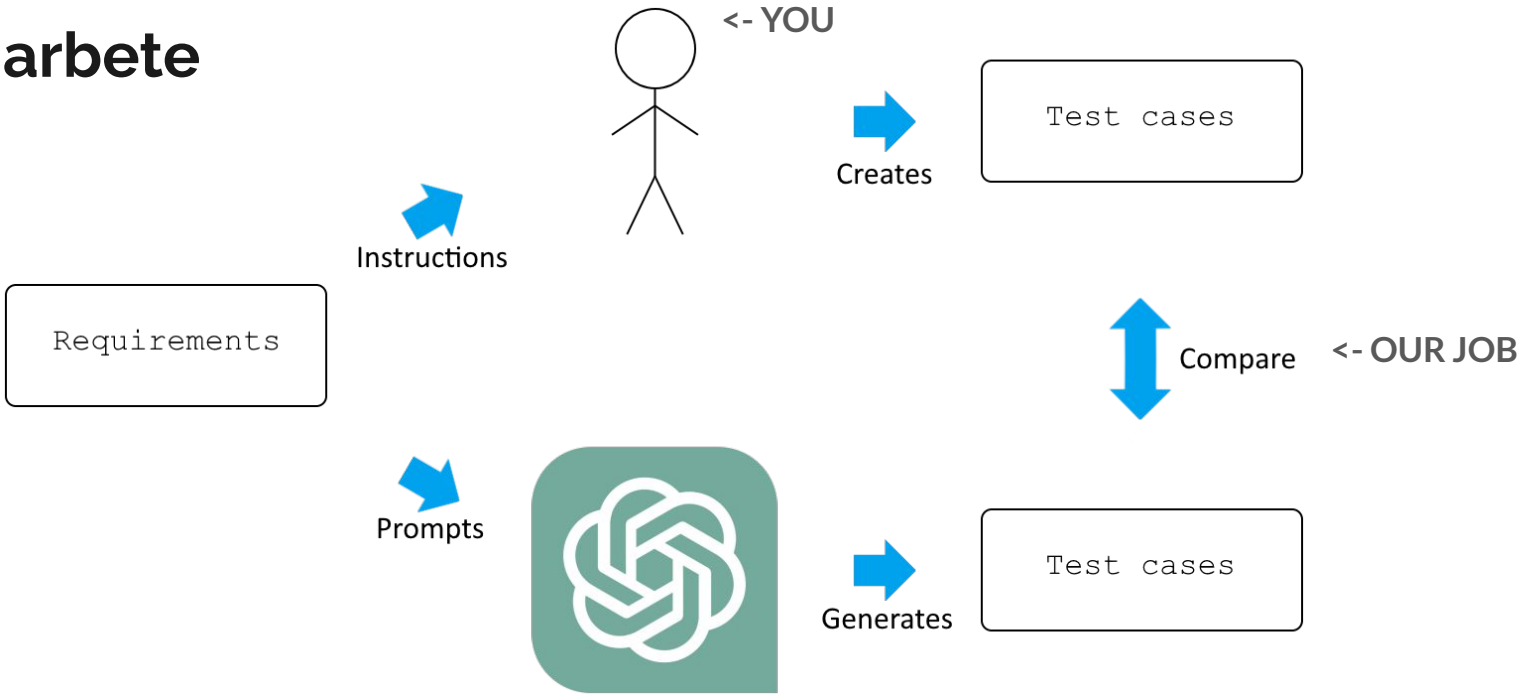
Automated test case generation from software requirements using ChatGPT

Intro till Black-box-testning

Ex-jobb 2024, Albin Andersson Axel Malmeling

Vårt arbete


<- YOU
Creates
Test cases
Instructions
Requirements
Compare
<- OUR JOB
Prompts
Generates
Test cases

Er uppgift

- Tre kompilerade Python-applikationer (ingen källkod)
- Läs och tyd kravspecifikationer
- Skapa testfall med hjälp av black-box-metodologi

- Hjälp från Internet är okej, men inte LLM/AI-baserade hjälpmedel
- Hjälp från oss är tillåtet, men inte hjälp från varandra
- Svaren bedöms och lagras anonymt

Unit-testning i Python

- unittest
- assertEqual
- assertRaises
- ValueError
- TypeError

```python
import unittest

class TestExample(unittest.TestCase):

    def test_correct_use(self):
        self.assertEqual(len(["a", "b", "c"]), 3)

    def test_wrong_type(self):
        with self.assertRaises(TypeError):
            len(1)

if __name__ == "__main__":
    unittest.main()
```

Black-box-testning

- Att testa mjukvara utan källkod
- Fullständig testning (exhaustive testing) - Oändliga inputs, oändliga outputs
- Krävs viss generalisering
- “Ett bra testfall reducerar det **totala antalet** testfall som behövs med **mer än ett**”

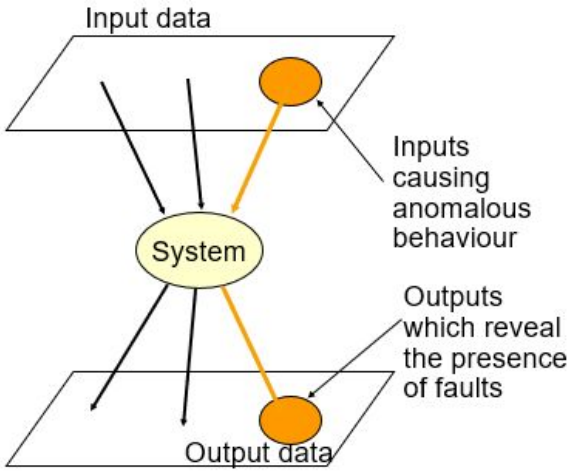
Black-box metodologi:

- Error guessing
- Equivalence partitioning
- Boundary-value analysis
- ~~Cause–effect graphing~~

x = [1, 10]

Klass 1:	*x = 0,99*	*Invalid*
Klass 2:	*x = 1*	*Valid*
Klass 3:	*x = 10*	*Valid*
Klass 4:	*x = 10,1*	*Invalid*

EQUIVALENCE PARTITIONING



Input- and output data can be grouped into classes where all members in the class behave in a comparable way.

- Define the classes
- Choose representatives
  - Typical element
  - Borderline cases

$x \in [25 .. 100]$

- class 1: $x < 25$
- class 2: $x >= 25$ and $x <= 100$
- class 3: $x > 100$

Exempel

Repetitiva testfall (som täcker samma “klass” av input) ❌

```
def test_too_many(self):
    self.assertEqual(len(["a", "b", "c"]), 3)
    self.assertEqual(len(["d", "e", "f"]), 3)
    self.assertEqual(len(["g", "h", "i"]), 3)
```

Varierade testfall ✅

```
def test_good_variety(self):
    self.assertEqual(len(["a", "b", "c"]), 3)
    self.assertEqual(len((1, 2, 3)), 3)
    self.assertEqual(len("abc"), 3)
```

Vad intresserar oss:

- **Täckning** (av många olika input-klasser)
- **Effektivitet** (hög täckning via få testfall)
- **Läsbarhet** (variabelnamn och kommentarer)
- **Passerad tid** (tiden det tar att skriva testfall)

Utlämning, Inlämning

Ni hittar arbetsmaterialet här, rekommenderas att ladda ned repo som zip

(denna presentation, samt exempel, finns även med):

- github.com/axmalm/Thesis-Lab-Session

Vid inlämning, zip:a din arbetsmapp igen och skicka till följande mejladress:

- albin.andersson.3@student.hv.se

Lycka till!