МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Тестирование и верификация программного обеспечения»

Практическая работа №2

Студенты группы *ИКБО-50-23*, *Иващенко А.В.*
Галкин М. В

(подпись)

Преподаватель *Ильичев Г. П.*

(подпись)

Отчет представлен «___»_____202__г.

Москва 2025 г.

1. Цели и задачи практической работы

Цель работы: познакомить студентов с процессом модульного и мутационного тестирования, включая разработку, проведение тестов, исправление ошибок, анализ тестового покрытия, а также оценку эффективности тестов путём применения методов мутационного тестирования.

Для достижения поставленной цели работы студентам необходимо выполнить ряд задач:

изучить основы мутационного тестирования и освоить инструменты для его выполнения (mutmut).

2. Ход работы

Листинг 1 – Исходный код Иващенко А. В.

```
import random


def rock_paper_scissors():
    sp = {'1': "Scissors", '2': "Paper", '3': "Rock"}
    k1, chs = random.choice(list(sp.items()))
    chs = k1+chs
    x = input("Are you ready?\nRock..Paper..Scissors!\n").capitalize()
    if not(x in sp.values()):
        return "No..."
    k2 = next((k for k, v in sp.items() if v == x), None)
    x = k2+x
    print(x[1:]," VS ",chs[1:])
    if (x == chs):
        return "Draw!"
    else:
        match(int(x[:1])):
            case 1:
                if int(chs[:1])==2:
                    return "You Win!"
                else:
                    return "You Lost!"
            case 2:
                if int(chs[:1])==3:
                    return "You Win!"
                else:
                    return "You Lost!"
            case 3:
                if int(chs[:1])==1:
                    return "You Win!"
                else:
                    return "You Lost!"


def head_or_tails():
    sp = ["Heads","Tails"]
    chs = random.choice(sp)
    x=input("Heads or Tails?\n").capitalize()
    if (x==chs):
        return "Correct!"
    elif x in sp:
        return "You Lost!"
    else:
        return "...Huh? Anyway..You Lost!"
```

```
def guess_number():
    sp = random.randint(1, 200)
    print("Can you guess a number?")
    while True:
        x=int(input("Take a guess!\n"))
        if x>sp:
            print("Too much! Try lesser number.")
        elif x<sp:
            print("Not enough! Try bigger number.")
        else:
            return "Correct! Good job!"

def hangman():
    sp = ["determination","patience","bravery","justice","integrity","kindness","perseverance"]
    chs = random.choice(sp)
    att = 7
    gs = set()
    print("Can you guess a word?")
    while att > 0:
        show = " ".join([letr if letr in gs else "_" for letr in chs])
        print(show)
        x = input("Take a guess on letter: ")
        if x in chs:
            print("Correct letter!")
            gs.add(x)
            if all(ch in gs for ch in chs):
                return "You Win!\nCorrect word is " + chs
        else:
            print("Wrong letter...")
            att -= 1
    return "You couldn't guess word... Correct answer was " + chs

def dice_battle():
    print("Ready to throw a dice?")
    hp = 1
    ehp = 1
    print("You both had 5 hp!")
    while (hp>0 and ehp>0):
        x = input("Press Enter to throw a dice!")
        yu = random.randint(1, 6)
        en = random.randint(1, 6)
        print("You threw",yu,"and your enemy threw",en)
        if yu > en:
            print("You Won!")
            ehp-=1
        elif yu < en:
            print("You Lost..")
```

```
            hp-=1
        else:
            print("Draw!")
    if (hp==0):
        return "You died...Game over!"
    else:
        return "Congratulations! Hp left... " + str(hp)
```

Листинг 2 – Модульное тестирование Галкина М. В. по коду Иващенко А. В.

```
import pytest
from unittest.mock import patch
import module as a

@patch('builtins.input',return_value="Heads")
@patch("random.choice",return_value="Heads")
def test_hh_head_or_tails(mock_input, mock_choice):
    assert a.head_or_tails() == "Correct!"

@patch('builtins.input',return_value="Heads")
@patch("random.choice",return_value="Tails")
def test_ht_head_or_tails(mock_input, mock_choice):
    assert a.head_or_tails() == "You Lost!"

@patch('builtins.input',return_value="Tails")
@patch("random.choice",return_value="Heads")
def test_th_head_or_tails(mock_input, mock_choice):
    assert a.head_or_tails() == "You Lost!"

@patch('builtins.input',return_value="Tails")
@patch("random.choice",return_value="Tails")
def test_tt_head_or_tails(mock_input, mock_choice):
    assert a.head_or_tails() == "Correct!"

@patch('builtins.input',return_value="tails")
@patch("random.choice",return_value="Tails")
def test_cc_head_or_tails(mock_input, mock_choice): #Check capitalize
    assert a.head_or_tails() == "Correct!"

@patch('builtins.input',return_value="ITSPIKACHU")
def test_rr_head_or_tails(mock_input):
    assert a.head_or_tails() == "...Huh? Anyway..You Lost!"

@patch("random.randint",return_value=10)
@patch('builtins.input',return_value="10")
def test_ft_guess_number(mock_input, mock_randint, capsys): #First try
    res = a.guess_number()
    cap =  capsys.readouterr()
    assert res == "Correct! Good job!"
```

```
    assert "Can you guess a number?" in cap.out
    assert "Too much! Try lesser number." not in cap.out
    assert "Not enough! Try bigger number." not in cap.out

@patch("random.randint",return_value=10)
@patch('builtins.input',side_effect=["2","10"])
def test_lt_guess_number(mock_input, mock_randint, capsys): #Less try
    res = a.guess_number()
    cap = capsys.readouterr()
    assert res == "Correct! Good job!"
    assert "Can you guess a number?" in cap.out
    assert "Too much! Try lesser number." not in cap.out
    assert "Not enough! Try bigger number." in cap.out

@patch("random.randint",return_value=10)
@patch('builtins.input',side_effect=["12","10"])
def test_bt_guess_number(mock_input, mock_randint, capsys): #Big try
    res = a.guess_number()
    cap = capsys.readouterr()
    assert res == "Correct! Good job!"
    assert "Can you guess a number?" in cap.out
    assert "Too much! Try lesser number." in cap.out
    assert "Not enough! Try bigger number." not in cap.out

@patch("random.randint",return_value=10)
@patch('builtins.input',side_effect=["100","1","10"])
def test_bt_guess_number(mock_input, mock_randint, capsys): #Big try
    res = a.guess_number()
    cap = capsys.readouterr()
    assert res == "Correct! Good job!"
    assert cap.out.splitlines()[1] == "Too much! Try lesser number."
    assert cap.out.splitlines()[2] == "Not enough! Try bigger number."

#rps = rock paper scissorgshfk Ножницы короче
@patch('builtins.input',return_value="rock")
@patch('random.choice',return_value=('2','Paper'))
def test_cc_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="You Lost!"

@patch('builtins.input',return_value="Rock")
@patch('random.choice',return_value=('2','Paper'))
def test_l_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="You Lost!"

@patch('builtins.input',return_value="Scissors")
@patch('random.choice',return_value=('2','Paper'))
def test_w_rps(mock_input,mock_choice):
```

```
    assert a.rock_paper_scissors()=="You Win!"

@patch('builtins.input',return_value='Paper')
@patch('random.choice',return_value=('2','Paper'))
def test_d_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="Draw!"

@patch('builtins.input',return_value="Эксперимент не был провальным")
@patch('random.choice',return_value=('2','Paper'))
def test_ri_rps(mock_input,mock_choice):
    assert a.rock_paper_scissors()=="No..."

@patch('builtins.input',return_value="qxzcdwu")
@patch('random.choice',return_value="bravery")
def test_l_hangman(mock_input,mock_choice,capsys):
    res = a.hangman()
    cap = capsys.readouterr()
    assert res == "You couldn't guess word... Correct answer was bravery"
    assert "Can you guess a word?" in cap.out
    assert "Wrong letter..." in cap.out

@patch('builtins.input',side_effect=["a","b","v","e","r","y"])
@patch('random.choice',return_value="bravery")
def test_w_hangman(mock_choice,mock_input,capsys):
    res = a.hangman()
    cap = capsys.readouterr()
    assert res == "You Win!\nCorrect word is bravery"
    assert "Can you guess a word?" in cap.out
    assert "Wrong letter..." not in cap.out

@patch('builtins.input',side_effect=["Выкидываем
игру","5","f","c","d","q","a","b","v","e","r","y"])
@patch('random.choice',return_value="bravery")
def test_cl_hangman(mock_choice,mock_input,capsys):
    res = a.hangman()
    cap = capsys.readouterr()
    assert res == "You Win!\nCorrect word is bravery"
    assert "Can you guess a word?" in cap.out
    assert "Wrong letter..." in cap.out
    assert "Correct letter!" in cap.out
    assert "_ _ a _ _ _ _" in cap.out

@patch('builtins.input',return_value="")
@patch('random.randint',side_effect=[6,1])
def test_w_dice_battle(mock_input,mock_choice, capsys):
    res = a.dice_battle()
    cap = capsys.readouterr()
```

```
    assert res == "Congratulations! Hp left... 1"
    assert "You Won!" in cap.out
    assert "Ready to throw a dice?" in cap.out
    assert "You both had 5 hp!" in cap.out

@patch('builtins.input',return_value="")
@patch('random.randint',side_effect=[1,6])
def test_l_dice_battle(mock_input,mock_choice, capsys):
    res = a.dice_battle()
    cap = capsys.readouterr()
    assert res == "You died...Game over!"
    assert "You Lost.." in cap.out
    assert "Ready to throw a dice?" in cap.out
    assert "You both had 5 hp!" in cap.out

@patch('builtins.input',return_value="")
@patch('random.randint',side_effect=[1,1,6,1])
def test_dw_dice_battle(mock_input,mock_choice, capsys):
    res = a.dice_battle()
    cap = capsys.readouterr()
    assert res == "Congratulations! Hp left... 1"
    assert "You Won!" in cap.out
    assert "Ready to throw a dice?" in cap.out
    assert "You both had 5 hp!" in cap.out
    assert "Draw!" in cap.out
```

Исходный код представляет из себя 5 функций. Каждая функция – игра, написанная на языке Python: Камень Ножницы Бумага, Орел или Решка, отгадай число, Висельница, пвп с кубиками.

К исходному коду другим участником команды были прописаны тестирования, покрывающие все функции.

Для мутационного тестирования данные файлы были перенесены на виртуальную машину, имитирующую Unix-подобную систему, для работы с мутационным тестированием через mutmut.

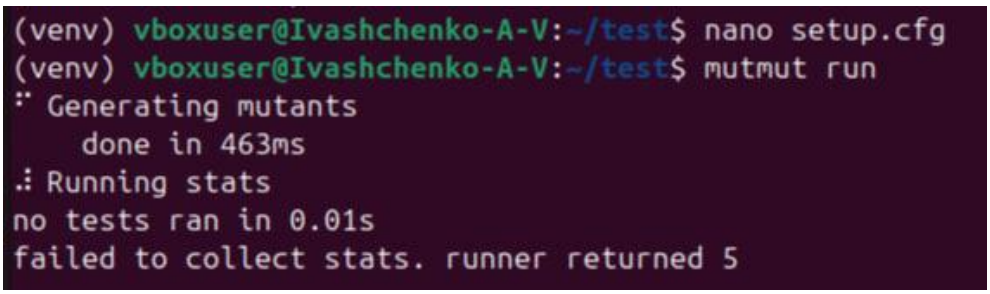

Рисунок 1 – Создание мутантов

Рисунок 2 – Проверка мутантов (никто не выжил – F=Failed)

По результатам мутационного тестирования ни один из мутантов не выжил.

Листинг 3 – Исходный код Галкина М. В.

```
import random

def generate_surname(is_female):
    common_surnames = ["Иванов", "Петров", "Сидоров", "Кузнецов", "Смирнов", "Попов", "Васильев", "Соколов", "Михайлов", "Фёдоров"]
    prefixes = ["Бело", "Ново", "Старо", "Красно", "Черно", "Мало", "Велико", "Добро"]
    bases = ["вин", "бор", "град", "мир", "род", "слав", "поль", "рус", "ход", "яр"]
    suffixes_male = ["ов", "ев", "ин", "ский", "цкий", "ников", "ич", "енко"]
    suffixes_female = ["ова", "ева", "ина", "ская", "цкая", "никова", "ич", "енко"]

    if random.random() < 0.5:
        surname = random.choice(common_surnames)
        if is_female:
            return surname+"a"
    else:
        surname = random.choice(prefixes) + random.choice(bases) + random.choice(
            suffixes_male if not is_female else suffixes_female)

    return surname

def generate_full_name():
    name_index = random.randint(0, len(names) - 1)
```

```
    name = names[name_index]

    is_female = name_index < 20
    initial = random.choice(name)[0].upper() + "."
    surname = generate_surname(is_female)
    return f"{name} {initial} {surname}"

def generate_born_date():
    year = random.randint(1960,2011)
    day = random.randint(1,31)
    moth = random.randint(1,12)
    if moth == 2 and day>28 and year%4==0 and year%100!=0:
        day = 29
    elif moth == 2 and day>28:
        day = 28
    elif (moth == 4 or moth == 6 or moth == 11 or moth == 9) and day>30:
        day = 30
    return f"{day}.{moth}.{year}"

def generatepasnum():
    seria1 = random.randint(10,50)
    seria2 = random.randint(10,20)
    num = random.randint(0,999999)
    for i in range(6):
        if len(str(num)) < 6:
            num = "0"+str(num)
    return f"{seria1} {seria2} {num}"

def displaypasinfo(name,num,borndate):
    print(f"ФИО - {name}")
    print(f"Дата рождения - {borndate}")
    print(f"Серия и Номер - {num}")

def validitycheck(pasinfo):
    if int(pasinfo[1].split(" ")[2])<0 or int(pasinfo[1].split(" ")[2])>999999:
        return False
    elif int(pasinfo[1].split(" ")[1])>20 or int(pasinfo[1].split(" ")[1])<10:
        return False
    elif int(pasinfo[1].split(" ")[0])>50 or int(pasinfo[1].split(" ")[0])<10:
        return False
    else: return True
```

Листинг 4 – Модульное тестирование Иващенко А. В. по коду Галкина М. В.

```
import pytest
from unittest.mock import patch
import module_G as a

@patch("random.random",return_value=0)
```

```
@patch("random.choice",return_value="Сидоров")
def test_cmn_gs(mock_random,mock_choice):
    assert a.generate_surname(False) == "Сидоров"
    assert a.generate_surname(True) == "Сидорова"

@patch("random.random",return_value=0.6)
@patch("random.choice",side_effect=["Бело","бор","ов"])
def test_bsm_m_gs(mock_random,side_effect):
    assert a.generate_surname(False) == "Белоборов"

@patch("random.random",return_value=0.6)
@patch("random.choice",side_effect=["Бело","бор","ова"])
def test_bsm_f_gs(mock_random,side_effect):
    assert a.generate_surname(True) == "Белоборова"

@patch("random.randint",return_value=1)
@patch("random.random",return_value=0)
@patch("random.choice",side_effect=["А","Сидоров"])
def test_f2_gfn(mock_randint,mock_random,mock_choice):
    names = ["София", "Анна", "Мария", "Ева", "Виктория", "Полина", "Алиса", "Варвара",
"Василиса", "Александра", "Елизавета", "Арина", "Ксения", "Екатерина", "Дарья",
"Милана", "Анастасия", "Мирослава", "Вероника", "Кира", "Михаил", "Александр",
"Максим", "Артем", "Марк", "Лев", "Иван", "Матвей", "Даниил", "Дмитрий", "Тимофей",
"Роман", "Мирон", "Мухаммад", "Кирилл", "Егор", "Илья", "Алексей", "Константин",
"Федор"]
    assert a.generate_full_name() == "Анна А. Сидорова"

@patch("random.randint",return_value=20)
@patch("random.random",return_value=0)
@patch("random.choice",side_effect=["А","Сидоров"])
def test_m1_gfn(mock_randint,mock_random,mock_choice):
    names = ["София", "Анна", "Мария", "Ева", "Виктория", "Полина", "Алиса", "Варвара",
"Василиса", "Александра", "Елизавета", "Арина", "Ксения", "Екатерина", "Дарья",
"Милана", "Анастасия", "Мирослава", "Вероника", "Кира", "Михаил", "Александр",
"Максим", "Артем", "Марк", "Лев", "Иван", "Матвей", "Даниил", "Дмитрий", "Тимофей",
"Роман", "Мирон", "Мухаммад", "Кирилл", "Егор", "Илья", "Алексей", "Константин",
"Федор"]
    assert a.generate_full_name() == "Михаил А. Сидоров"

@patch("random.randint",side_effect = [1991,31,2])
def test_2_and_big_gbd(mock_randint):
    assert a.generate_born_date() == "28.2.1991"

@patch("random.randint",side_effect = [2009,31,4])
def test_4_and_big_gbd(mock_randint):
    assert a.generate_born_date() == "30.4.2009"
```

```
@patch("random.randint",side_effect = [2004,30,2])
def test_vesokosny_gbd(mock_randint):
    assert a.generate_born_date() == "29.2.2004"

@patch("random.randint")
def test_gpn(mock_randint):
    mock_randint.side_effect = [20,15,2000]
    assert a.generatepasnum() == "20 15 002000"
    mock_randint.side_effect = [50,20,999999]
    assert a.generatepasnum() == "26 12 199100"
    mock_randint.side_effect = [26,12,199100]
    assert a.generatepasnum() == "26 12 199100"


def test_dpi(capsys):
    res = a.displaypasinfo("LEEROOOOY JENKINS!","88 00 5553535","11.09.2001")
    cap =  capsys.readouterr()
    assert "ФИО - LEEROOOOY JENKINS!" in cap.out
    assert "Дата рождения - 11.09.2001" in cap.out
    assert "Серия и Номер - 88 00 5553535" in cap.out

def test_vc():
    assert  a.validitycheck(["LEEROOOOY  JENKINS!","88  00  5553535","11.9.2001"])  ==
False
    assert a.validitycheck(["LEEROOOOY JENKINS!","50 20 999999","11.9.2001"]) == True
```

Исходный код представляет из себя 6 функций: Создание фамилии, создание полного имени, создание даты рождения, создание номера паспорта, вывод информации и проверка валидации данных паспорта.

К исходному коду другим участником команды были прописаны тестирования, покрывающие все функции.

Для мутационного тестирования данные файлы были перенесены на виртуальную машину, имитирующую Unix-подобную систему, для работы с мутационным тестированием через mutmut

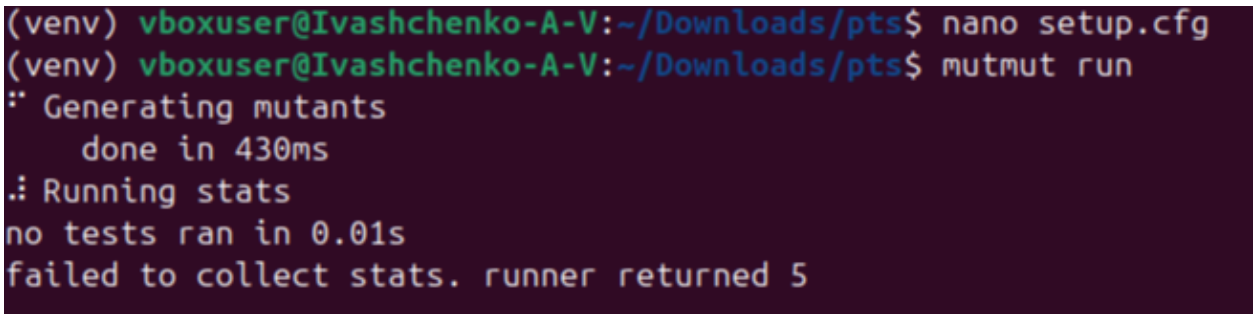

Рисунок 3 – Создание мутантов

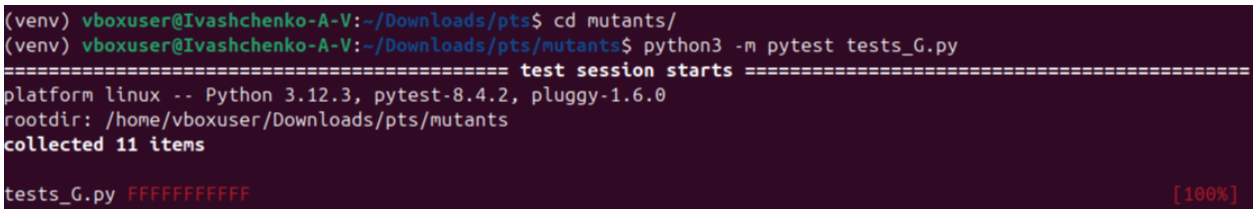

Рисунок 4 – Проверка мутантов (никто не выжил – F=Failed)

По результатам мутационного тестирования ни один из мутантов не выжил.

3. Анализ и выводы

Тестирования, проведенные участниками команды, продемонстрировали свою стойкость при мутационном тестировании. Тесты покрыли основной функционал каждой из функций модулей.

Программа Иващенко содержала избыточную конструкцию в функции rock_paper_scissors() и возможность образования цикла while в функции hangman(). Устранением неполадок является добавление в функцию hangman() списка уже отвеченных букв. Упрощение структуры выбора системой предметов в функции rock_paper_scissors(), заменяя картеж списком.

Листинг 5 – Исправление неполадок в коде Иващенко А. В.

```
def rock_paper_scissors():
    sp = ["Scissors","Paper","Rock"]
    chs = random.choice(sp)
    x = input("Are you ready?\nRock..Paper..Scissors!\n").capitalize()
    if not(x in sp):
        return "No..."
    print(x," VS ",chs)
    if (x == chs):
        return "Draw!"
    else:
        if x=="Scissors":
            if chs=="Paper":
                return "You Win!"
            else:
                return "You Lost!"
        elif x=="Paper":
            if chs=="Rock":
                return "You Win!"
            else:
                return "You Lost!"
        elif x=="Rock":
            if chs=="Scissors":
                return "You Win!"
            else:
                return "You Lost!"
```

Программа Галкина имеет не полную проверку валидации данных в функции validitycheck(pasinfo). Устранением неполадок является добавление проверок на корректность даты рождения.

Листинг 6 – Исправление неполадок в коде Иващенко А. В.

```
def validitycheck(pasinfo):
    if int(pasinfo[1].split(" ")[2])<0 or int(pasinfo[1].split(" ")[2])>999999:
        return False
    elif int(pasinfo[1].split(" ")[1])>20 or int(pasinfo[1].split(" ")[1])<10:
        return False
    elif int(pasinfo[1].split(" ")[0])>50 or int(pasinfo[1].split(" ")[0])<10:
        return False
    elif int(passinfo[2].split(".")[0])<0 or int(passinfo[2].split(".")[0])>31:
        return False
    elif int(passinfo[2].split(".")[2])<1960 or int(passinfo[2].split(".")[2])>2011:
        return False
    elif int(passinfo[2].split(".")[1]) < 1 or int(passinfo[2].split(".")[1]) > 12:
        return False
    else: return True
```

В ходе работы мы научились работать с библиотекой pytest, проводить модульное тестирование с помощью этой библиотеки, познакомились с библиотекой mutmut, и получили навыки по проведению мутационного тестирования с использованием этой библиотеки.