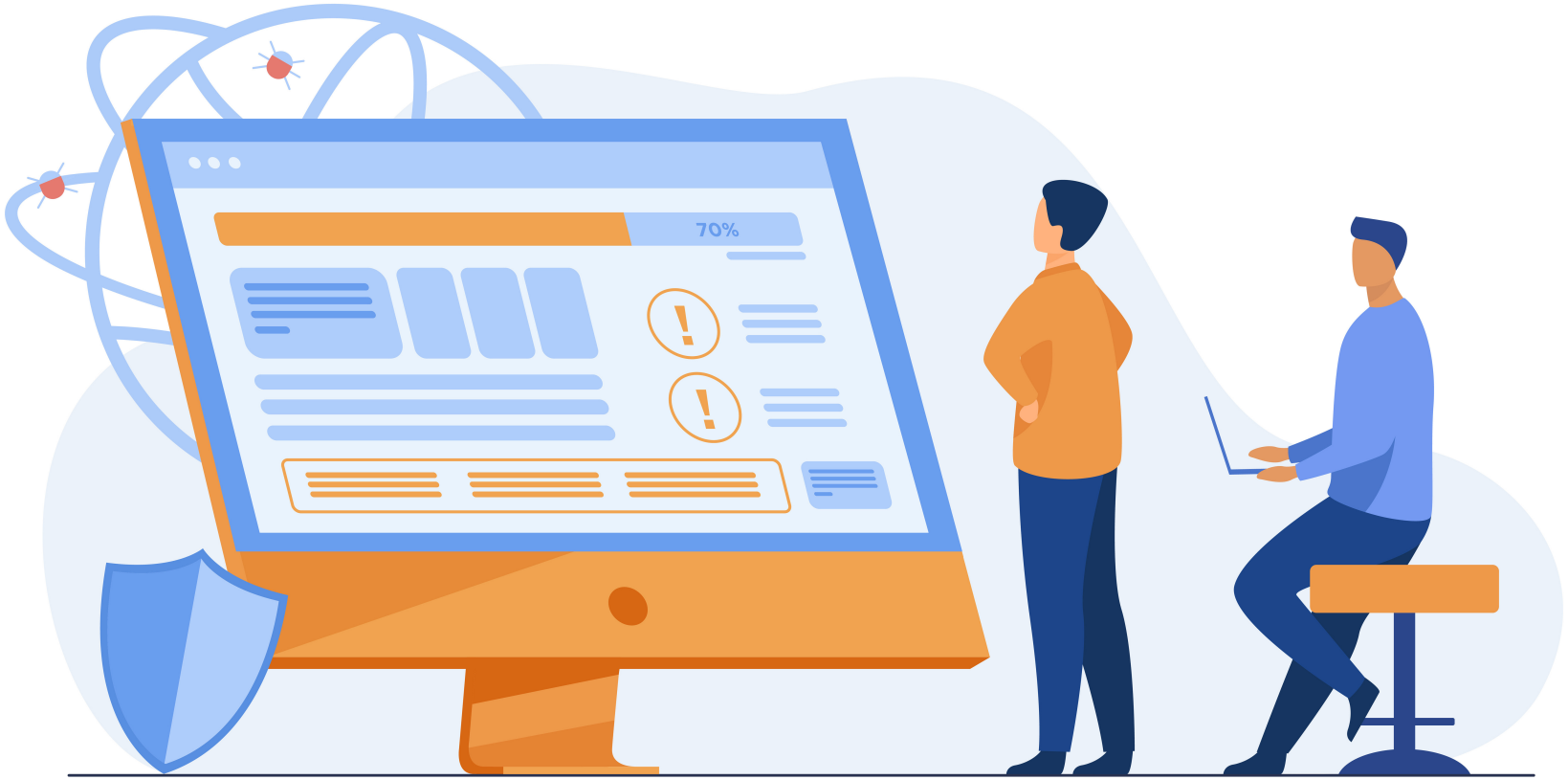
PENGUJIAN PERANGKAT LUNAK

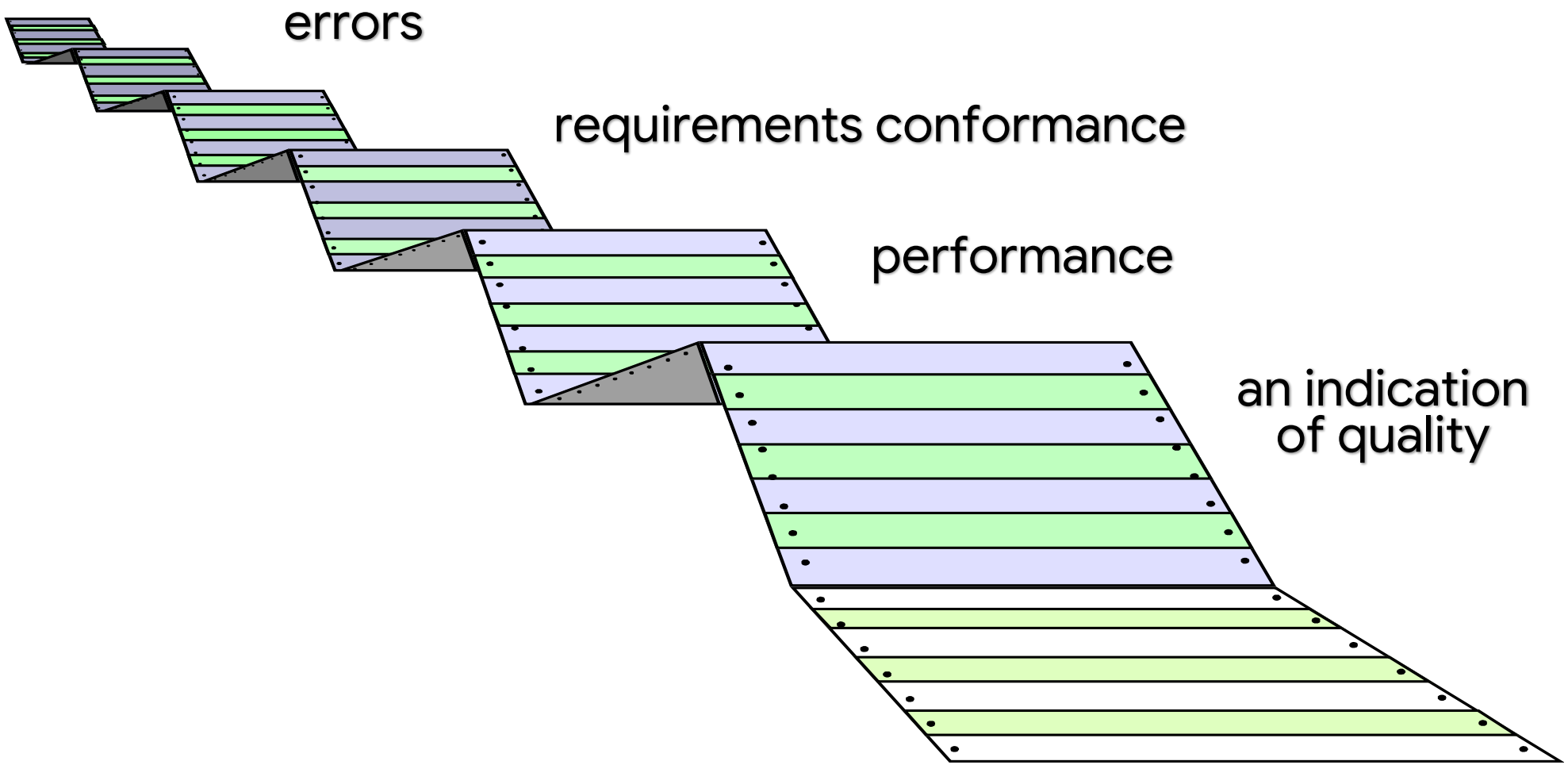


Apa itu pengujian perangkat lunak?

Proses **menelusuri dan mempelajari** sebuah program dalam rangka **menemukan kesalahan** pada perangkat lunak sebelum diserahkan kepada pengguna.

Sumber: Software Engineering, Roger S. Pressman 7th edition

Apa saja yang diperiksa?



“Pengujian yang baik **bukan untuk memastikan tidak ada kesalahan** tetapi **mencari sebanyak mungkin kesalahan**”

Pendekatan Pengujian Perangkat Lunak

MANUAL TESTING

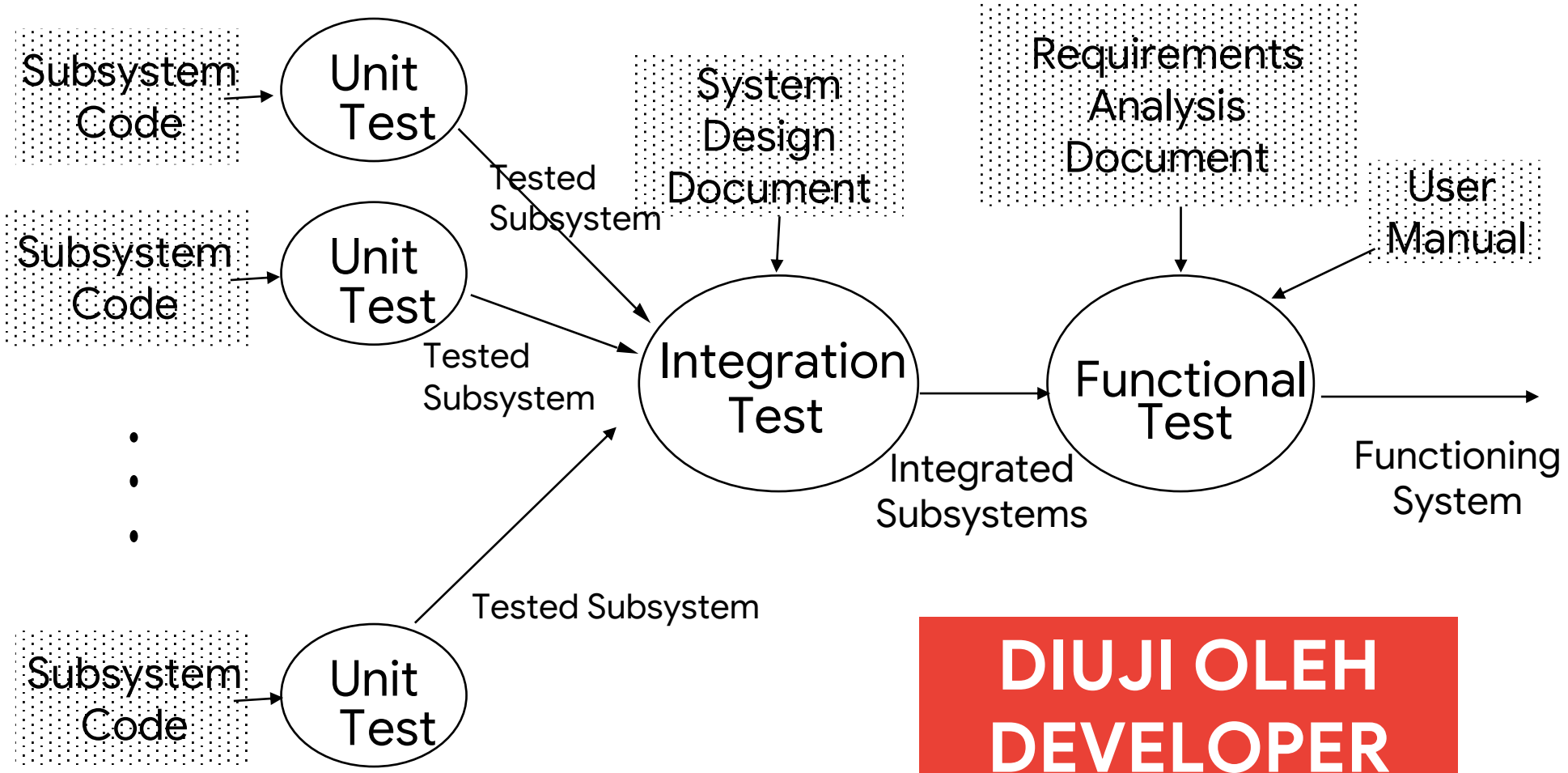
Pengujian perangkat lunak dimana proses pengujian dieksekusi manual oleh QA (Quality Assurance) Analyst. Tujuannya untuk menemukan bugs ketika perangkat lunak sedang under development.

AUTOMATED TESTING

Pengujian perangkat lunak dimana tester menuliskan kode/script pengujian untuk mengautomasi eksekusi pengujian. Tester menggunakan alat bantu automasi untuk mendevelop script pengujian dan memvalidasi perangkat lunak.

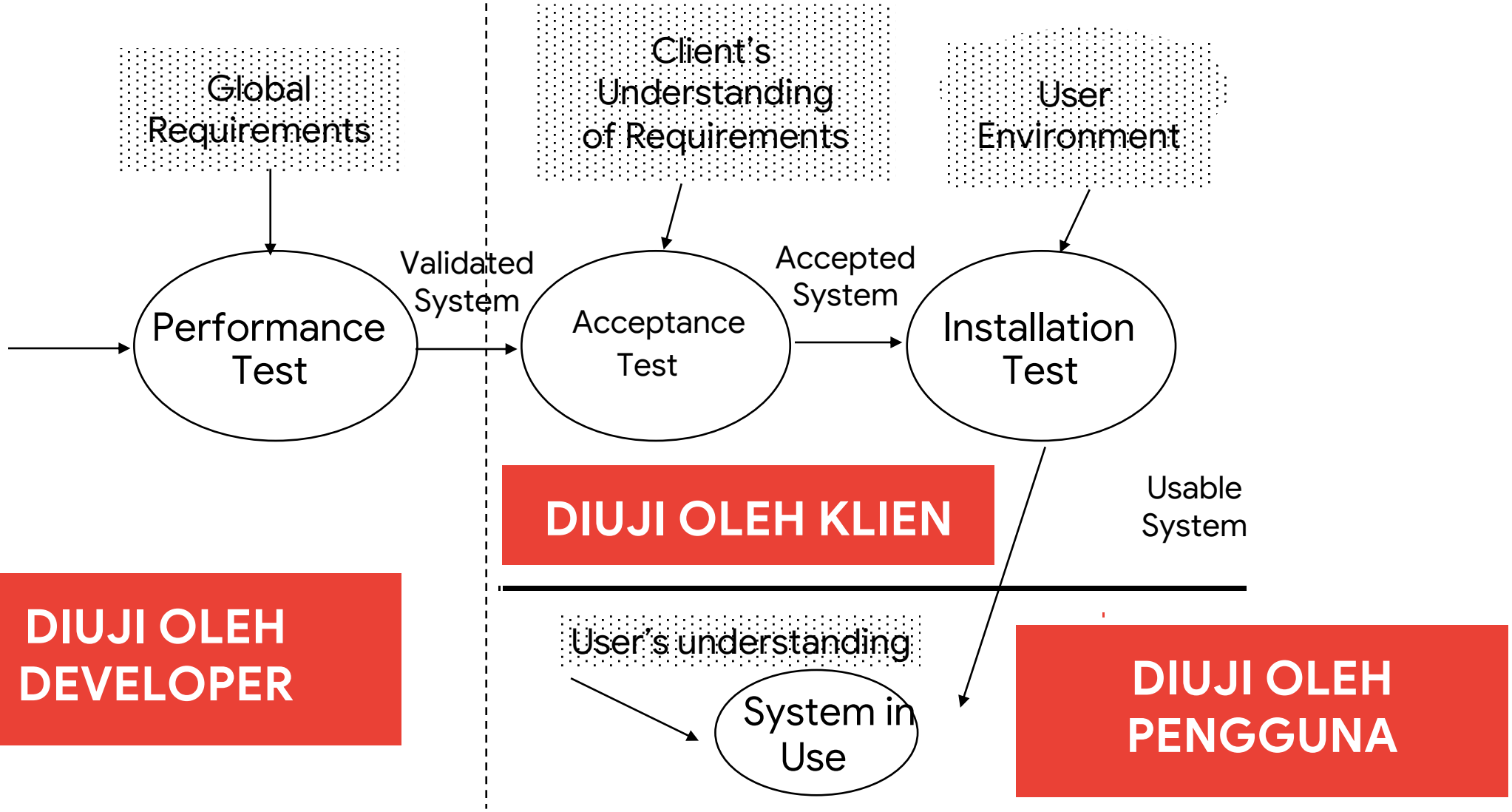
“Pengujian dilakukan untuk dua hal dalam sebuah perangkat lunak, yaitu **pengujian fungsional** dan **non-fungsional**.”

Aktivitas dan strategi pengujian (1)



DIUJI OLEH DEVELOPER

Aktivitas dan strategi pengujian (2)

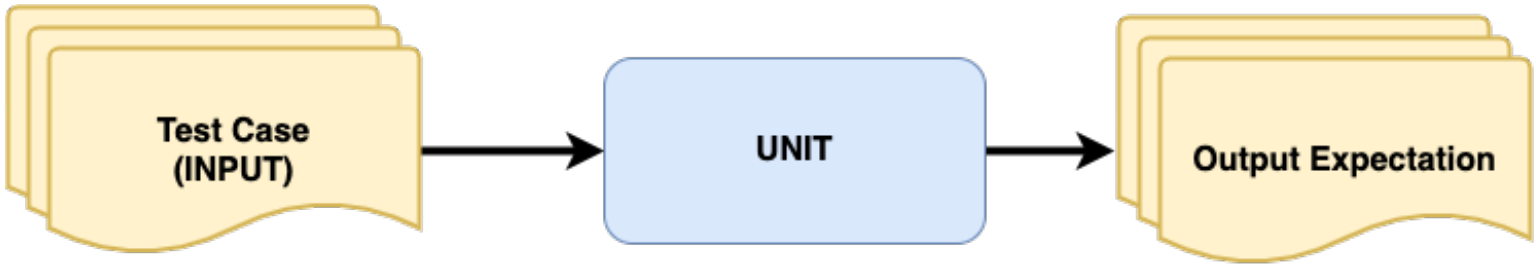


“Aktivitas dan strategi pengujian bisa **disesuaikan dengan kebutuhan pengujian**. Functional dan performance test mewakili uji fungsional dan non-fungsional.”

Unit Testing

UNIT TESTING

Pengujian untuk memverifikasi apakah suatu bagian program yang **kecil dan terisolasi** itu benar.



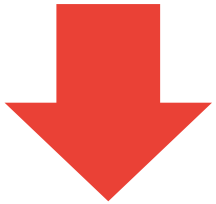
(2,3) → tambah(a,b) → 5

(2,-2) → tambah(a,b) → 0

("2",3) → tambah(a,b) → TypeError

MANUAL UNIT TESTING

```python
#!/usr/bin/env python3

def tambah_bil(a,b):
    return a + b;
```



```python
#!/usr/bin/env python3

def tambah_bil(a,b):
    return a + b;

#manual testing 1 - hasilnya 5
print(tambah_bil(2,3));
```

```python
#!/usr/bin/env python3

def tambah_bil(a,b):
    return a + b;

#manual testing 1 - hasilnya 0
print(tambah_bil(2,-2));
```

```python
#!/usr/bin/env python3

def tambah_bil(a,b):
    return a + b;

#manual testing 1 - hasilnya TypeError
print(tambah_bil("2",3));
```

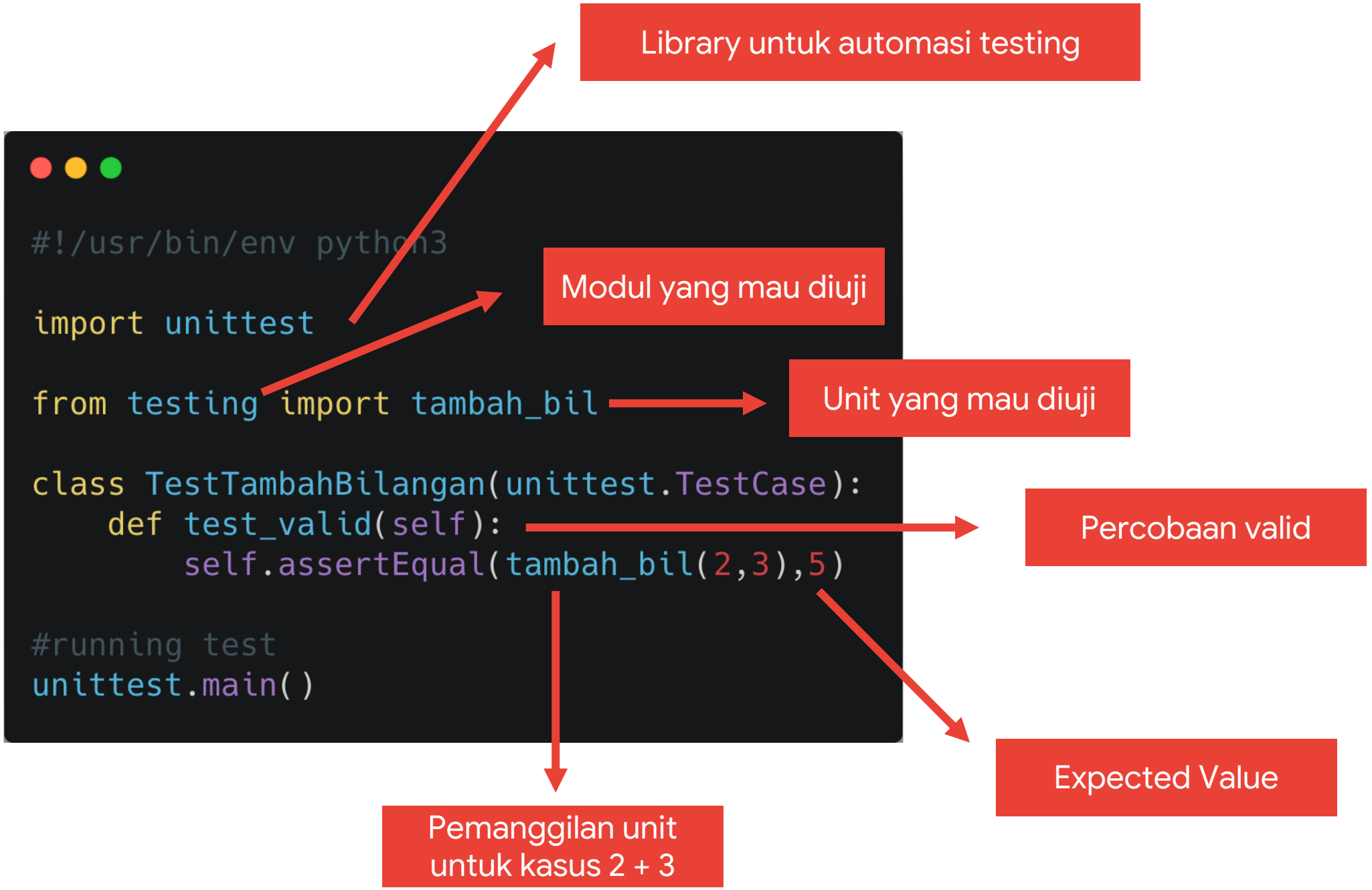
AUTOMATED UNIT TESTING

```python
#!/usr/bin/env python3

def tambah_bil(a,b):
    return a + b;
```

```python
#!/usr/bin/env python3

import unittest

from testing import tambah_bil

class TestTambahBilangan(unittest.TestCase):
    def test_valid(self):
        self.assertEqual(tambah_bil(2,3),5)

#running test
unittest.main()
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

/usr/local/bin/python3 /Users/adambachtiar/Downloads/testing_test.py
❯ /usr/local/bin/python3 /Users/adambachtiar/Downloads/testing_test.py
❯ /usr/local/bin/python3 /Users/adambachtiar/Downloads/testing_test.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

OK, karena expected value sama dengan hasil return value dari tambah_bil(2,3)


Library untuk automasi testing
#!/usr/bin/env python3
import unittest
from testing import tambah_bil
class TestTambahBilangan(unittest.TestCase):
def test_valid(self):
self.assertEqual(tambah_bil(2,3),5)
#running test
unittest.main()
Modul yang mau diuji
Unit yang mau diuji
Percobaan valid
Expected Value
Pemanggilan unit
untuk kasus 2 + 3

“Pengujian harus dilakukan menggunakan test case yang **valid** maupun **tidak valid**. Jumlah test untuk test case tidak valid berjumlah **lebih banyak** dibanding jumlah test untuk test case valid.”

validasi_username.py

```python
#!/usr/bin/env python3

def validate_username(username, minlen):
    assert type(username) ==  str, "username harus berupa tipe data string"

    if minlen < 1:
        raise ValueError("Panjang minimum username minimum 1")

    if len(username) < minlen:
        return False

    if not username.isalnum():
        return False

    return True
```

validasi_username_test.py

```python
#!/usr/bin/env python3

import unittest

from validasi_username import validate_username

class TestValidateUser(unittest.TestCase):
    def test_valid(self):
        self.assertEqual(validate_username("budisetiawan",3), True)

    def test_too_short(self):
        self.assertEqual(validate_username("Ani",5), False)

    def test_invalid_characters(self):
        self.assertEqual(validate_username("Ridwan@Setiawan",9), False)

    def test_invalid_minlen(self):
        self.assertRaises(ValueError, validate_username, "kikipriyana", 0)

unittest.main()
```

Uji dengan Parameter yang Valid

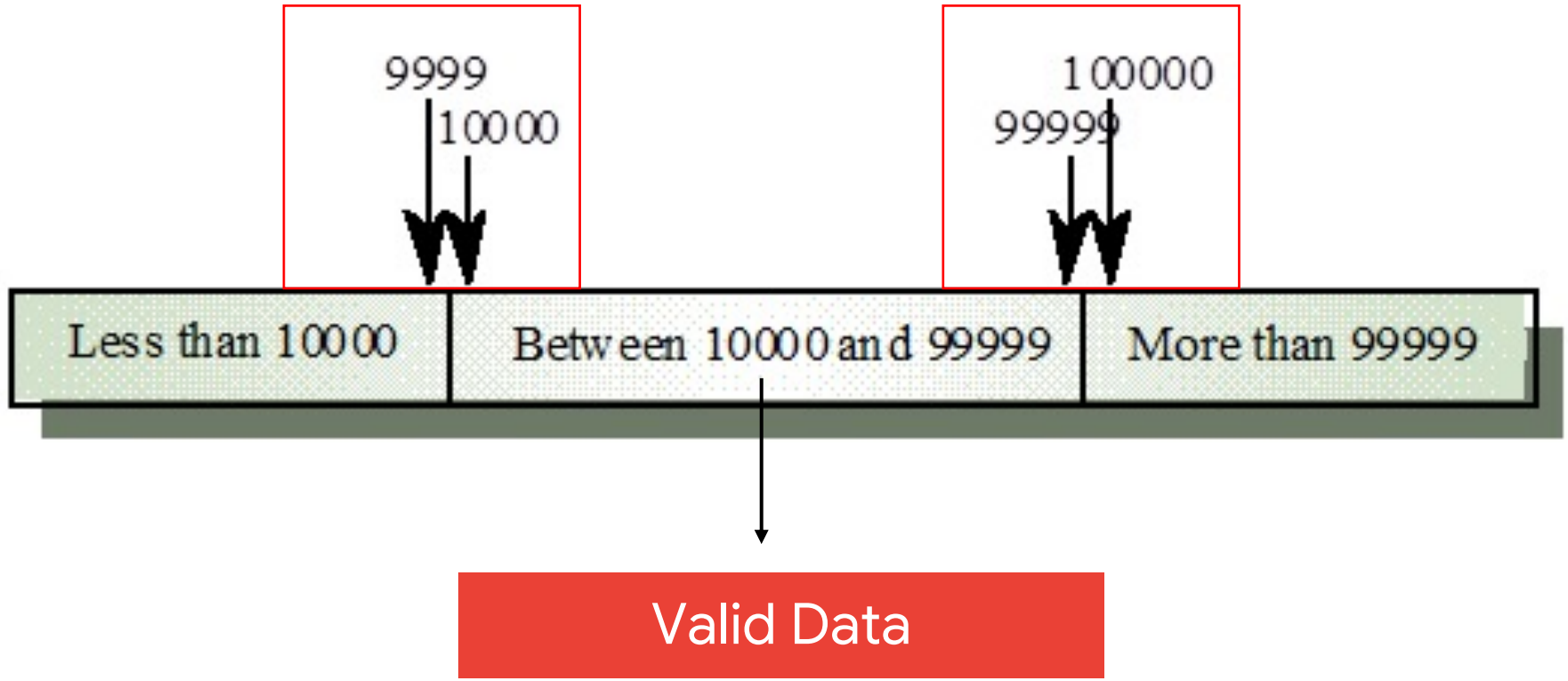
Uji dengan Parameter yang Tidak Valid

“Memilih parameter uji harus dilakukan dengan jeli agar proses pengujian makin baik hasilnya. Diharapkan tester mampu menemukan **Edge Cases** dalam memilih test case.”

EDGE CASES

Edge cases are inputs to our code that produce unexpected results, and **are found at the extreme ends of the ranges of input** we imagine our programs will typically work with.

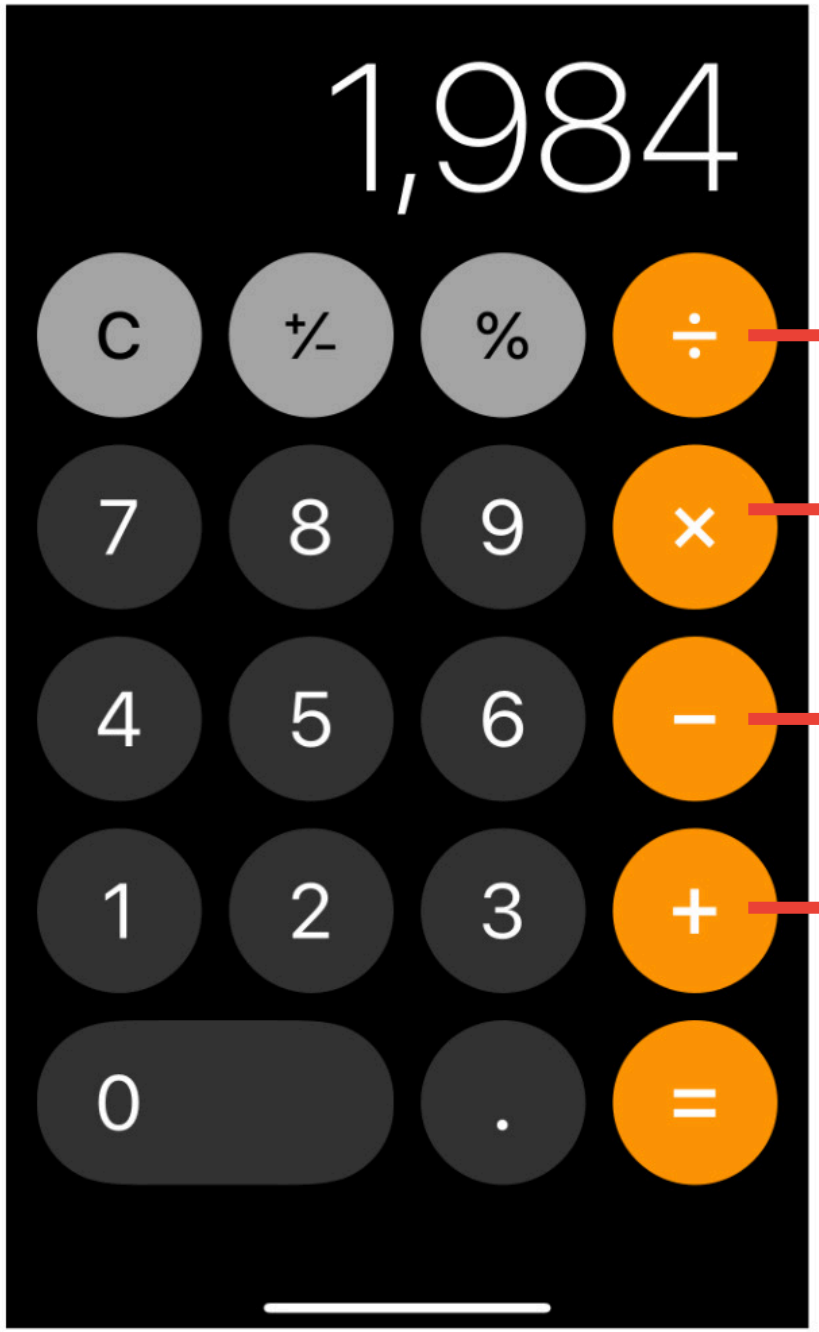
Ilustrasi Edge Cases/Limit Testing



Integration Testing

INTEGRATION TESTING

Pengujian terhadap **koleksi unit** yang bekerja atau berinteraksi dalam sebuah fungsional.



bagi_bil()

kali_bil()

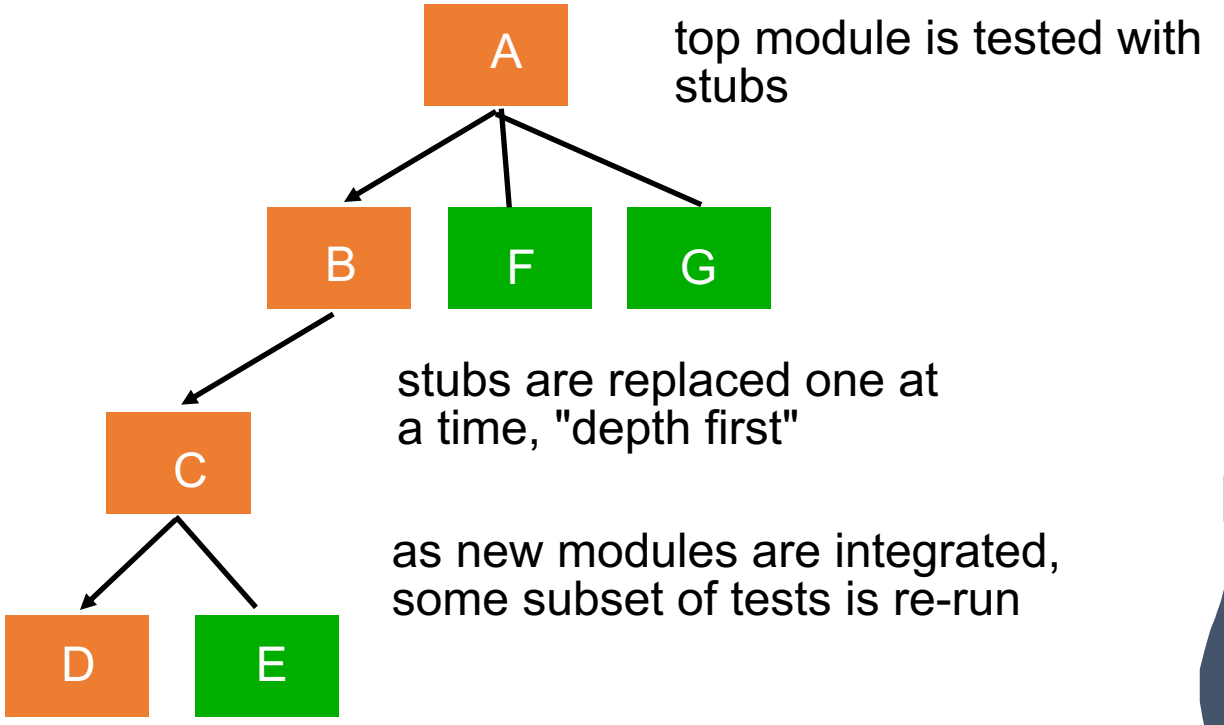
kurang_bil()

tambah_bil()

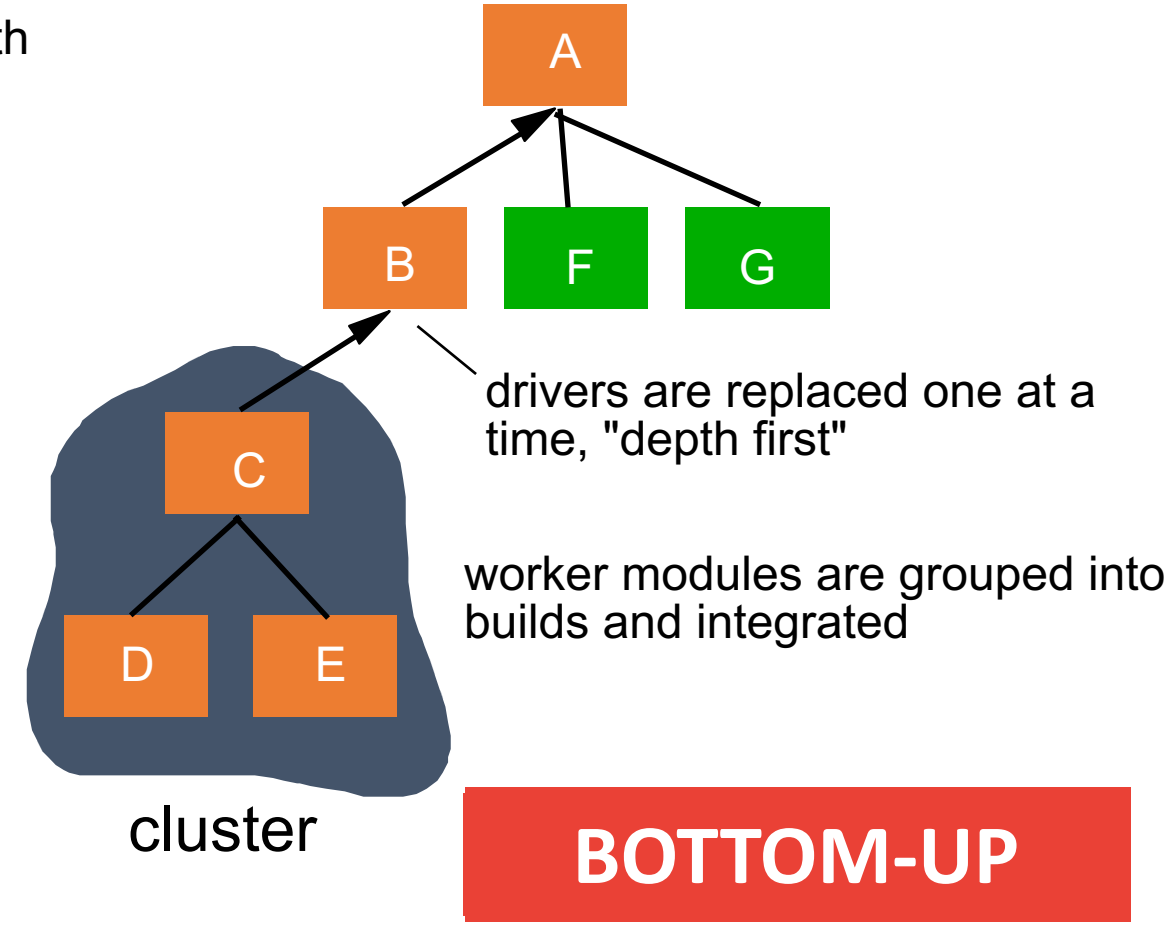
ILUSTRASI INTEGRATION TESTING

Hitung 3 x 2 + 1 / 2 - 10 =

Pendekatan Integration Testing



TOP-DOWN



BOTTOM-UP

METODE TESTING

Metode Testing

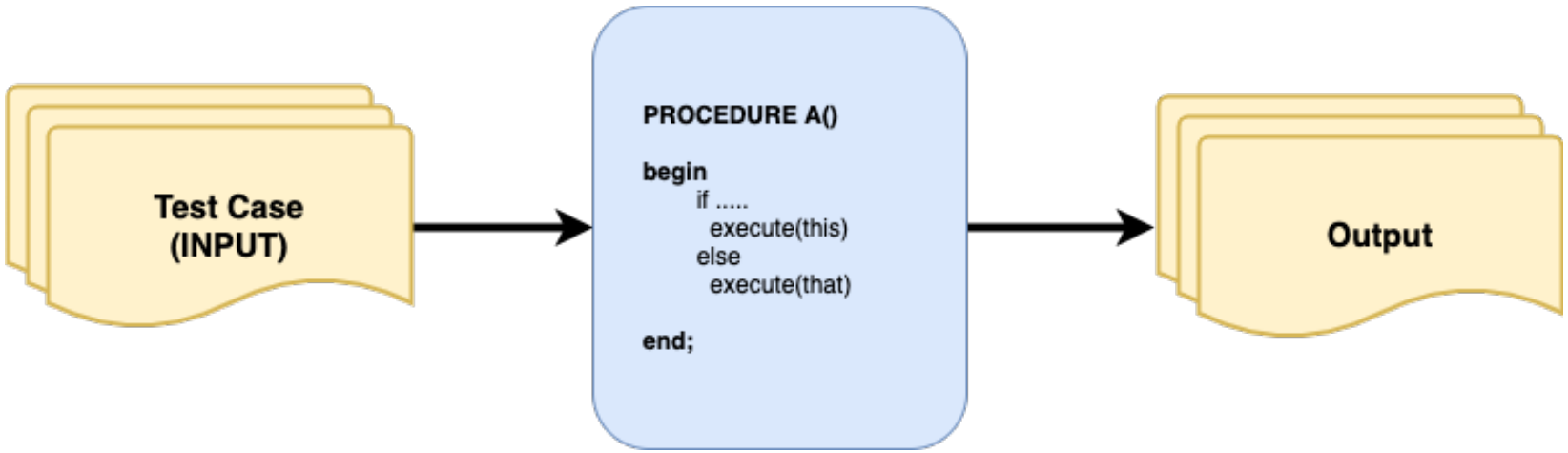
White Box

Sering juga disebut **clear-box** atau **transparent testing** dimana pengujian ini bergantung pada pengetahuan dari tester tentang **bagaimana suatu kode program bekerja** dalam sebuah program dan juga melihat apakah kode program tersebut menghasilkan keluaran sesuai ekspektasi atau tidak.

Black Box

Pengujian dimana tester membentuk test case berdasarkan **spesifikasi kebutuhan** (untuk melihat ekspektasi output dari suatu fungsi) tanpa mengetahui kode programnya.

Ilustrasi White Box Testing

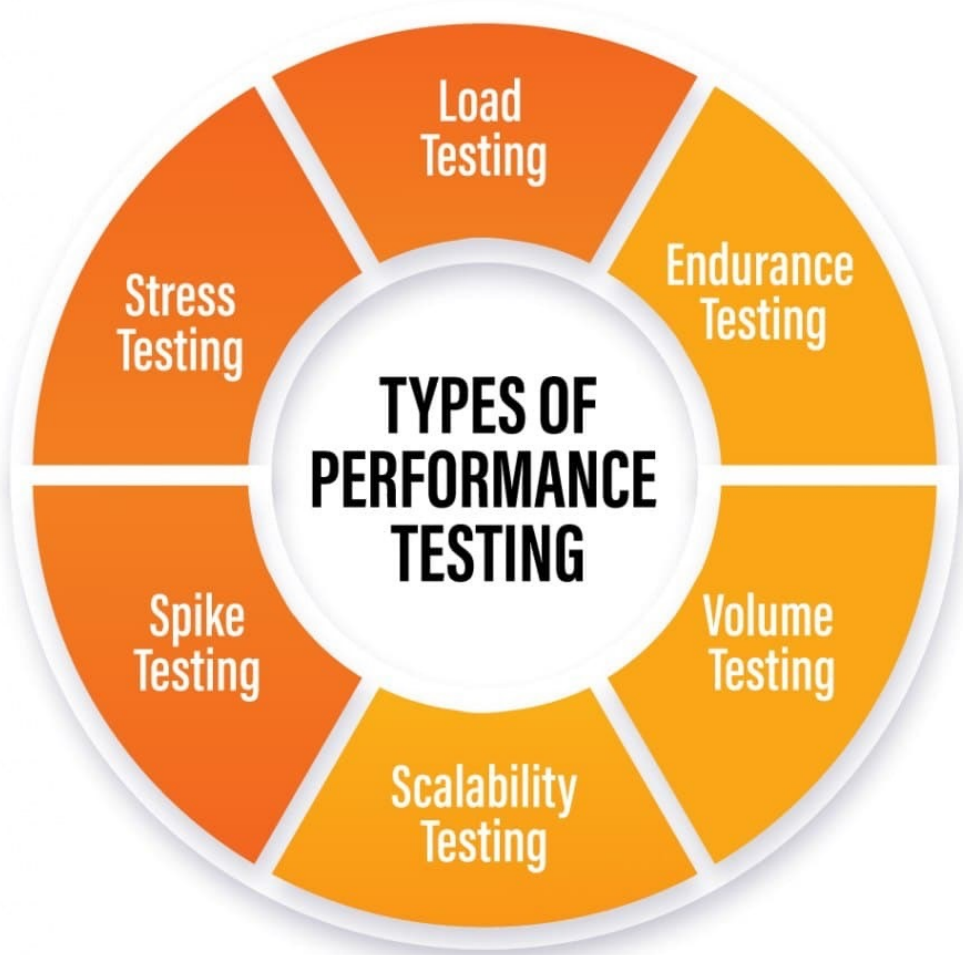


Ilustrasi Black Box Testing

Catatan: Selain digunakan untuk menguji kebutuhan fungsional, Black box testing bisa digunakan untuk menguji kebutuhan non-fungsional (misal: performance testing)

Performance Testing

Salah satu bentuk pengujian non-fungsional dan berkaitan dengan Faktor Kualitas Perangkat Lunak



“Baik black box maupun white box, kedua metode tersebut akan membandingkan **hasil output dari perangkat lunak** terhadap **ekpektasi output** (baik secara kode program atau spesifikasi kebutuhan)”

Dokumentasi Hasil Pengujian

Diterima jika hasil yang diharapkan sama dengan hasil sesuai uji kasus begitu pun sebaliknya

No.	Kasus Uji	Hasil yang Diharapkan	Hasil Sesuai Uji Kasus	Keterangan
				[ ] Diterima [ ] Ditolak

Jumlah kasus uji tergantung metode

Kasus uji/Test Cases sesuaikan dengan metode.

Contoh Hasil Pengujian White Box

Kasus mengacu kepada kode program validasi_username.py (lihat slide halaman 17 dan 18)

No.	Kasus Uji	Hasil yang Diharapkan	Hasil Sesuai Uji Kasus	Keterangan
1	Valid test: Username = “budisetiawan” Minlen = 3	Fungsi untuk test case ini menghasilkan nilai True	Hasil uji menghasilkan True	[X] Diterima [ ] Ditolak
2	Invalid Test: Username = “Ani” Minlen = 5	Fungsi untuk test case ini menghasilkan nilai True karena username tidak memenuhi panjang minlen	Hasil uji menghasilkan True	[X] Diterima [ ] Ditolak
..	..	..	..	..
4	Invalid Test: Username = “kikipriyana” Minlen = 0	Fungsi untuk test case ini harus menangkap error berupa ValueError karena minlen < 1	Hasil uji menangkap error berupa Value Error dengan pesan error “Panjang minimum username minimum 1”	[X] Diterima [ ] Ditolak

Kesimpulan pengujian: n buah test case lolos uji

Bukti Hasil Pengujian White Box

“Jumlah kasus uji yang harus disiapkan pada metode white box dihitung menggunakan **Cyclomatic Complexity**.”

Catatan: Metode perhitungan Cyclomatic Complexity akan dibahas pada mata kuliah “**Kualitas dan Pengujian perangkat lunak**”.

Contoh Hasil Pengujian Black Box

Jika username valid maka halaman isi password akan muncul dan apabila tidak valid maka akan muncul pesan kesalahan sesuai kesalahannya

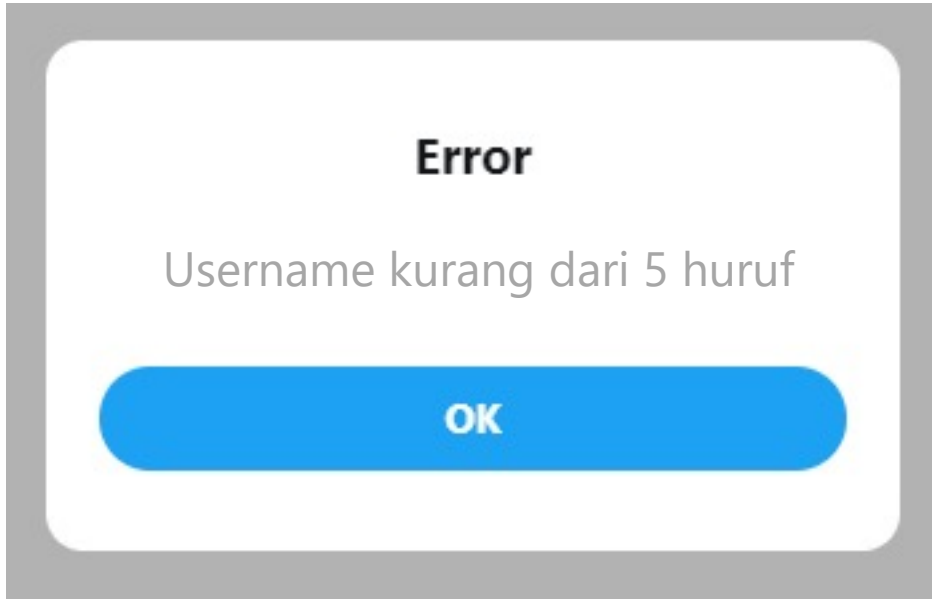
Contoh Hasil Pengujian Black Box

Kasus mengacu kepada gambar pada slide halaman 35

No.	Kasus Uji	Hasil yang Diharapkan	Hasil Sesuai Uji Kasus	Keterangan
1	Valid test: Username = “budisetiawan”	Setelah mengisi username maka akan muncul halaman isi password	Halaman isi password muncul	[X] Diterima [ ] Ditolak
2	Invalid Test: Username = “Ani”	Setelah klik tombol Log In maka akan muncul pesan kesalahan bahwa username kurang dari 5 huruf	Muncul pesan kesalahan “Username kurang dari 5 huruf”	[X] Diterima [ ] Ditolak
..	..	..	..	..
4	..	..	..	..

Kesimpulan pengujian: n buah kasus uji lolos uji

Bukti Hasil Pengujian Black Box



“Dalam suatu pengujian unit, unit bisa diuji menggunakan metode white box atau black box (**cukup salah satunya**). Bisa seragam atau hybrid.”