

LAPORAN PRAKTIKUM SOFTWARE QUALITY ASSURANCE

PERTEMUAN KE 3



Disusun Oleh :

NAMA : MARFIANA AYU IRAWATI
NIM : 235611053
JURUSAN : SISTEM INFORMASI
JENJANG : S1

Universitas Teknologi Digital Indonesia

2024

PERTEMUAN 3

PENGUJIAN DOMAIN (DOMAIN TESTING)

I. PENDAHULUAN DAN DASAR TEORI

Pengujian domain adalah salah satu aspek penting dalam pengembangan perangkat lunak dan sistem informasi. Pengujian ini berfokus pada memastikan bahwa sistem memenuhi kebutuhan dan persyaratan yang ditentukan dalam domain tertentu. Domain di sini merujuk pada area spesifik di mana aplikasi atau sistem akan beroperasi, seperti kesehatan, pendidikan, finansial, atau manufaktur. Pengujian domain membantu mengidentifikasi kesalahan dan memastikan bahwa sistem berfungsi dengan baik dalam konteks yang relevan.

Dalam dunia yang semakin kompleks ini, pengujian domain tidak hanya menjadi suatu keharusan, tetapi juga dapat meningkatkan kepercayaan pengguna dan pemangku kepentingan terhadap produk yang dihasilkan. Dengan pengujian yang tepat, risiko kesalahan yang dapat mengakibatkan kerugian finansial atau reputasi dapat diminimalisir.

1. Definisi Pengujian Domain :

Pengujian domain merupakan proses evaluasi sistem untuk memastikan bahwa aplikasi memenuhi persyaratan yang ditentukan dalam konteks domain tertentu. Ini melibatkan penggunaan pengetahuan khusus tentang domain untuk merancang dan melaksanakan pengujian yang lebih efektif.

2. Tujuan Pengujian Domain :

- Validasi Fungsi : Memastikan bahwa semua fungsi yang diharapkan berjalan dengan baik.
- Identifikasi Kesalahan : Mendeteksi bug dan masalah sebelum aplikasi diluncurkan.
- Kepatuhan terhadap Standar : Memastikan bahwa sistem mematuhi regulasi dan standar industri yang berlaku.

3. Metodologi Pengujian :

- Pengujian Manual : Melibatkan pengujian yang dilakukan oleh penguji manusia dengan menggunakan skenario dan data dunia nyata.
- Pengujian Otomatis : Menggunakan skrip dan alat untuk menguji fungsi dan performa aplikasi secara otomatis, yang seringkali lebih efisien untuk pengujian regresi.

4. Teknik Pengujian Domain :

- Pengujian Berbasis Kasus Penggunaan : Memfokuskan pada bagaimana pengguna sebenarnya akan menggunakan sistem dalam konteks domain tertentu.
- Pengujian Berbasis Risiko : Mengidentifikasi area dengan risiko tinggi dan mengalokasikan lebih banyak sumber daya untuk pengujian bagian tersebut.
- Analisis Data : Menggunakan data historis dan analisis statistik untuk memandu pengujian dan pengambilan keputusan.

5. Peran Domain Expert :

Ahli domain memiliki pengetahuan mendalam tentang area spesifik di mana aplikasi beroperasi. Keterlibatan mereka dalam proses pengujian sangat penting untuk memastikan bahwa semua aspek kritis telah diperiksa dan dipahami dengan baik.

6. Tantangan dalam Pengujian Domain :

- Kompleksitas Sistem : Sistem yang lebih kompleks dapat membuat pengujian menjadi lebih sulit.
- Variasi dalam Kebutuhan : Setiap domain memiliki kebutuhan yang unik, sehingga pendekatan pengujian harus disesuaikan.
- Evolusi Teknologi : Perkembangan teknologi yang cepat menuntut pengujian untuk terus beradaptasi dan memperbarui metodologinya.

II. PEMBAHASAN

Praktik :

Berikut, akan dijelaskan tentang prosedur untuk mengidentifikasi domain. Diketahui suatu fungsi bahasa C pada Gambar-3.2 yang menggambarkan suatu prosedur untuk mengidentifikasi domain.

Fungsi menerima dua input X dan Y, dan nilai pengembalian (*return*) integer. Gambaran CFG dari *codedomain()* ditunjukkan pada gambar 3.3. Dua predikat dalam dua perintah *if()* yang ditunjukkan dengan node 3 dan 6 pada gambar 3.3.

Predikat:

```
P1 : c > 5
int codedomain(int x, int y){
    int c, d, k
    c = x + y;
    if (c > 5) d = c - x/2;
    else d = c + x/2;
    if (d >= c + 2) k = x + d/2;
    else k = y + d/4;
    return(k);
}
```

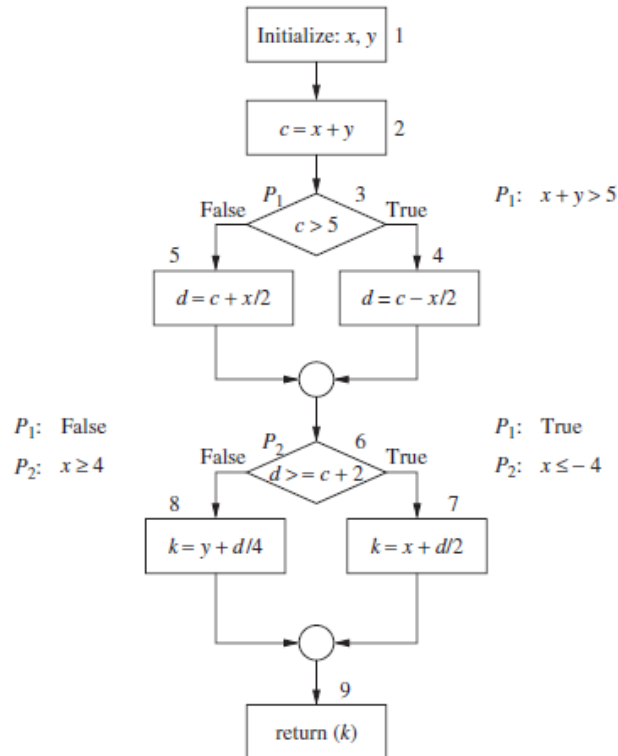
Gambar-3.2 Fungsi untuk menjelaskan domain program.

Interpretasi Kode Program

Pada perintah `if()` pertama hanya mempunyai satu interpretasi, yaitu:

$$P_1 : x + y > 5$$

Karena kendali program mencapai perintah `if()` hanya melalui satu *path* dari *node* awal.



Gambar-3.3 CFG Representasi dari Fungsi Gambar 3.2

$$P_2: d \geq c + 2$$

Pada perintah `if()` kedua terdapat dua interpretasi, karena kendali program dapat mencapai perintah `if()` kedua melalui 2 *path*: (a) bila evaluasi `if()` pertama adalah *true* dan (b) bila evaluasi `if()` pertama adalah *false*. Dua interpretasi ini diringkas dalam tabel 3.1.

Tabel-3.1 Dua Interpretasi dari pernyataan `if()` kedua gambar-3.2

Evaluasi dari P1	Interpretasi dari P2
True	$X \leq -4$
False	$X \geq 4$

Dijelaskan prosedur untuk memperoleh domain dari interpretasi P1 dan P2 (Gambar-3.3).

Analisis Domain Program

Terdapat grid dua dimensi yang dinyatakan dengan X dan Y pada gambar 3.4. Ukuran grid terlalu besar untuk menampilkan seluruh domain dari program (gambar-3.3). Predikat P₁ dibagi kedalam 2 wilayah. Batas P₁ ditunjukkan dengan garis lurus yang dinyatakan dengan persamaan $x + y = 5$.

Selanjutnya, terdapat dua interpretasi dari predikat P₂, yaitu untuk P₁ = *True*, maka P₂ mempunyai interpretasi berikut:

$$P_2 : x \leq -4$$

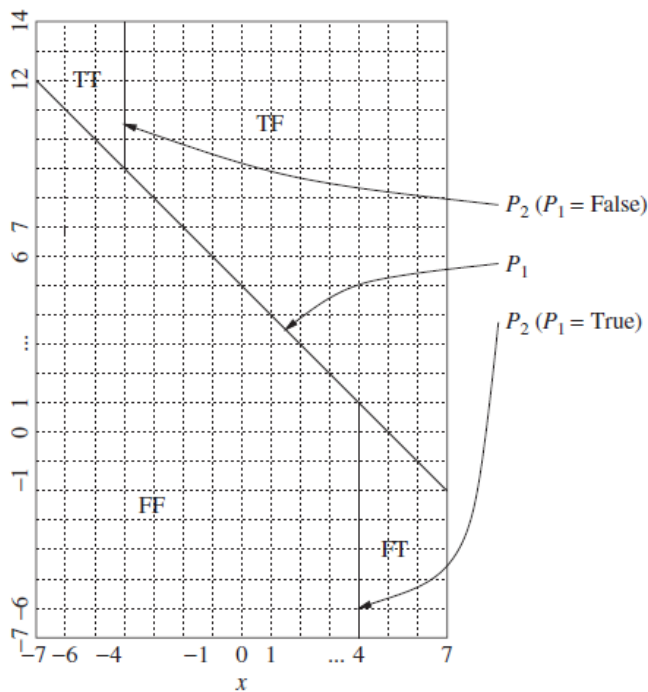
Oleh karena itu, selanjutnya P₂ dibagi kedalam area (kumpulan dari titik-titik), yang didefinisikan

dengan $P_1 = \text{True}$ kedalam dua kumpulan yang sesuai dengan dua nilai kebenaran. Untuk batas P_2 , bila P_1 dievaluasi dengan True , dinyatakan dengan garis lurus $x = -4$. Pada area sisi kiri batas P_2 dan diatas batas P_1 yang sesuai dengan $P_1P_2 = \text{TT}$, dan area di kanan batas P_2 dan diatas batas P_1 sesuai dengan $P_1P_2 = \text{TF}$.

Untuk $P_1 = \text{False}$, maka P_2 mempunyai interpretasi berikut ini:

$P_2: x > 4$

Dengan kata lain, P_2 dibagi kedalam area (kumpulan titik-titik), yang didefinisikan dengan $P_1 = \text{False}$ kedalam dua kumpulan yang sesuai dengan dua nilai kebenaran. Batas P_2 , bila P_1 dievaluasi dengan False , dinyatakan dengan garis lurus $x = 4$. Maka daerah disisi kanan batas dari P_2 dan dibawah batas sesuai dengan $P_1P_2 = \text{FT}$, dan area disisi kiri dari P_2 dan batas P_1 sesuai dengan $P_1P_2 = \text{FF}$ dalam gambar-3.4.



Jenis dari Domain Error

- Domain adalah sekumpulan dari nilai pada program yang melakukan perhitungan identic
- Domain dapat dinyatakan dengan sekumpulan dari predikat. Setiap elemen dari domain sesuai dengan predikat dari domain.

Contoh: domain TT dalam gambar-3.4 dinyatakan secara matematis dengan kumpulan predikat yang ditunjukkan gambar-3.5.

Domain didefinisikan, dari suatu pandangan geometri, dengan sekumpulan dari kendala yang disebut ketidak samaan batas. Property dari domain dijelaskan sebagai berikut.

P1:	$x + y > 5$	$\equiv \text{True}$
P2:	$x \leq -4$	$\equiv \text{True}$

Soal Latihan :

Berikut ini diberikan kode program binary search. Input array `V[]` diasumsikan disimpan urut naik, `n` adalah ukuran array, dan anda akan mencari indeks dari elemen `X` dalam array. Jika `X` tidak ada dalam array, rutin akan memberikan nilai kembalian (return -1).

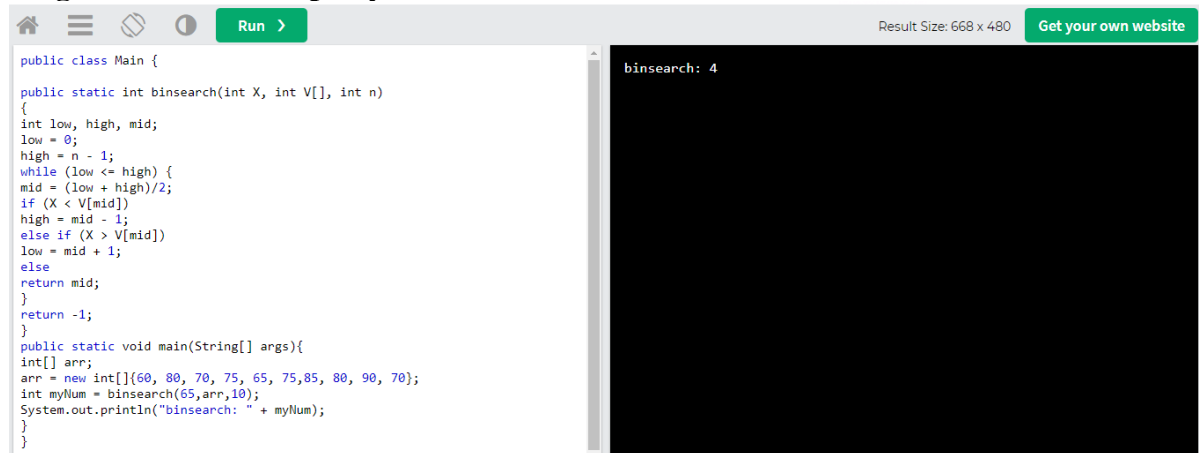
```
int binsearch(int X, int V[], int n)
{
    int low, high, mid;
    low = 0;
    high = n - 1;
    while (low <= high) {
        mid = (low + high)/2;
        if (X < V[mid])
            high = mid - 1;
        else if (X > V[mid])
            low = mid + 1;
        else
            return mid;
    }
    return -1;
}
```

Dari kode program fungsi `binSearch` tersebut selesaikan berikut ini.

1. Buatlah CFG dari `binSearch`?
2. Dari CFG tersebut, tentukan jalur entry dan exit yang memenuhi kriteria cakupan tersebut?
3. Buatlah jalur independen (independent paths) dari CFG tersebut?
4. Ujilah menggunakan kasus uji, sehingga seluruh jalur terlewati? Buatlah deskripsi pengujian dari kasus tersebut?

Penyelesaian Dan Pembahasan Latihan :

Program Dan Hasil Outputnya :

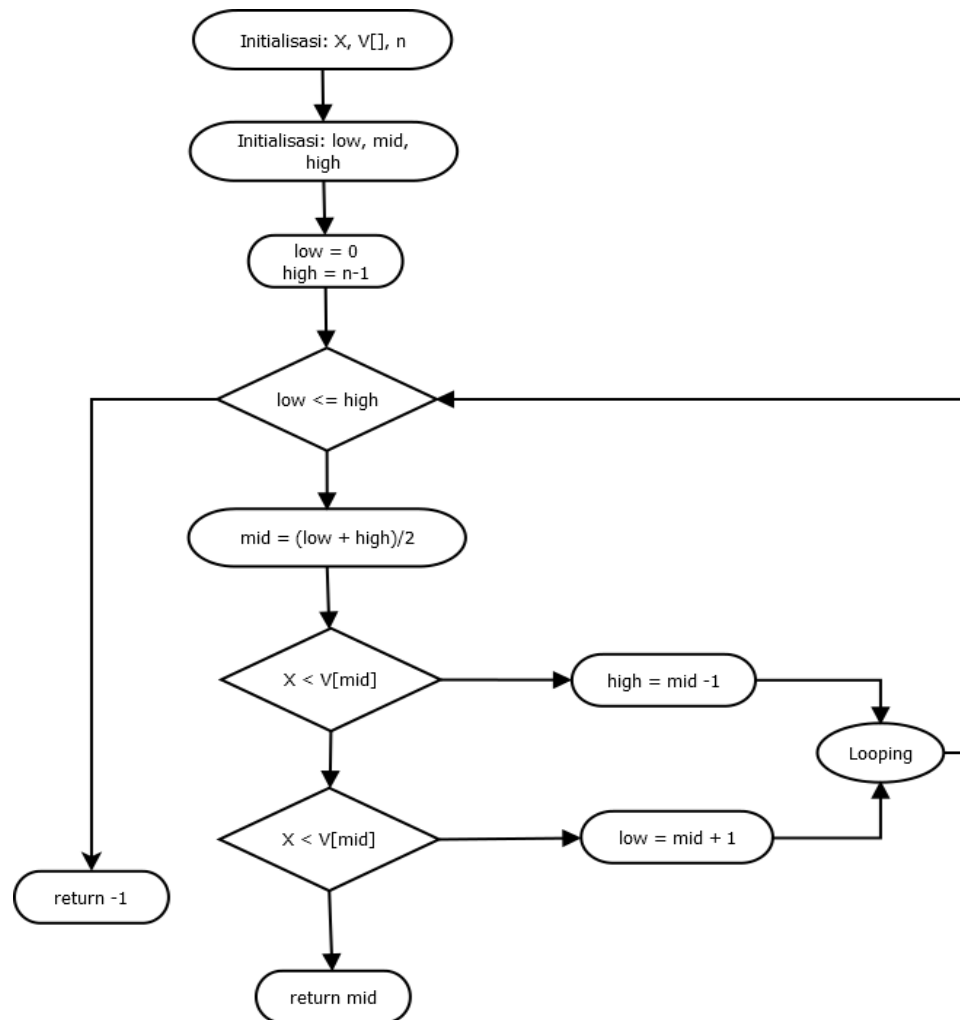


The screenshot shows a Java IDE with a code editor on the left and a console output on the right. The code implements a binary search function and a main method that tests it with a specific array and value.

```
public class Main {
    public static int binsearch(int X, int V[], int n)
    {
        int low, high, mid;
        low = 0;
        high = n - 1;
        while (low <= high) {
            mid = (low + high)/2;
            if (X < V[mid])
                high = mid - 1;
            else if (X > V[mid])
                low = mid + 1;
            else
                return mid;
        }
        return -1;
    }
    public static void main(String[] args){
        int[] arr;
        arr = new int[]{60, 80, 70, 75, 65, 75, 85, 80, 90, 70};
        int myNum = binsearch(65, arr, 10);
        System.out.println("binsearch: " + myNum);
    }
}
```

The console output shows the result of the binary search: `binsearch: 4`.

Flowchart :



path

1. 1-2-3-14(F)-15
2. 1-2-3-4(T)-5-6(T)-7-8
3. 1-2-3-4(T)-5-9(F)-10(T)-11-8
4. 1-2-3-4(T)-5-9(F)-12(F)-13

Untuk unit program sederhana sejumlah path dapat dilewati dan diuji menggunakan kasus uji

tertentu sehingga seluruh path dapat terlewati.

Di sisi lain, untuk unit program dengan sejumlah besar path (jalur), melaksanakan setiap jalur

yang berbeda mungkin tidak praktis. Oleh karena itu, programmer lebih produktif untuk memilih

sejumlah kecil jalur program dalam upaya mengungkapkan cacat dalam kode.

Soal Tugas :

Buatlah program konversi suhu dari Celcius ke Fahrenheit dan Reamur, Kelvin kemudian buatlah CFG dan cek menggunakan 5 buah data uji , serta buatlah analisisnya.

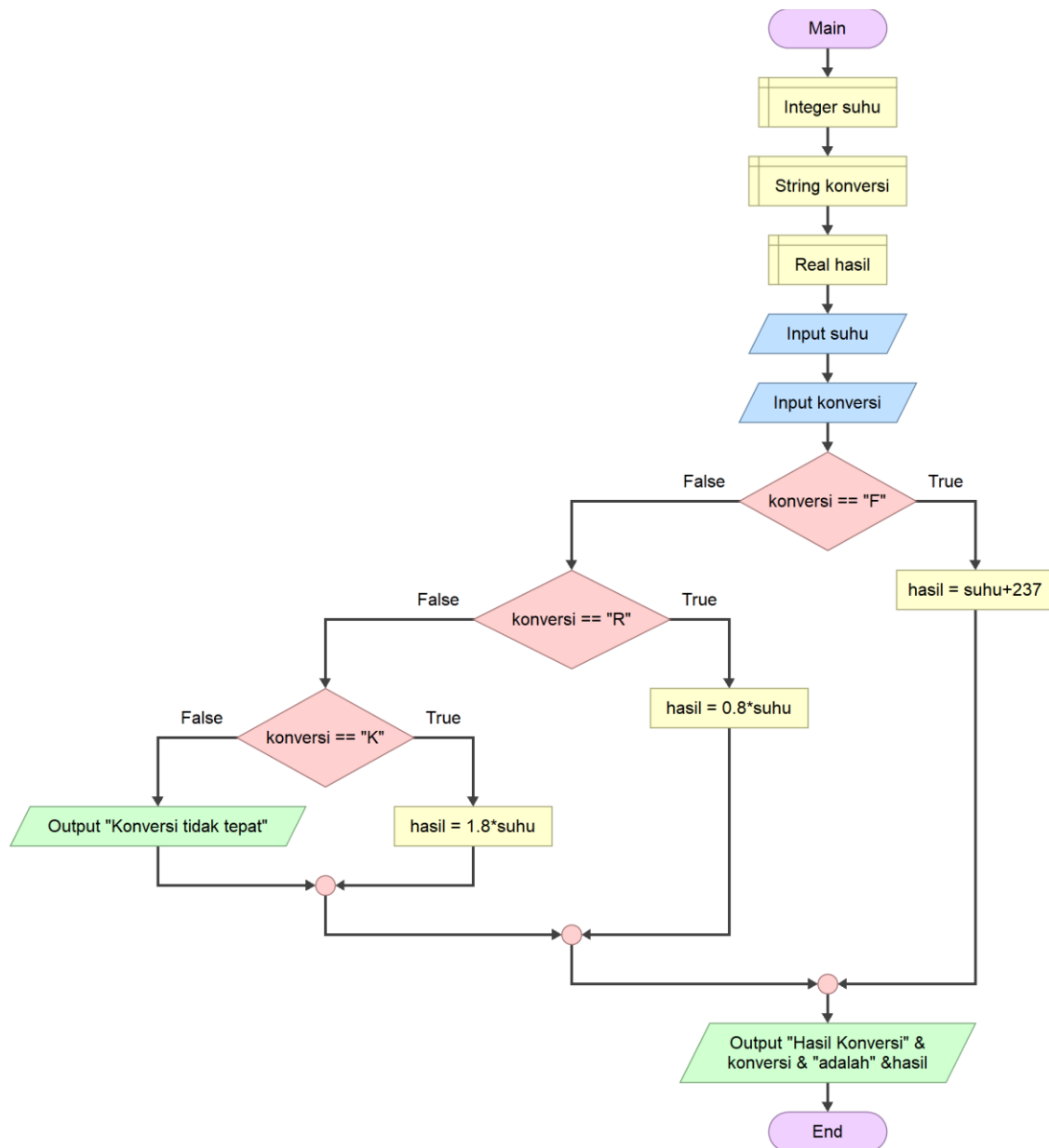
Dimana nilai konversinya adalah sebagai berikut :

$$F = C + 237$$

$$R = 0,8 \times C$$

$$K = 1,8 \times C$$

Pembahasan Dan Penyelesaian Tugas :



Path

1. 1-2-3-4-5-6(T)-7-16

2. 1-2-3-4-5-8(F)-9(T)-10-16
3. 1-2-3-4-5-8(F)-11(F)-12(T)-13-16
4. 1-2-3-4-5-8(F)-11(F)-14(F)-15-16

Hasil uji coba

1. Suhu celsius ke Reamur (R)

56

R

Hasil Konversi R adalah 44.8

2. Suhu celsius ke Fahrenheit (F)

70

F

Hasil Konversi F adalah 307

3. Suhu Celsius ke Kelvin (K)

34

K

Hasil Konversi K adalah 61.2

4. Suhu Celsius ke Kelvin (K)

30

K

Hasil Konversi K adalah 54

5. Suhu celsius ke Fahrenheit (F)

98

F

Hasil Konversi F adalah 335

III. KESIMPULAN

Pengujian domain adalah proses krusial dalam pengembangan perangkat lunak yang bertujuan untuk memastikan bahwa sistem berfungsi dengan baik dalam konteks spesifik suatu industri atau bidang. Melalui pendekatan ini, berbagai aspek penting dapat diringkas sebagai berikut:

1. Validasi Kesesuaian : Pengujian domain membantu memastikan bahwa aplikasi memenuhi persyaratan pengguna dan beroperasi sesuai dengan standar yang ditetapkan dalam domain tertentu.
2. Pentingnya Pengetahuan Domain : Keterlibatan ahli domain memberikan wawasan yang diperlukan untuk mendesain pengujian yang relevan dan efektif, memastikan bahwa semua kebutuhan khusus terpenuhi.
3. Metodologi Beragam : Pengujian domain menggunakan berbagai teknik, seperti pengujian berbasis kasus penggunaan dan pengujian berbasis risiko, yang memungkinkan evaluasi yang lebih mendalam dan terfokus.
4. Mengurangi Risiko : Melalui pengujian yang menyeluruh, potensi kesalahan dapat diminimalkan, yang pada gilirannya mengurangi risiko finansial dan reputasi bagi organisasi.
5. Meningkatkan Kualitas Produk : Implementasi pengujian domain yang efektif berkontribusi pada pengembangan produk yang lebih andal dan berkualitas tinggi, serta meningkatkan kepuasan pengguna.

Dengan demikian, pengujian domain bukan hanya langkah tambahan, tetapi merupakan bagian integral dari siklus hidup pengembangan perangkat lunak yang mendukung keberhasilan produk di pasar dan membangun kepercayaan di antara pemangku kepentingan.