-152116025-
TASARIM SÜREÇLERİ

Ders 10: Tasarım Evresi-5

Dr. Yıldıray Anagün,
yanagun@ogu.edu.tr
Eskişehir Osmangazi University
Computer Engineering Department

Tasarım Evresinde Neler Yapılır?

- Kullanıcı-Sistem ara yüzü tasarımı
- Veri tabanı tasarımı
- Gömülü Donanım tasarımı
- Yazılım tasarımı
- Sistem kontrolleri/test tasarımı
- Ağ tasarımı ve güvenliği

Kavramlar

- Hata
  - Beklenen davranışın sergilenmemesi
  - Bu durumun sebebi (Bug)
- Doğrulama
  - Beklenen davranışların sağlandığının gözlenmesi işlemi.

Test

- Etkin test
  - Sistemin detaylı anlaşılması
  - Uygulama alanı ve teknik (çözüm) alanı hakkında bilgi
  - Test tekniklerinin bilinmesi
  - Test teknikleri uygulama kabiliyeti ve tecrübesi
- Testin bağımsız test uzmanları tarafından yapılması
  - Programın çalıştığı tutumunu sergilemek / İç yapıyı bilmenin verdiği bir genişlik (Yazılımcılar)
  - Yazılımcıların, programın çalıştığı veri kümesi üzerinde yoğunlaşmaları
  - Başkalarının, hataları daha kolay görebilmesi (Hata bulma hedefi)

Test süreci aktiviteleri

Test hedeflerinin belirlenmesi

Test durumlarının tasarımı (oluşturulması)

Test durumlarının yazılması

Test durumlarının, test edilmesi

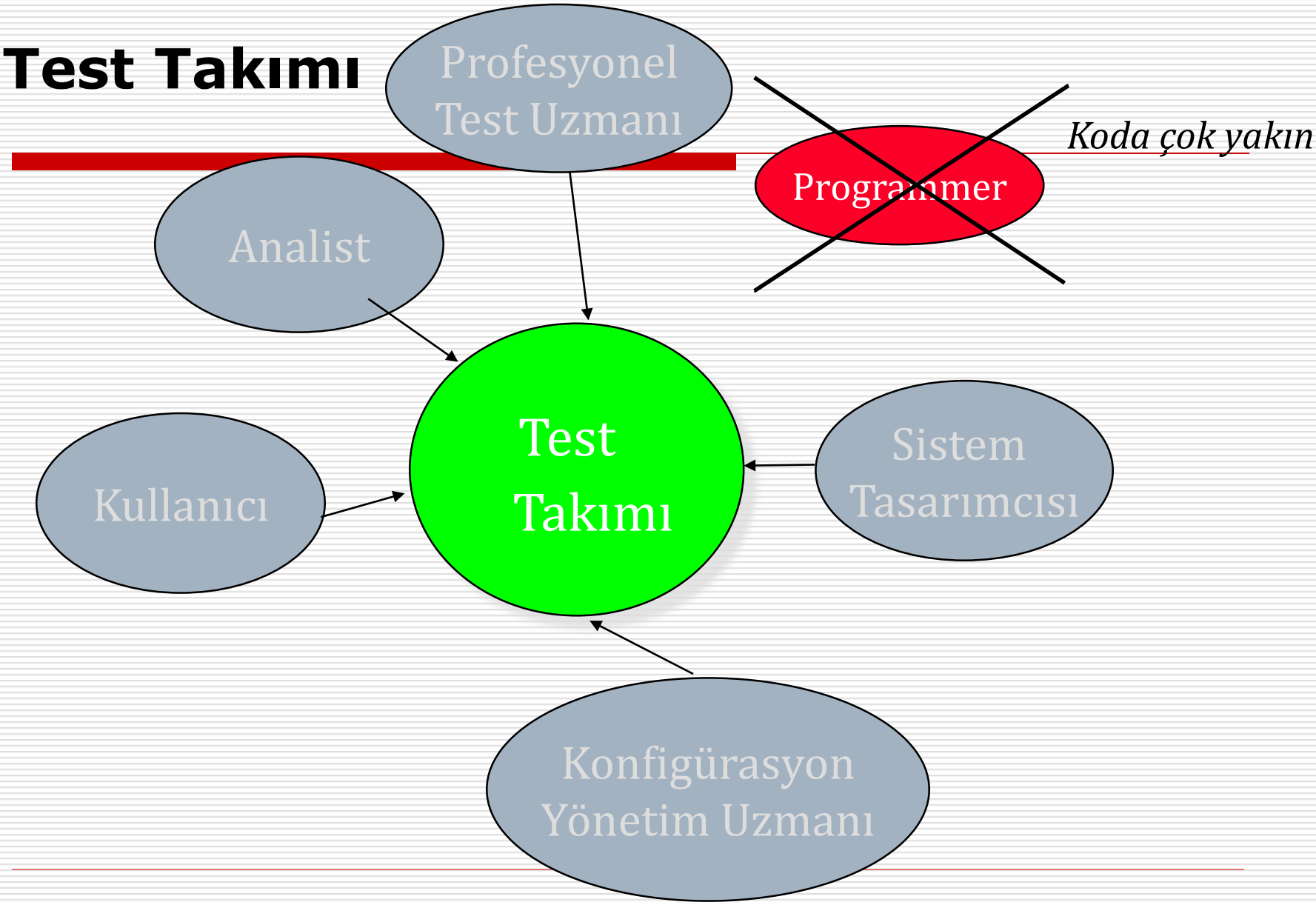
Testlerin çalıştırılması

Test sonuçlarının değerlendirilmesi

Sistem üzerinde değişiklik yapılması

Değişiklikler sebebiyle, yeni hataların olup olmadığının tespit edilmesi (Regression Testing)

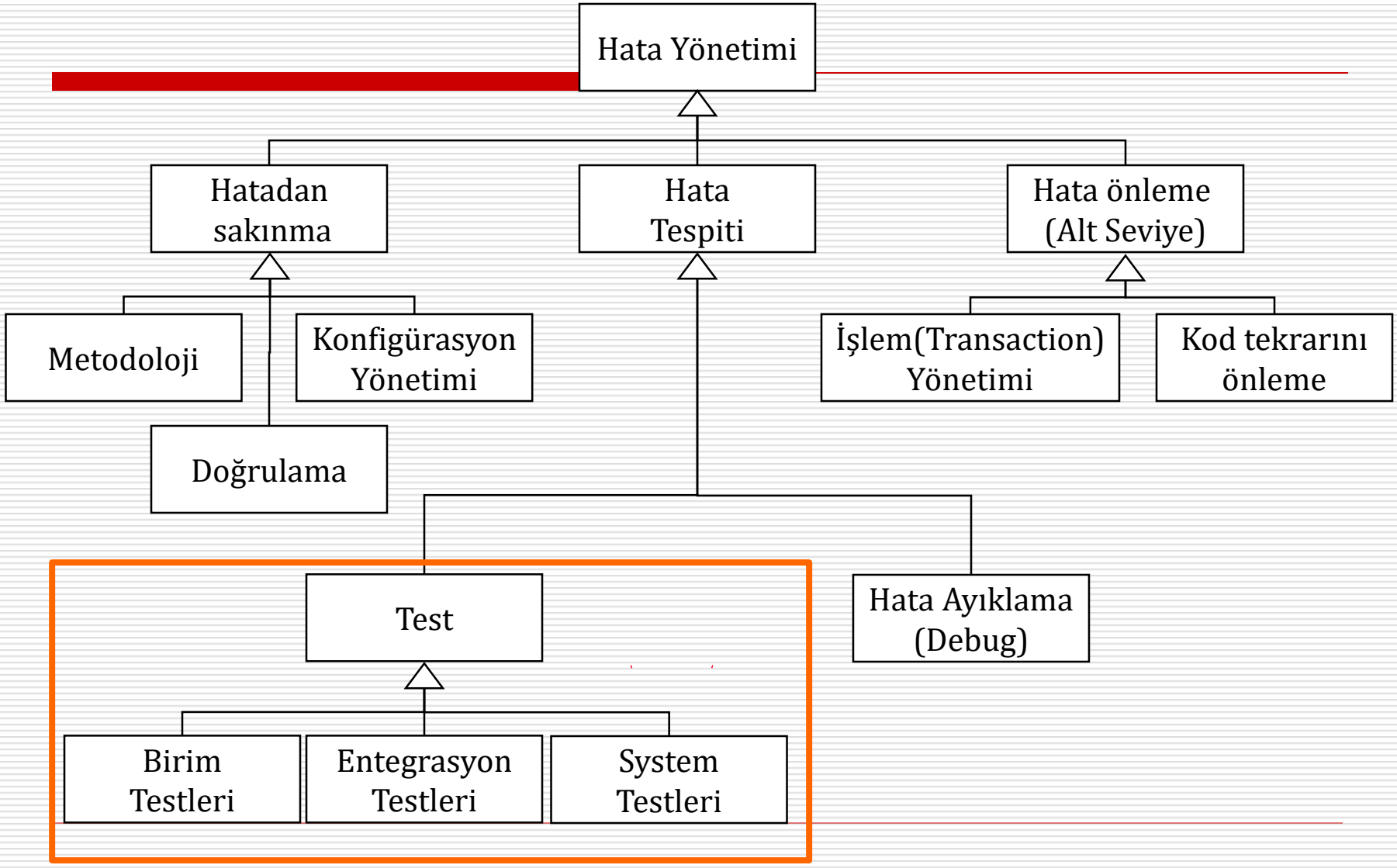
Test Takımı


Profesyonel
Test Uzmanı
Koda çok yakın
Programmer
Analist
Test
Takımı
Kullanıcı
Sistem
Tasarımcısı
Konfigürasyon
Yönetim Uzmanı

Hata Yönetimi

- Hatadan sakınma (Genel Yaklaşım)
  - Karmaşıklığın azaltılması için metodoloji kullanılması
  - Tutarsızlıkların önlenmesi için, konfigürasyon yönetimi kullanılması
  - Algoritmik hataların önlenmesi için, doğrulama yapılması
  - Daha tasarım üzerinde iken, hataları belirlemek için, gözden geçirmelerin yapılması
- Hata tespiti
  - Test : Planlı bir şekilde hataların bulunma aktivitesi
  - Ayıklama (Debugging): Gözlenen bir hatanın sebebinin bulunması ve çıkarılması
  - İzleme (Monitor) : Durum / davranışları izleme ve bilgi yayınlama
- Hata önleme (Alt seviye)
  - İstisna (Exception) Yönetimi
  - İşlem (Transaction) Yönetimi
  - Kod tekrarını önleme

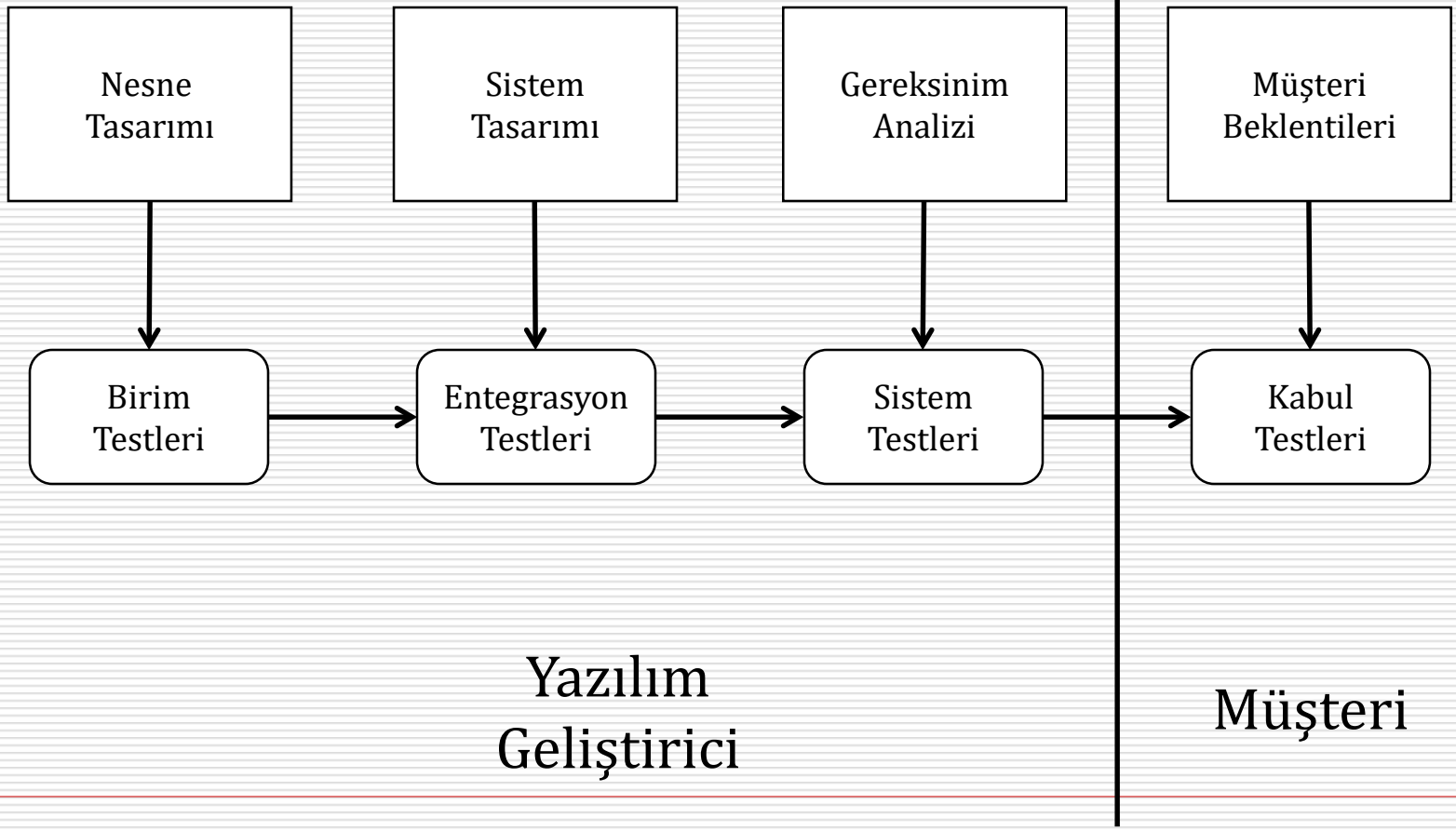
Hata yönetim teknikleri



Test Modeli

- Test modeli, test ile ilgili tüm kararları ve bileşenleri bir paket içinde birleştirir. (Test paketi veya Test gereksinimleri olarak da adlandırılır.)
- Test modeli, testleri, testleri çalıştıran programı, girdi verilerini, beklenen çıktı değerini ve test alt yapısını içerir.
- The test model contains tests, test driver, input data, expected output and the test harness
  - Testi çalıştıran program (Test Driver)
  - Test için girdi değerleri
  - Testten elde edilen değer ile, beklenen değerin karşılaştırılması
  - Test alt yapı bileşeni (Test harness)
    - ✓ Testleri çalıştırıldığı, davranışların ve çıktıların izlendiği alt yapı bileşeni
    - ✓ Otomatik testler için gereklidir.

Test aktiviteleri ve modeller



Test tipleri

- Birim (Unit) Testleri
  - Sınıf (metodlar) ve alt sistemler test edilir.
  - Yazılım geliştiriciler tarafından yapılır.
  - Doğru bir şekilde geliştirildiği ve hedeflenen fonksiyonu gerçekleştirdiği onaylanır.
- Entegrasyon Testleri
  - Alt sistemler ve netice olarak, tüm sistem test edilir.
    - ✓ Tüm yazılım bileşenleri birlikte çalışıyor mu?
  - Yazılım geliştiriciler tarafından yapılır.
  - Alt sistemler arası arayüzler test edilir. (Alt sistem beklenen işi yapıyor mu?)
- Sistem testleri
  - Tüm sistem test edilir.
    - Fonksiyonel test (Fonksiyonel gereksinimler)
    - Performans test (Fonksiyonel olmayan gereksinimler)
  - Yazılım geliştiriciler tarafından yapılır.
  - Sistemin gereksinimleri karşılanıp karşılamadığı belirlenir. (Fonksiyonel ve Fonksiyonel olmayan gereksinimler)
- Kabul Testleri
  - Tüm sistem değerlendirilir.
  - Müşteri tarafından gerçekleştirilir. Genel süreçler, deneme amaçlı çalıştırılır.
  - Sistemin gereksinimleri karşıladığı ve kullanıma hazır olduğu gösterilir.

Fonksiyonel Test

Hedef : Sistem fonksiyonlarının test edilmesi

- Test durumları, gereksinim analizinden (RAD) oluşturulur. “Use case” ler, test durumlarının, temelini teşkil eder. Kullanıcı klavuz dokümanları da, iyi bir kaynaktır.
- Sistem, bir kutu (Black-Box) gibi ele alınır.
- Birim testleri tekrar kullanılabilir. Yeni testlerinde, geliştirilmesi gerekir.

Niçin entegrasyon testleri ?

- Birim (Unit) testleri; birimi, diğerlerinden bağımsız olarak test eder.
- Bazı hatalar, alt sistemlerin entegrasyonundan kaynaklanır.
- Hazır (satın alınan) bileşenler üzerinde, birim testleri yapılamaz.
- Entegrasyon testi olmadan yapılan sistem testi, çok zaman alır.
- Entegrasyon testinde tespit edilmeyen hatalar, sistemin gerçek ortama geçirildikten sonra tespit edildiğinde, çok pahalı olacaktır.

Test Listesi

- Stres Testi
  - Sistem limitlerinin zorlanması
- Büyük hacimli (Volume) verilerin testi
  - Büyük veri karşısında, sistem nasıl davranıyor.
- Konfigürasyon Testi
  - Farklı donanım ve yazılım konfigürasyonlarının test edilmesi
- Uyumluluk testi
  - Geriye yönelik uyumluluk testleri (Var olan sistemler üzerinde)
- Zaman analiz testi
  - Cevap verme süreleri ve foksiyon işlem sürelerinin testi
- Güvenlik testi
  - Güvenlik gereksinimlerinin testi / Güvenlik açıklarının bulunması
- Ortam testi
  - Isıya, Neme ve hareketlere göre testler
- Kalite testi
  - Güvenilirlik (Reliablility), Bakım kolaylığı ve süreklilik (Availability) testleri
- Kurtarma testi
  - Hatalara ve veri kaybına yönelik sistem davranışlarının testi
- İnsan faktörleri testi
  - Son kullanıcılar ile test.

Kabul Testleri

- Testlerin bir çoğu, entegrasyon testlerinden alınır.
- Alpha test
  - Müşteri, sistemi, geliştiricilerin ortamında kullanır.
  - Yazılımcılar, hataların düzeltilmesi için, hazır bir şekildedir. (Kontrollü kullanım)
- Beta test
  - Müşteri ortamında yürütülür. (Yazılım geliştirici, bulunmaz.)
  - Gerçek ortamda, test edilir.

Test işleminin otomatikleştirilmesi

- Test modeli oluşturma yöntemleri
  - El ile (Manually): Test uzmanları; test verisi hazırlar, testleri çalıştırır ve sonuçları izler. Sonuç, test uzmanları tarafından belirlenir.
  - Otomatik olarak : Test verisi ve test durumlarının otomatik olarak üretilmesi. Testler otomatik çalıştırılır ve sonucuna otomatik olarak karar verilir.
- Otomatik test
  - Tüm testler, test alt yapısı içinde (Test Harness) otomatik olarak çalıştırılır.
- Otomatik testin avantajları
  - Yazılım geliştirici için, daha rahat
  - Daha iyi test imkanı (Kapsanan bölümler açısından. Testler ile ne kadar kod kapsandı?)
  - Test süreci maliyetlerinin düşürülmesi
  - Geriye dönük test işlemleri için vazgeçilmez bir imkan. (Var olan sistem üzerinde yapılan değişiklikler, yeni hatalara sebep oldu mu ?) (Regression Testing)

Test Dublörleri

- Test dublörü etkileşimde olan nesnenin yerine koyulur. (Filmlerdeki gibi)
- Etkileşimde olan nesnenin kullanımı pratik değilse, test dublörü kullanılır.
- 4 Çeşit test dublörü (Gerçek etkileşim nesnesinin yerine geçerek, erişim yapan nesnelere, gerçek nesne ile iletişim yapıyormuş izlenimi verirler.)
  - **Taklidi geçici (Dummy) nesnesi :** Parametre olarak verilir. Gerçekten kullanılmazlar. Bu nesneler, genellikle, parametre listesini doldurmak için kullanılır.
  - **Taklidi çalışan (Fake) nesnesi:** Bu nesne, gerçek ortama geçirilemeyecek, fakat çalışan bir halidir. Kısa yoldan geliştirilmiştir. (Gerçek veritabanı yerine, bellekte tutulan veritabanı gibi)
  - **Taklidi şablon (Stub) nesnesi :** Test sürecini kolaylaştırmak için, özel olarak geliştirilmiş, teste cevap verebilecek nesnelerdir. Programlama hedefi doğrultusundan hariç kullanılamazlar. (Doğrulama içermezler. Hızlı çözüm.)
  - **Taklidi davranış (Mock) nesnesi :** Bu nesneler, gerçek nesnenin davranışını taklid ederler. Alabilecekleri isteklere göre, nasıl karşılık vereceklerini bilirler. (Doğrulama içerirler.)

Statik ve Dinamik Analiz

- **Statik Analiz**
  - Kodun incelenmesi
  - Formal olmayan sunumlar
  - Formal olan sunumlar
  - Otomatik araçların kullanımı
    - ✓ Sözdizimsel ve anlamsal hatalar için
    - ✓ Kod standartlarına uyum kontrolü için
- **Dinamik Analiz**
  - Kapalı kutu (Black-box) testi (Girdi/Çıktı davranış testi)
  - Açık kutu (White-box) testi (Alt sistemin veya sınıfın çalışma mantığının testi)
  - Veri yapıları temelli test (Veri yapıları için test durumları)

Kapalı kutu (Black-box) Test işlemi

- Girdi/Çıktı davranışı
  - Verilen bir değere göre, çıktı tahmin edilebiliyorsa, “birim” testi geçecektir.
- Test durum sayılarının düşürülmesi (Denklik bölümlemesi)
  - Girdilerin bölümlenmesi (Denklik bölümlemesi)
  - Her bölümleme için, test durumu seçilmesi
    - ✓ Örnek : Bir nesne, negatif değer kabul ediyorsa, bir negatif sayı ile test edilmesi yeterli olacaktır.

Açık kutu (White-box) Test işlemi

- Hedef, kapsamadır.
  - Sınıf içindeki her ifade, en az bir defa çalıştırılır.
- 4 çeşit test
  - İfade testi
  - Döngü testi
  - Yol (Path) testi
  - Dal (Branch) testi

Açık kutu (White-box)(Devamı)

- İfade testi (Cebirsel Test)
  - Her bir ifadenin test edilmesi (Çoklu terimler içinde operatörlerin seçimi gibi)
- Döngü testi
  - Bir defa çalıştırılacak döngü
  - Birden fazla çalıştırılacak döngü
  - Döngünün hiç çalıştırılmaması
- Yol testi
  - Program içindeki tüm yolların (akışın) test edilmesi
- Dal (Branch) Testi (Koşullara bağlı test)
  - Her bir koşul durumunun en az bir defa test edilmesi
  - Örnek :

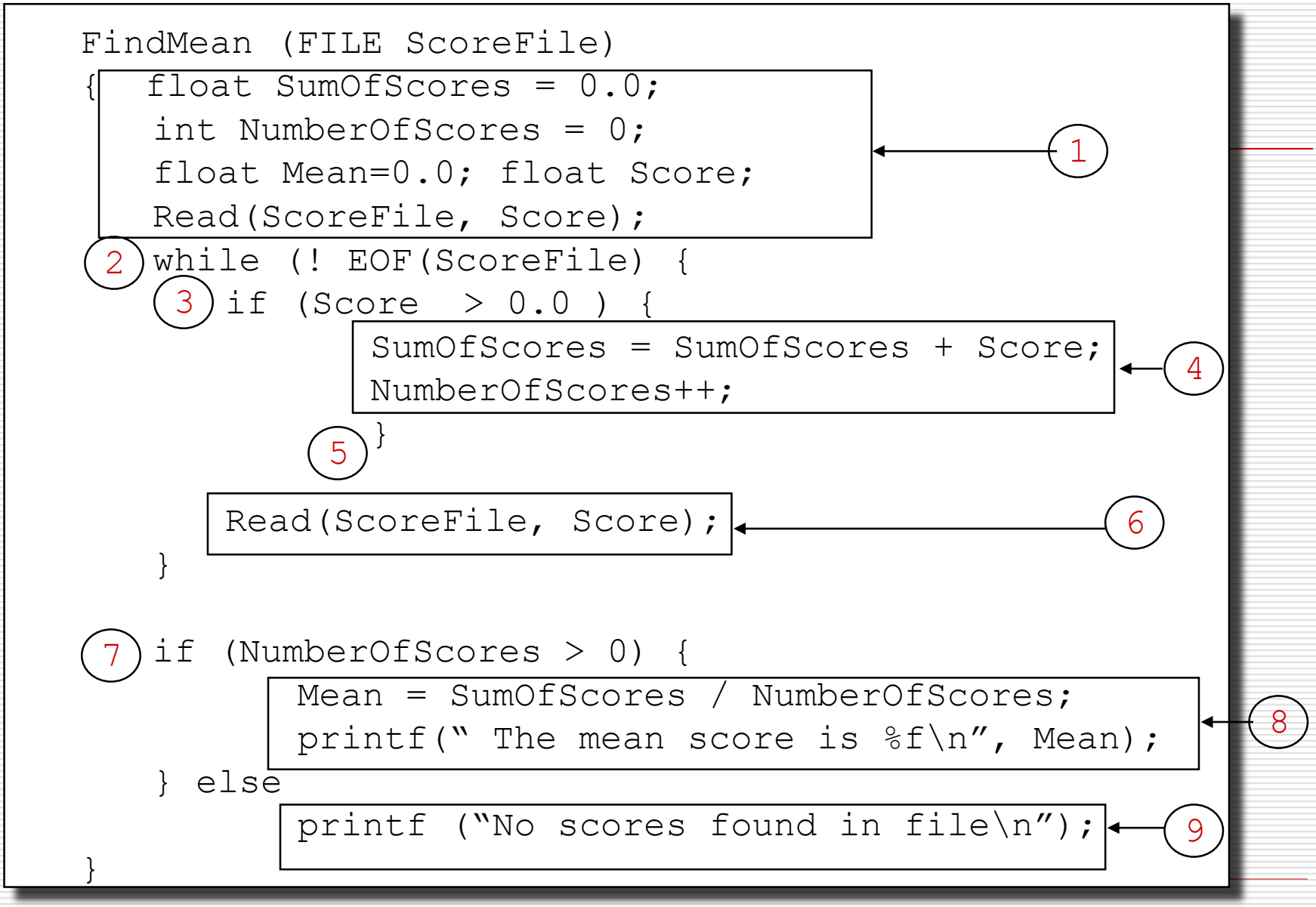
Bu ifadeyi birim testi yapmak için, kaç adet test durumuna ihtiyaç duyarız ?:2

```
if ( i =  TRUE) printf(”Yes");      else  printf(”No");
```

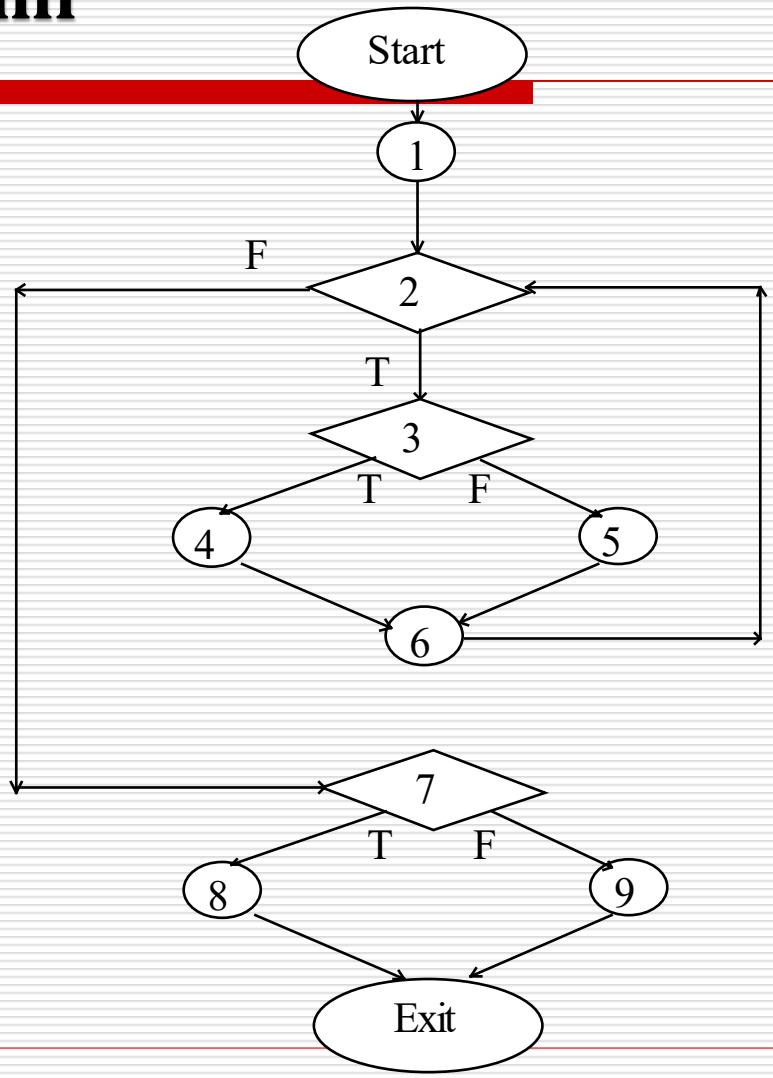
Açık kutu test örneği

```
FindMean(float Mean, FILE ScoreFile)
{ SumOfScores = 0.0; NumberOfScores = 0; Mean = 0;
 Read(ScoreFile, Score);  /* Oku ve puanları topla.*/
 while (! EOF(ScoreFile) {
         if ( Score > 0.0 ) {
               SumOfScores = SumOfScores + Score;
              NumberOfScores++;
          }
          Read(ScoreFile, Score);
 }
 /* Ortalamayı hesapla ve sonucu yaz.*/
 if (NumberOfScores > 0 ) {
             Mean = SumOfScores/NumberOfScores;
             printf("The mean score is %f \n",  Mean);
 } else
             printf("No scores found in file\n");
}
```

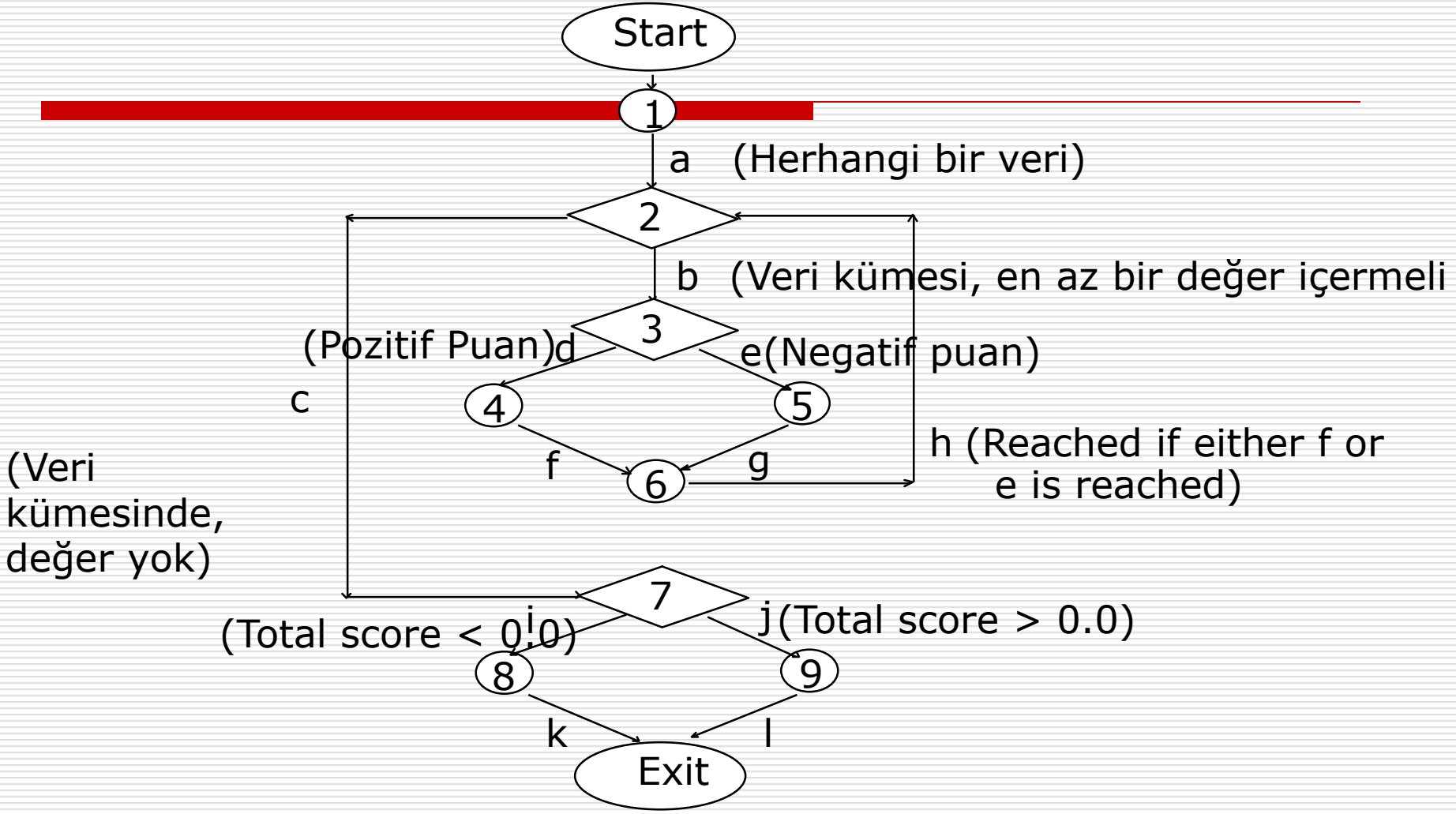
Açık kutu test örneği : Yolların belirlenmesi



Akış diyagramı



Test durumlarının bulunması



Kapalı ve Açık kutu testlerinin karşılaştırılması

- **Açık kutu (White-box) testi**
  - Bir çok alternatif akış test edilmek zorundadır.
  - ”Ne yapmalı” yerine, “Ne yapıldığını” test eder
  - Eksik “Use Case” ler bulunamaz.
- **Kapalı kutu (Black-box) testi**
  - Çok fazla sayıda (Adet Patlaması) test durumu (Geçerli ve geçersiz veriler)
  - Hatanın bulunduğu açık değildir. (Seçilen testler, kapsamayabilir.)

- Her iki test çeşidide gereklidir
- Test sürecinin, en uç noktalarıdır. (2 Uç)
- Herhangi seçilen bir test, aşağıdaki durumlara bağlıdır :
  - Mantıksal yol sayısı
  - Girdi verisi yapısı
  - Hesaplama miktarı
  - Veri yapıları ve algoritmaların karmaşıklığı

Test durum seçimi için rehber

- Analiz bilgisi (Kapalı kutu Testi)
  - "Use case" ler
  - Beklenen girdi değerleri
  - Geçersiz girdi değerleri
- Sistem yapısı ve algoritmalar hakkında Tasarım bilgisi (Açık kutu testi)
  - Kontrol yapıları
    - ✓ If blokları, Döngüler…
  - Veri yapıları
    - ✓ Dizinler…

- Uygulama bilgisi (Veri yapıları ve algoritmalar hakkında)
  - 0 'a bölme hatasına zorlama
  - Dizin kapasitesini zorlama (10 ise, 11 değeri verme gibi)

Testi ne zaman yazmalısınız ?

- Genel olarak, Kaynak kodu yazdıktan sonra
- XP (Extreme Programming) metodolojisinde, kaynak kodu yazmadan önce.
- Test öncelikli yazılım geliştirme süresi
  - Test modeline yeni test ekleme
  - Otomatik testlerin çalıştırılması
    - ✓ Yeni test başarısız olacak.
  - Hatayı düzeltecek kodun yazılması
  - Otomatik testlerin çalıştırılması
    - ✓ Çalıştığının görülmesi
  - Kodun iyileştirilmesi