**BLM 426 YAZILIM MÜHENDİSLİĞİ**

**BAHAR 2016**

**Yrd. Doç. Dr. Nesrin AYDIN ATASOY**

**11. HAFTA: YAZILIM TESTİ**

# Yazılım Testinin Temelleri

## Yazılım Testi Nedir?

Adından da anlaşılacağı üzere yazılım testi, geliştirilen bir yazılımın sistematik olarak kontrol edilmesidir.

## Tarihi Gelişimi

İlk ortaya çıktığı yıllarda, hata ayıklamak amacıyla yapılmaktaydı. Zamanla, yazılımın doğru çalıştığını doğrulamak amacıyla da yapılmaya başlandı. Yazılım testlerinin yapılışına dair kriterler, 1980'li yıllardan sonra daha da belirginleşmeye başladı. Testler, yazılım geliştirme sürecinin tümünü içerisine alacak kapsamda, aşamalı olarak yapılmaya başlandı. Günümüzde ise bunun yanı sıra hataları önlemeye yönelik testler de yapılmaktadır.

# Testin Amaçları

Yazılım testinin en temel amacı, yazılım kullanıcıya teslim edilmeden önce, olası kusurların belirlenebilmesidir. Yazılım testini de, *deneme ve kabul* olmak üzere ikiye ayırabiliriz.

## Deneme Testleri

Yazılımın doğru çalışıp çalışmadığını anlamak amacıyla yapılan ön testlerdir. Deneme testlerinde şu sorulara yanıt aranır:

1. Yazılım çalışıyor mu?
2. Çalışmıyorsa, derleme ve bağlama aşamaları gözden geçirilir. Çalışıyorsa;
3. Yazılım, girdiler üzerinde doğru işlem yapıp, doğru çıktı üretiyor mu?
4. Yazılımın hata toleransı nedir?
5. Yazılımın, sistem kaynaklarının kullanımına dair davranışlarını nasıldır?
6. Varsa, mevcut hatalar; yeni hatalar da doğuruyorlar mı?

## Kabul Testleri

Kabul testleri, yazılımın hedef sistem üzerinde doğru çalışıp çalışmadığını sorgularlar.

1. Kullanıcı, program ara yüzünü kullanabiliyor mu?
2. Kullanıcının platformunda, program işlevlerini yerine getiriyor mu?
3. Program kullanıcının isteklerini karşılamada yeterli mi?
4. Yazılım özelleştirebiliyor mu, konfigüre edilebiliyor mu?
5. Sistemin ağır yük altında olduğu durumlarda, programın performansı ne derece etkileniyor?

gibi soruların yanıtları aranır.

# Testin Yapılışı

Yazılım testini yapabilmek için ihtiyacımız olanlar:

* Test edileceklerin listesi
* Test planlaması (Hangi test, ne zaman yapılacak)
* Test sistemi (Hedef sistemin benzeri)
* Kaynak kod
* Test yardımcı gereçleri (Ölçüm, veri girişi-çıkışı için)

Yukarıdakilere sahip olunduktan sonra, test senaryoları, tek tek uygulanır. Sonuçlar toplanır, kaydedilir; kayıtlar beklenen değerlerle karşılaştırılır ve sonuca varılır. Sonuç elde edilemiyorsa, buna mani teşkil eden hatalar tespit edilir ve giderilir.

Sonuçlar elde edildiğinde ise, kabul edilebilir hataya sahip yazılımlar kullanıcıya teslim edilir.

# Test Yöntemleri

## Kara Kutu Testi

Yazılım ara yüz düzeyinde, yazılımın işlevlerini yerine getirip getirmediğini sınamak amacıyla yapılan testlerdir. Girdi değerleri yazılıma temin edilir, uygulama kapalı bir kutu gibi düşünülür ve doğrudan çıktıya bakılır. Eğer yazılım, beklenen çıktıyı veriyorsa, test başarılıdır.

## Saydam Kutu Testi

Yazılımın tüm içyapısının sınandığı testlerdir.

## Tasarım Tabanlı Test:

Modüllerin iç yüzlerinin birer kara kutu gibi görülmesine dayanılarak, programın test edilmesidir. Örneğin, programın arayüzünde yer alan bileşenlerin, değişik girdi-çıktı değerlerine göre, aldığı şeklin, beklenen sonuca, uygun olup olmadığının değerlendirilmesi.

## Kod Tabanlı Test:

Modüllerin içyapısının da test edilmesi gerektiği durumlarda, tasarım tabanlı test yeterli olmaz. Bu durumda kod tabanlı test yapılır. Modüllerin içyapısını test etmek amacıyla, kaynak kodda; mantık hataları, kodlama hataları, yazım hataları gibi hataların aranmasıdır.

Bu hatalara örnekler;

* Mantık hataları: Değişkene ilk değer atanmaması.
* Kodlama hataları: Dinamik belleğe sahip verilerin taşınması esnasında, sınırların aşılması.
* Akış yolu varsayımı: Bir girdinin 1 ila 10 arasında bir değer almasını varsaymak fakat, girdinin 'a' değerini alması.
* Yazım hataları: İmla hatası, syntax hataları.

# Özel Sistemlerin Testleri

## Gömülü Sistemler:

Gömülü sistem yazılımları, donanımı kumanda ettikleri için, testlerinin donanımla beraber yapılması gerekir. Örneğin, bir yazıcının test edilmesi.

## Gerçek Zamanlı Sistemler:

Gerçek zamanlı sistemler kolay bulunmadığından, testleri eldeki imkanlarla yapılır. Bu da demek oluyor ki, sistemi bulamıyorsak, benzerini kullanırız, şartları oluşturamıyorsak, laboratuar kullanırız, test aşamalarını gerçekleştiremiyorsak, varsayımlar yaparız.

## Güvenlik Sistemleri:

Güvenlik sistemleri, kesintisiz çalışıyor olmalıdır. Sistem üzerinde yer alan, alıcılar, sensörler v.b. modüllerin sürekli faal olduğu test edilmeli, yanlış alarm üretmediklerinden emin olunmalıdır.

## Geniş Paket Yazılımlar:

Paket yazılımların tam sürümü, yoğun testler gerçekleştirildikten sonra sürülmelidir. Ilk etapta, alfa, beta gibi sürümler piyasaya sürülmelidir.

## Veritabanı Sistemleri:

Veritabanı sistemleri, kayıt tutma, kayıt sorgulama, kayıtlara erişim sağlama gibi imkanlar sağladığı için, önemlidirler ve telafi edilemeyecek kayıplara yol açabilirler. Bu nedenle kullanıma sunulmadan önce, bu sistemler büyük hassasiyetle test edilmelidirler.

# Otomatik Test Araçları:

Yazılım testleri, süre ve maliyet bazında pahalı oldukları için, bu işlemin de otomasyonu kısmen mümkündür.

## Akıllı Derleyiciler:

Kaynak kodu tarayıp, tip denetimi yaparak, makine kodu üretirler. Denetim işlemi sıkı veya esnek olabilir. Sıkı denetim yapan akıllı derleyiciler, daha güvenilir kod üretirler.

## Durağan Çözümleyiciler:

Kaynak kodu inceleyerek, yapısındaki zayıf noktaları bulur ve uyarı verirler.

## Simulasyon Ortamları:

Yazılımın sanal bir sistem üzerinde çalıştırılmasını sağlarlar.

## Test Yazılımları:

Test edilmesi gereken bir yazılım birimine, veri veya olay girişi sağlayan araçlardır.

## Çevre Benzeticileri:

Özellikle gömülü, tahsisli ve gerçek zamanlı kontrol sistemlerini, gerçek çalışma koşullarında, dinamik olarak test etmek üzere, çevreden gelen etkileri, benzetim yoluyla sisteme giren ve sistemden çıkışlar alabilen benzeticilerdir.

## Sergileme Yazılımları:

Ölçüm ve hesaplama sonuçlarını grafik üzerinde temsil eden yazılımlardır. Sonuçları değerlendirme kolaylığı sağlarlar.

# Test Stratejileri:

Sistem geliştirmede önemli testlerden başlıcaları "V Modeli" ve "W Modeli" 'dir. Bu test modellemeleri, sistemin farklı kısımlarına hitap eden, değişik testlerin bir araya gelmesiyle oluşmuştur. O testler aşağıda verilmiştir:

## Birim Testi:

Bilgisayar yazılımlarının en küçük birimleri, birer yürütülebilir programdırlar. Birim testi ise, bu birimlerin test edilmesidir.

## Tümleştirme Testi:

Tümleştirme testi, ilgili donanım birimi üzerinde sorunsuz çalışan yazılım birimlerinin, bir araya getirilerek, sistem üzerinde bir bütün olarak da sorunsuz çalışıp çalışmadıklarını sınamak amacıyla yapılan testtir. Daha önceden fark edilmesi mümkün olmayan bir takım hatalar veya eksiklikler, bu test aşamasında fark edilebilirler. Örneğin, bir birimin çalışmasının diğer birimi etkilemesine dayalı beklenmeyen sonuçlar.

## Yukarıdan Aşağı Tümleştirme:

Önce ana denetim biriminin testi yapılır, sonra ona en yakın birimler ile teker teker beraber test edilir.

## Aşağıdan Yukarı Tümleştirme:

Atomik birimlerin çalıştırılıp test edilmesiyle başlar. Alt düzey birimler birleştirilerek kümeler haline getirilir.

## Yeterlilik Testleri:

Adından da anlaşılacağı üzere, yazılımın hatalarının giderilmesinin akabinde, ortaya çıkan yazılımın, yeterli olup olmadığının test edilmesidir. Sistemin yeterliliği ile ilgili testler Yazılım İsterleri Beliritimi belgesine dayanır. Genellikle iki aşamadan oluşur.

## Doğrulama:

Yazılım ürününün, gerçekleştirmesi gereken bütün işlevleri, aynen olması gerektiği şekilde gerçekleştirdiğinin doğrulanması işlemidir. Örneğin bir yordamın çıktı verme süresi 2 saniye olarak verilmişse, o yordam da 3 farklı alt yordamı çağırıyor ise ve her bir alt yordamın çalışma süresi 1 saniye ise, istenen çıktı 3 saniyede üretilir, bu da doğrulama testinden geçemeyecek bir duruma örnektir. Doğrulama sürecini de, sözleşme doğrulaması, süreç doğrulaması, isterler doğrulaması, tasarım doğrulaması, tümleştirme doğrulaması, belgelendirme doğrulaması ve kod doğrulaması şeklinde sınıflandırmak mümkündür.

## Geçerleme:

Geçerleme işlemi için, doğrulama safhasında doğruladığımız sonuçların, gerçekten doğru olup olmadığının sınanması da denilebilir ve hazırlanan yazılım gerçek sistemler üzerinde denenir. Şöyle ki, işlevin tamamlanması 2 saniye süren bir modül, sisteme entegre edildiğinde uygulama harici etkenlerden ötürü aslında 2 değil de, 2.5 saniyede tamamlanıyor olabilir. Bu da bizim teoride mevcut olan 2 saniyelik çalışma süremizin, mevcut sistemde aslında 2.5 saniye olması anlamına gelir.

## Rastgele Testler (Maymun Testi):

Bütün testlerden geçen bir yazılım, bilinçsizce, keyfi bir şekilde kullanılır ve hata oluşturulmaya çalışır, hata oluşmuyorsa, bu sevindirici haberdir.

## Sistem Testi:

Sistem kavramı, yazılımla beraber, donanımı da içine alır. Dolayısıylada, sistem testi denilince akla, bilgisayar tabanlı sistemler üzerinde, doğrulama ve tümleştirme amaçlarıyla yapılan testler gelmelidir. Örneğin donanım bağlantılarını kontrol etmek, yazılımın olası arayüz problemlerini araştırmak, veri akışını takip etmek ve hata bulucu test tasarımları hazırlamak, olası hataları raporlayabilecek düzenekler hazırlamak gibi aşamaları göz önüne getirebiliriz.

## Yükleme Testi:

Bu testin amacı, sistemin sınırlarını zorlayarak, veri işleme kapasitesini ölçmektir. Ayrıca aşırı yükleme olduğu takdirde, neler meydana gelebileceğini de önceden hesaplama ve durumu kontrol altına alabilmek adına önlemler hazırlama imkanını bize sağlar. Genellikle yoğun veri akışına sahip sistemler için yükleme testleri yapılır. Bu test farklı şekillerde yapılabilir; sistemin yüksek miktarda veri ile yüklenmesi, sistemin bellek ve disk kullanımının zorlanması ve tüm girişlerin yüksek hızda veriler ile yüklenmesi gibi.

## Germe Testi:

Anormal koşullar oluşturulduğunda, yazılım ve donanımın nasıl davranacağını tespit etmek amacıyla yapılır. Yükleme testine de benzetmek mümkündür. Bu anormal koşullara örnekler, aşağıdaki gibidir:

* Sistem donanımlarının bir kısmı çöktüğünde, geri kalan kısmıyla yazılım ayakta kalabiliyor mu?
* İşletim sistemini yükleyerek çökertmeye yönelik testler.
* Sistemin aşırı yükleme durumunda, ani etkilere verdiği tepki süresini ölçmek.
* Beklenenden fazla bellek ya da işlemci gücü kullanmak.

## Geri Kazanma Testi:

Yazılım ve donanım birimlerinde oluşan hatalara karşı sistemin kendini toparlaması ve tekrar en son sağlıklı çalışır konuma geri getirmesini sağlamak amacıyla yapılan testlerdir. Genellikle iki farklı yol ile bu geri kazanım gerçekleştirilir.

Birinci yol, yedek bir yazılım birimi, ana yazılım ile beraber sürekli çalışır. Ana yazılım çöktüğünde, yardımcı yazılımın çıkışları, ilgili donanıma verilir ve böylece kullanıcının veri kayıpları yaşamasının önüne geçilir.

İkinci yol, hataya dayanıklı yazılım tasarlamaktır. Şöyle ki, yazılım farklı farklı modüller halinde tasarlanır. Eğer herhangi bir modül çökerse, çöken modül yeniden başlatılır ve yazılım eski sağlıklı konuma geri döndürülür.

"Bu işin testi nasıl yapılır ?" sorusuna cevap olarak, şunu gösterebiliriz. Ana yazılım devreden çıkartılır, bakılır yedek yazılım devreye giriyor mu ? Devreye giriyorsa, veri kaybı yaşanıyor mu ? Yedek yazılımı devreye alana kadar, ne kadar süre geçiyor ? gibi soruların cevaplarını arayarak bu test gerçekleştirilir.

## Emniyet Testi:

Bazı bilgisayarlı sistemler, işlevlerini emniyetli olarak yerine getirmek zorundadırlar. Yani bir iş, ya gerçekleştirilir, ya gerçekleştirilmez, bunun orta yolu yoktur. Örneğin, şifre doğruysa veritabanına bağlan, oda sıcaklığı 27 dereceyi geçerse alarm sistemini çalıştır gibi. İşte bu tür sistemlerde, bir yazılım yada donanım kusuru oluştuğu takdirde, sistemin yerine getirmesi gereken ana işlevler üzerinde bir tehlike oluşur mu, oluşursa, sistemin davranışı nasıl değiştirilmelidir, gibi hususların test edilmesi, emniyet testidir.

## Başarım Testi:

Sistemin performansını değerlendirmek amacıyla yapılır. Örneğin, veri girişi ile çıkışı arasında geçen süre ne kadardır? Sistemimiz toplamda ne kadar bilgiyi işleme kapasitesine sahiptir? gibi soruların cevapları aranır.

Başarım testleri bazen germe testleri ile yapılarak aşırı yüklenmeler durumunda sistemin başarımı ölçülür.

Bu kriterler, sisteme farklı miktarda yüklemeler yapılarak ölçülür ve hangi yükte, performansın ne olduğu, aşırı yüklemede performans kaybının ne kadar olduğu gibi çıkarımlar yapılır.

# Kabul Testleri:

Sistemin müşteri tarafından kabul edilebilir olup olmadığı konusunda tasarımcıyı bilgilendiren testlerdir.

## Üretim Hattı Testleri:

Üreticinin kendi tesislerinde, tanımlanmış bir test donanımı üzerinde, üretilen yazılımı yapay verilerle sınamaya dayalı bir testtir. Bu teste, Fabrika Yeterlilik Testi ya da Fabrika Kabul Testi de denilir. Seri üretime geçirilecek olan bir donanıma uygulanan, ilk örnek kabul testide, üretim hattı testlerindendir.

## Kullanım Hattı Testleri:

Bu teste, yerleşke testleri de denilir. Sistemin kullanılacağı yerdeki donanıma, oranın şartları altında, gerçek verilerle yapılan testlerdir. Sistem çalışıyor mu, elektrik bağlantıları tamam mı, sisteme yüklü olan yazılım, sistemin çevre birimlerini kontrol edebiliyor mu, sistem çevre birimleriyle iletişime geçebiliyor mu gibi durumlar denenir.

## Deneme Testleri:

Uygulama alanında, karşılaşılması olası durumları gerçek koşularda denemek için kullanım sırasında gerçek verilerle yapılan testlerdir. Bu testlerde akla gelebilecek her türlü olağan dışı durumun denenmesi gerekmektedir. Örneğin, kombilerde yer alan kontrollü arızalar. Kombiye, gaz geliyordur, su gelmiyordur, ilgili hata kodu LCD ye yazdırılır ve hata modülü çalıştırılır.

## Alfa ve Beta testleri:

Yazılım paketi çok sayıda kullanıcı için hazırlanıyorsa, her bir müşteri için tek tek resmi kabul yapılamaz.Yazılım deneme süreci altında piyasaya sürülür. Bu süreçler alfa ve beta süreçleridir.

**Alfa Testi:** Yazılım geliştirici, ürünü kendi denetimi bulunan bir ortamda kullanıcıya sunar. Kullanıcı ürünü kullanır ve geliştiriciye izlenimlerini aktarır.

**Beta Testi:** Alfa testinden farkı, ürünün, geliştiricinin denetimi olan bir ortamda kullanılması gibi bir zorunluluğun bulunmamasıdır. Örneğin, müşteri sistemi alır götürür kendi sistemine entegre eder ve orada kullanır. Sonra da geliştiriciyle iletişime geçer ve deneyimlerini paylaşır.

# Kabul Kıstasları

Sistem kabulü için kıstaslar mutlaka önceden belirlenmeli, üzerinde anlaşma sağlanmalı ve resmi olarak belgelendirilmelidir. Test senaryosu test edilen sonuçlara geçerli ve geçersiz ibaresi konarak Test Sonuç Raporu hazırlanmalıdır.

**Şekilsel Hatalar :** Kullanıcı arayüzünün, renk ve şekillerinde, yazılarda ve kısaltmalarda, hizalamada yer alan hatalardır.

**Küçük Boyutta Hatalar :** Sistemi etkilemezler, giderilmesi kolaydır, kodlamada yapılan hatalardır.

**Büyük Boyutta Hatalar :** Geliştirme sürecinin bir kısmının tekrar yapılmasını gerektirebilen, büyük hatalardır.

**Ölümcül Hatalar :** Sistemin yanlış işlemesine neden olan hatalardır. Önemli işlevler yerine getirilemez.

# Test Yönetimi:

Test yönetimi büyük projeler için önemlidir. Projede bir test yöneticisi bulunmalıdır. Bu yöneticiye yardım edecek, ilgili ve hevesli kişilerden bir test grubu oluşturulmalıdır. Bu kişiler arasında görev dağılımı yapılmalıdır. Testlerin, ne zaman, nasıl ve hangi sırayla yapılacağını belirten sistem test planı hazırlanmalı. Sistemin kabul testlerine, müşteriler de katılmalı, izlemeli ve düşüncelerini beyan etmeli. Test sonunda, sonuçlar ilan edilir. Sonuçlara bakılarak, ürün niteliği değerlendirilir. Maaliyet, nitelik ve gerekli iyileştirme seçenekleri tartışılır. Gerekli bir iyileştirme varsa, görev dağılımı yapılır. İyileştirmeler yapılır veya daha sonraki sürümde yapılacağı müşteriye beyan edilir ve ürün müşteriye teslim edilir.

# Hata Ayıklama:

Adı üzerinde, hataların kaynağının bulunması ve giderilmesine hata ayıklama denir. Debugger ile yapılır. Debugger kullanılamayan hallerde ise, değişkenlerin değerlerini outputa göndermek gibi yöntemler kullanılır.

# Hata Ayıklama Süreci:

Önce hatanın kaynağı bulunmalıdır. Burada iş, tecrübeye kalmıştır. Farklı senaryolar türetilir ve teker teker bu senaryolar uygulamaya konularak, hataya sebebiyet veren şeye dair ip uçları yakalanmaya çalışılır. Eğer bir ipucu yakalanamıyorsa, diğer senaryoya geçilir. Eğer ip ucu yakalandıysa, bu bizim senaryomuzu doğrulayan bir delil hükmüne geçer. Gerekli düzeltmeler yapıldıktan sonra, işe başlanılan noktaya geri dönülür ve bakılır, ilerleme kat etmiş miyiz? Sorun çözülene kadar bu böyle devam eder.