1. Bölüm

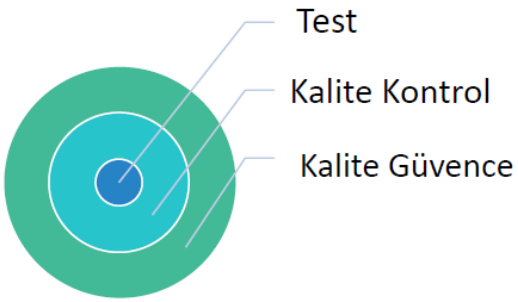
**Kalite,** müşteri isteklerine cevap verebilmektir. Yazılım kalitesi ise; en basit tanımıyla standartlara sadık kalma olarak tanımlanabilir.

**Kalite Kontrol: (**Ürün üretiminden sonra) Hata ayıklamaya yönelik, sistematik ve belirli bir anda yapılır.

**Kalite Güvence:** (Ürün üretimi veya üretim süresince) Hata oluşmasını önlemeye yönelik, sistematik, zamana yayılmıştır.

**Yazılımda Kaliteye Ulaşmak İçin**, Yazılım mühendisliği metodları, proje yönetim teknikleri, kalite kontrol eylemleri, yazılım kalite güvencesi uygulanır. Kalite güvence yönetimi sayesinde ise:

- Geliştirilen yazılımda sonradan ortaya çıkan kusurlar azalmış olur.
- Test ve bakım aşamalarında daha az iş gücü gerekir.
- Yüksek güvenilirlik müşteri memnuniyetini artırır.
- Maliyet düşer.
- Çalışanlar arasında kurulan iş disiplini sayesinde verimlilik artar.




Resim 1: Test, kalite kontrol ve kalite güvence arasındaki ilişki.

## Kalite Ölçütleri : Bir yazılımın kalitesini belirlemek için kullanılan ölçütlerdir ve 3 gruba ayrılır. Bunlar:

1. Kullanmaya Yönelik Özellikler . Kullanıcı hoşnutluğunu en iyi şekilde sağlayacak özelliklerdir.
   - ✓ İşlevsellik : Tüm isterleri karşılayabilmeli
   - ✓ Doğruluk : Tüm işlevleri istenildiği şekilde yeterli hassaslıkta yerine getirebilmeli
   - ✓ Sağlamlık : Normal olmayan çalışma koşullarında çalışmalı
   - ✓ Güvenilirlik : Sürekli ve hatasız çalışabilmeli

- ✓ Verimli Çalışma : Yazılım çalışırken sistem öz kaynaklarını uygun kullanmalı
- ✓ Korunmalı Olma : Yetkisiz kişilerin yapabileceği değişikliklere izin vermemeli
- ✓ Kullanım Kolaylığı : Üretilen yazılım rahatlıkla kullanılabilmeli
- ✓ Ekonomiklik : Uygun fiyat
- ✓ İşletim Sürekliliği : Kapat aç gerekmeden çalışabilmeli

2. Taşınmaya Yönelik Özellikler **:** Geliştirilen yazılımın başka bir yerde kullanılabilmesi için gereken özelliklerdir.
   - ✓ Taşınabilirlik : Başka ortamlara aktarılabilirlik
   - ✓ Uyumluluk : Önceden üretilen ürünlerle tam uyumlu olmalı
   - ✓ Tekrar Kullanılabilirlik : Yeni geliştirilen yazılım yeni bir uygulamada kullanılabilmeli
3. Yenileştirmeye Yönelik Özellikler: Yazılımın ihtiyaç dahilinde değişiklik yapılabilmesini sağlayan özelliklerdir.
   - ✓ Bakım Kolaylığı : Kaynak kodun anlaşılabilir bir şekilde yazılmış ve sorun dahilinde testinin kolay olması gereklidir.
   - ✓ Doğrulanabilirlik : Yazılımın doğru çalıştığının test edebilme kolaylığı
   - ✓ Genişleyebilirlik : Bir yazılıma yeni işlevler ekleyebilme özelliği

## Garvin Kalite Ölçütleri

- ✓ Performans kalitesi
- ✓ Özellik kalitesi
- ✓ Güvenilirlik
- ✓ Uygunluk
- ✓ Dayanıklılık
- ✓ Kullanılabilirlik
- ✓ Estetiklik
- ✓ Algılama

## McCall Kalite Faktörleri

- ✓ **Ürün Taşıma**
  - Taşınabilirlik
  - Yeniden Kullanılabilirlik
  - Birlikte Çalışabilirlik
- ✓ **Ürün İşletme**

- Doğruluk
- Güvenilirlik
- Verimlilik
- Bütünlük
- Kullanılabilirlik

✓ **Ürün İnceleme**

- Bakım kolaylığı
- Esneklik
- Test edilebilirlik

## ISO 9126 Kalite Faktörleri

✓ Kullanılabilirlik
✓ Taşınabilirlik
✓ Verimlilik
✓ Güvenilirlik
✓ Fonksiyonellik
✓ Bakım Kolaylığı

**Kalitenin maliyeti:** Kaliteli yazılım üretmenin maliyeti, düşük kaliteli yazılım üretmenin maliyetinden daha düşüktür.

## Resmi Kalite Güvence Yöntemleri

✓ Doğruluğun Kanıtlanması : Kullanılan programlama dili ile yazılan kodun algoritmalara uygun
✓ İstatiksel Yaklaşım : Yazılım kusurları toplanır ve sınıflandırılır, kusurlar düzeltilir
✓ Temiz Oda Süreci : Geliştirme ortamının her türlü zararlı etmenden arındırıldığı kabul edilir
✓ Yardımcı Araç Desteği : Nitelikli kod üretimi ve üretilen kodun sınanması için araçların kullanılması

## Nitelik Sistem Standartları

✓ Yetenek Olgunluk Modeli (CMM)

- **Başlangıç (Initial)** : Başarı sadece bireylere bağlıdır, Örgüt başarılı bir yazılım geliştirebilir.
- **Yönetilen (Managed)** : Yazılı olmayan kısmen tutarlı süreçler vardır.
- **Tanımlı (Defined)** : Firma kültürü yazılı hale gelmiştir.
- **Nicel Olarak Yönetilebilen (Quantitatively Managed)** : Süreç ve ürün ölçütleri toplanmış süreç iyileştirme çalışmasına geçilmiştir.
- **En iyilenen (Optimizing)** : Kurumsallaşmaya gerçekleşmiş, süreç iyileştirmek hedef

✓ Trillium : Teknolojik olgunluk içeren iyileşmeyi bu olgunlukla düzenleyen yol haritası

✓ Spice : İki boyutlu model, İçe dönük iyileştirme --- Dışa dönük yetenek belirleme

✓ TickIT : 2 aşamalıdır. Birinci aşama nitelik sistemi, ikinci aşama standarda uyumlu çalışıp çalışmadığı

ISO – 9000 : Nitelik yönetimi ve nitelik güvence standartları seçim, kullanım rehberi

ISO – 9001 : Nitelik sistemleri- tasarım, geliştirme, üretim, tesis ve servis için güvence

ISO – 9002 : Üretim ve tesis için nitelik güvence

ISO – 9003 : Nitelik Sistemleri – Son muayene ve testlerde nitelik güvence

ISO – 9004 : Nitelik yönetimi ve nitelik sistem öğeleri

AQAP-150/160 : Askeri amaçlı prosedür

| ISO 9001 | CMM | SPICE |
|---|---|---|
| 2 Düzeyli | 5 Düzeyli | 2 Boyutlu – Çok Düzeyli |
| Belgeleme | Yetenek belirleme ve belgeleme | İyileştirme ve yetenek belirleme |
| Kısa, Soyut ve genel amaçlı belge | Uzun ve somut ölçekler | Ayrıntılı |
| Kapsamlı değerlendirme | Kapsam değerlendirme | Küçük çapta yada kapsamlı |

2. Bölüm.

# Çevik Yazılım Yöntemi


- ***Tekrarlamalı*** ve ***artımsal*** bir ürün geliştirme yöntemidir.
- **Bireyler ve etkileşimi**, süreç ve araca tercih eder.
- **Çalışan bir yazılımı**, detaylı ürün belgelendirmeye tercih eder.
- **Müşteri ile işbirliğini**, sözleşmedeki kesin kurallara tercih eder.
- **Değişikliklere uyum sağlayabilmeyi**, belirli bir plana tercih eder.

✓ Özellikler

- Hızlı, devamlı ve kullanışlı yazılım ile müşteri memnuniyeti
- Geliştirici ile iş adamları arasında yakın işbirliği
- Çalışan yazılım en önemli ölçüt
- Tekrarlamalı ve artımsal bir ürün geliştirme yöntemidir.
- Bireyler ve etkileşimi, süreç ve araca tercih eder.
- Çalışan bir yazılımı, detaylı ürün belgelendirmeye tercih eder.
- Yüz yüze görüşme iletişim için en güzel yol
- Basitlik önemli
- Yazılım geliştirilmesindeki geri dönüş (feedback) ile değişikliklere uyum sağlamak önemli

✓ Temel Prensibler

- Müşteri memnuniyet
- Değişen ihtiyaç karşılanması
- Yüz yüze iletişim önemi
- Kendi kendine organize takımlar kurmak

✓ Çevik Model Takımları

- Bir araya gelmiş
- Çapraz fonksiyonlu
- İşine odaklanmış
- Hedefleri net olan

- Küçük(3-7) kişilik gruplar

## Geleneksel Model vs. Agile :

Çağlayan modeli yazılım projesinin baştan sona planlandığı bir modeldir. Bu sebepten dolayı değişikliklere uyum gösterilemez.

- ✓ Çağlayan :
  - Baştan sona planlanır,
  - Belli aralıklara bölünmeye uygun değil
  - Geleneksel Yöntemler
    - Müşteriler ne istediğini iyi bilir
    - Geliştiriciler neyi, ne şekilde üreteceğini iyi bilir
    - Bu yol boyunca hiç bir şey değişmez
  - Çevik Yöntemler
    - Müşteriler ne istediğini keşfeder
    - Geliştirici keşfeder
    - Yol boyunca değişiklik yapılabilir

| Ölçüm | Çevik Modelleme | Çağlayan Modeli |
|---|---|---|
| Planlama ölçeği | Kısa dönemlik | Uzun dönemlik |
| Müşteri ile geliştirici arasındaki mesafe | Kısa | Uzun |
| Özelleştirme ve uygulama arasındaki zaman | Kısa | Uzun |
| Sorunları keşfetmek için zaman | Kısa | Uzun |
| Proje tamamlanma riski | Düşük | Yüksek |
| Değişikliklere uyum yeteneği | Yüksek | Düşük |

Resim 2: Çevik Model ile Çağlayan Modeli arasındaki farklar tablosu.

## Scrum Modeli

- ✓ Scrum Takımı
  - Ürün sahibi – Geliştirme ekibi – Scrum Master' dan oluşur
  - Takım kendi kendini örgütler
  - Uyum sağlayan takımlar başarılı sonuçlar verir
- ✓ Backlog
  - Müşteriden ve son kullanıcıdan gelen gereksinimleri içerir
  - Herkese açık
  - User Story'lerden oluşur
- ✓ Sprint
  - Belirli bir süreye sahiptir
  - Sonucunda ortaya değeri olan bir çıktı
  - Toplantılarla içerik belirlenir.
  - Sprint boyunca her gün toplantı yapılır.

**User Story** : Müşteri, son kullanıcı veya ürün sahibi için değerli olan ve anlam ifade eden fonk.

Özellik olan ifadelerdir.

| **SCRUM** | **XP(Extreme Programming)** |
|---|---|
| Sprint (2 hafta-1 ay) | Sprint (1 yada 2 hafta) |
| Sprintler en son halini aldıktan, toplantı yapıldıktan sonra değişmez | Sprint değişebilir. |
| Özellikler geliştirici tarafından derecelendirilir. | Özellikler ürün sahibi tarafından derecelendirilir. |
| Mühendislik pratiği tanımlamaz | Mühendislik pratiği tanımlar |


Resim 3: Örnek bir user story.

## 3. Bölüm

**Yazılım testi,** yazılımın daha önce tanımlanmış teknik ve işlevsel gereksinimleri karşılayıp karşılamadığının ve yazılımın beklendiği gibi çalışıp çalışmadığının tespitidir. **Yazılım test süreci** de temel olarak elde edilen ürünün beklenen kalitede olduğunu belirlemek, değilse istenilen kaliteye ulaştırılmasını sağlamayı amaçlayan bir süreçtir. Bir başka ifadeyle bir yazılımın **doğrulanması ve geçerlenmesi** süreci olarak da tanımlanabilir.

**Hiçbir** yazılım mükemmel değildir ve bir yazılım asla %100 test edilemez.

**Testin amaçları**

- Müşteriye sunmadan önce **ürün kalitesi**nden emin olmak,
- Yeniden çalışma ve geliştirme için **masrafları azaltmak**,
- Geliştirme işleminin erken aşamalarında hataları saptayarak ileri aşamalara yayılmasını önlemek, böylece **zaman ve maliyetten tasarruf sağlamak**,
- **Müşteri memnuniyeti**ni arttırmak ve izleyen siparişler için zemin hazırlamak.

**Yazılım testinde;**

- İş gereksinimleri,
- İşlevsel tasarım gereksinimleri,
- Teknik tasarım gereksinimleri,
- Düzenleyici gereksinimler,
- Yazılımın kaynak kodu,
- Ortakların standartları,
- Donanım yapılandırılması ve dil farklılıkları gibi konular test edilmektedir.

**Doğrulama:** Yazılımı doğru mu üretiyoruz, sistemin hatasız ve iyi bir mühendislik ürünü olup olmadığını ölçer, geliştiriciler veya QA ekibi tarafından gerçekleştirilir, doğrulama aşamasında bulunan hataların maliyeti daha azdır.

**Geçerleme:** Üretilen yazılım doğru mu, sistemin kullanıcı gereksinimlerine uygunluğunu ölçer, test ekibi

tarafından gerçekleştirilir, geçerleme aşamasında bulunan hataların maliyeti daha fazladır.

**Doğrulama Süreci:** Sözleşme, Süreç, İsterler, Tasarım, Kod, Belgelendirme

| DOĞRULAMA | GEÇERLEME |
|---|---|
| Yazılımı doğru mu üretiyoruz? | Üretilen Yazılım Doğru mu? |
| Sistemin hatasız ve iyi bir mühendislik ürünü olup olmadığını ölçer. | Sistemin kullanıcı gereksinimlerine uygunluğu ölçer |
| Geliştiriciler veya QA ekibi tarafından gerçekleştirilir. | Test ekibi tarafından gerçekleştirilir. |
| Doğrulama aşamasında bulunan hataların maliyeti daha azdır | Geçerleme aşamasında bulunan hataların maliyeti daha fazladır. |

**Yazılım hataları:** Error, Failure, Fault.

**Error:** Kodlayıcı kaynaklı, doğru olmayan sonuç elde edilmesi.

**Failure:** Sistemin veya bir parçasının gerekli fonksiyonu yeterli performans tayerine getirememesi.

**Fault:** Bir yazılım içerisindeki doğru olmayan adım, işlem veya veri tanımı.

Özetlemek gerekirse; **Failure** kullanıcı gereksinimlerini veya ürün özelliklerinin doğru olarak karşılanamaması, **Fault** yazılımdan kaynaklanan hata ve problemler, **Error** ise kodlayıcı kaynaklı problemlerin ortaya çıkmasıdır.

**Hata Ayıklama Stratejileri:** Bruteforce, Backtracking, Cause Elimination.

**Bruteforce:** Yürütme anındaki davranışlar izlenir, yazılım biriminin çeşitli noktalarına ekrana veya bir dosyaya o an akışın neresinde olduğunu, genel durumunu veya bir değişkenin değerini yazan deyimler eklenir.

**Backtracking:** Kodun okunarak geri izlenmesi esasına dayanır. Hatanın oluştuğu yerden itibaren geriye doğru gidilerek kod incelenir.

**CauseElimination:** Tümevarım veya tümdengelim yöntemlerine dayanarak eldeedilen verileregöre hatanın nedeni araştırılır.

**Testi kim yapar:** Yazılım test ekibi, yazılım geliştirici, proje lideri, son kullanıcı

**Geliştirme Metodolojilerinde Test:** Şelale Modeli, Agile, V Modeli, Spiral Model

**Şelale Modeli**

Bir sonraki safhaya geçebilmek için bir önceki safhada yer alan aktivitelerin tamamlanmış olması gerekir. Yani test aşamasına gelebilmek için diğer aşamalar tamamlanmalıdır.

Hatalar sadece 5. aşamada giderilebildiğinden yazılımın maliyetini artırır ve başarısını azaltır

Kullanıcı katılımı başlangıç safhasında mümkündür. Kullanıcı isterleri bu safhada tespit edilir ve detaylandırılır. Daha sonra gelen tasarım ve kodlama safhalarında müşteri ve kullanıcılar ile diyaloğa girilmez. Bu yüzden kullanıcı testlerinde başarı yakalama oranları azdır.

**Agile**

Test profesyonelleri, yazılım geliştirme yaşam döngüsünün en başından itibaren sürece dahil olurlar. Kullanıcının geliştirici ve test sorumlusu ile aynı ortamı paylaşması prensibi, geliştirilen ürünün kullanıcının gerçekten istediği bir biçimde geliştirilmesi ve gerçekçi kullanım durumlarına göre test edilmesini sağlar. Test güdümlü geliştirme yöntemini( TDD) kullanır.

**TDD(TEST DRIVEN DEVELOPER):**

Tek satır kod yazmadan kodun testini yaz.

Testi çalıştır ve testin geçemediğini gör.

Testi geçecek en basit kodu yaz ve testin geçtiğini gör.

Kodu düzenle

Başa dön

**V modeli**

Test işlemlerinin ne zaman yapılacağını ön plana çıkarır.

Sol kanat üretim etkinliklerini, sağ kanat da test etkinliklerini gösterir.

Bu modelde geliştirme ve test paralel şekilde yapılır.

Her aşama sonunda test edilecek ürün, test grubu tarafından sınanır, onay verildikçe bir sonraki aşamaya geçilir.

İsterlerin iyi tanımlandığı, belirsizliklerin az olduğu ve aşamalar halinde ilerlenmesi gereken projelerde **«v» modeli** iyi sonuç verir.

## Spiral Model

- ✓ Aynı safhalara geri dönülmesinin bir zorunluluk olduğunu vurgular. Proje çevrimlere ayrılır ve her bir çevrimin riskleri ayrı ayrı ele alınır.
- ✓ Üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır.
- ✓ Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca karşılaştıkları için daha kolay izleme planlaması yapılır.

**Test Tipleri:** Manual ve Automation olmak üzere ikiye ayrılır.

**Manuel Test:** Küçük projelerde kullanılması uygundur.

**Automation Test:** Yük, performans, stres gibi çok kullanıcı gerektiren testlerde ve sık sık değişiklik yapılan regresyon testlerinde kolaylık sağlar.

| Otomasyon Testi | Manuel Test |
|---|---|
| Testler daha hızlı çalıştırılır. | Otomasyon testinden daha yavaştır. |
| Bir çok testi defalarca çalıştırabilir. | Bir veya iki kez çalıştırılacak olan testlerde kullanılması uygundur. |
| Sık sık değişiklik içeren regresyon testlerinde verimli çalışır. | Regresyon testlerini manuel olarak yapmak zordur. |
| Karmaşık projelerde kolaylık sağlar. | Karmaşık projeler manuel olarak test yapılmaz. |
| Test otomasyonlarını satın almak maliyetlidir. | Daha az maliyetlidir. |
| Kullanıcı ara yüzü testlerinde bazen verimli olabilir. | Kullanıcı ara yüzü testlerinde çok verimlidir. |
| Daha doğru sonuçlar üretir. | Otomasyon testinden daha az güvenilirdir. |

**Test Metotları:** Beyaz kutu, gri kutu ve kara kutu testi.

**Kara kutu testi:** Uygulamanın iç yapısıyla ilgili hiçbir bilgiye sahip olmayan test tekniğidir. Sistem mimarisiyle ilgilenilmez ve kaynak kodlara erişilmez.

Kara kutu testi kullanılarak yakalanabilecek hatalar:

1. Doğru olmayan ya da kayıp fonksiyonlar
2. Ara yüz hataları
3. Veri yapılarındaki hatalar ya da harici veritabanı bağlantısı hataları
4. Davranış ya da performans hataları
5. Başlatma ve sonlandırma hataları

| AVANTAJLARI | DEZAVANTAJLARI |
|---|---|
| ➢ Kod erişimi gerektirmediği için daha kolaydır.<br>➢ Birçok orta vasıflı test uzmanı, uygulamanın içeriği, programlama dili yada çalıştığı işletim sistemi hakkında bilgi sahibi olmadan uygulamayı test edebilir. | ➢ Test durumu tasarlamak zordur.<br>➢ Sadece birkaç test senaryosu seçilip uygulandığı için kapsamı dardır.<br>➢ Test uzmanı özel kod bölümlerini veya hata eğilimli alanları hedef alamadığı için kapsam yetersizdir. |

## Beyaz Kutu Testi:

Kodun yapısını ve iç mantık yapılarını detaylı olarak inceler. Saydam yada açıkkutu testi olarak da bilinir. Kodun iç çalışma yapısı bilinmek zorundadır.

Geliştiriciler ya da yazılım mühendisleri, birim testler hazırlayarak beyaz kutu test sürecini başlatırlar. Beyaz kutu testinde, projenin hem kaynak kodu hem de derlenmiş kodu test edilir.

Beyaz kutu testlerinde en önemli nokta kod bilgisidir.

Bu testlerin yazılım yaşam döngüsünün erken safhalarında yapılması ayrı önem taşımaktadır.

### BEYAZ KUTU TESTİ NE ZAMAN YAPILIR?

Beyaz kutu testleri yaşam döngüsünün erken aşamalarında gerçekleştiği için yazılım içindeki hatalarda erken safhada bulunmuş olur. Bu sayede düzeltme maliyeti düşük olmaktadır.

### BEYAZ KUTU TESTİNİN AVANTAJLARI VE DEZAVANTAJLARI

Beyaz kutu testinin avantajları;

Kaynak kodun yan etkileri saptanır.

Kod optimizasyonu yapılır.

Geliştiricilere kendilerini geliştirmek için fırsat verir.

Beyaz kutu testinin dezavantajları ise;

Testi yapan kişi testi yaparken önyargılı bir şekilde davranabilir. Bu durumda gözden kaçan ya da değerlendirilmek istenmeyen hatalar testin güvenirliğini sarsabilir.

Testi yapan kişi test mühendisi olacaksa, kaynak kodun ona anlatılması gerekir ve bu durumda bilgi transferi maliyetli olmaktadır.

Kara kutu ve beyaz kutu testinde sistemin yapısı, tasarımı ve uygulama tekniği test edilmektedir.

Ancak kara kutu testinde testi yapan kişinin bunların içeriğini bilmesine gerek yokken, beyaz kutu testinde testi yapan kişinin uygulamaya tamamen hakim olması gerekmektedir.

Hangi seviyelerde gerçekleştirilir?

Kara kutu testi yüksek seviyelerde (Kullanıcı Kabul Testi, SistemTesti) uygulanırken,

Beyaz kutu testi düşük seviyelerde (Birim Testi, Tümleştirme Testi, SistemTesti) uygulanmaktadır.

Testi kimler gerçekleştirmektedir?

Kara kutu testi genellikle bağımsız yazılım testçileri tarafından gerçekleştirilirken,
Beyaz kutu testi genellikle yazılım geliştiricileri tarafından gerçekleştirilir.

Programlama bilgisi

Kara kutu testinde test yapılırken herhangi bir programlama bilgisine gerek yokken,

Beyaz kutu testinde programlama bilgisine gerek vardır.

## Uygulama bilgisi

Kara kutu testinde test yapılırken herhangi bir uygulama bilgisine gerek yokken,

Beyaz kutu testinde uygulama bilgisine gerek vardır.

## Test senaryoları neye dayanır?

Kara kutu testinde test senaryoları, gereksinim özelliklerine dayanırken,

Beyaz kutu testinde, detaylı tasarıma dayanmaktadır.

İlk olarak test edilecek kodun akış diyagram oluşturulur.

Ardından tüm yollar belirlenir ve yollara değerler verilir.

Son olarak belirlenen yollar gerekli değerlerle test edilir.

| AVANTAJLARI | DEZAVANTAJLARI |
| --- | --- |
| ➢Kodun optimize edilmesine yardım eder.<br>➢Gizli hatalara sebep olabilecek gereksiz satırlar kaldırılabilir.<br>➢Test uzmanının kod hakkında bilgili olması sebebiyle, test senaryosunun kapsamı çok geniştir. | ➢Yetenekli bir test uzmanı gerektirdiği için maliyet artar.<br>➢Gizli hataları bulmak için her uç noktaya bakmak mümkün olmadığı zaman ufak problemler ortaya çıkabilir.<br>➢Kod analizcisi ve hata ayıklayıcı gibi bazı özel araçların kullanımını gerektirir. |

**Gri Kutu Testi:** Kara kutu ve beyaz kutunun birleşimidir. Veritabanı ve dokümanlara erişim vardır. Uygulamanın iç işlemlerine kısmen erişime izin verir.

| AVANTAJLARI | DEZAVANTAJLARI |
| --- | --- |
| ➢Beyaz kutu ve kara kutu testinin yararlarının birleşimini sunar.<br>➢Gri kutu test uzmanları, kaynak kod yerine ara yüz tanımlamaları ve fonksiyonel özellikleri kullanır.<br>➢Test, tasarımcı bakış açısıyla değil kullanıcı bakış açısıyla yapılır. | ➢Kaynak koda erişim sınırlı olduğundan kod inceleme ve test kapsamı sınırlıdır.<br>➢Yazılım tasarımcısı halihazırda bir test çalıştırıyorsa gri kutu testi gereksiz olabilir.<br>➢Mümkün olan her giriş sisteminin test edilebilmesi gerçekçi değildir çünkü çok fazla zaman alır. Bu yüzden bazı program yolları test edilemez. |

**Birim Test nedir?**

- Birim test, bir yazılımın en küçük test edilebilir bölümlerinin, tek tek ve bağımsız olarak doğru çalışması için incelendiği bir yazılım geliştirme sürecidir.
- Birim test yazılım testinin ilk seviyesidir ve entegrasyon testinden önce gelir.
- Birim testleri geliştiriciler kendileri yazar ve yürütürler.

**NEDEN BİRİM TEST YAPILIR?**

- Amaç, her birimin tasarlandığı şekilde gerçekleştiğini doğrulamaktır.
- Birim test yapmak kodda yeniden düzenleme(Refactor) işlemini yapmayı kolaylaştırır.
- Kodda değişiklik yapıldığında birim testi çalıştırıp oluşturduğumuz algoritmaya uygun bir şekilde çalışıp çalışmadığını test edebiliriz.
- Birim testler tüm hataları ortaya çıkarmayabilir.
- Her parça izole bir şekilde test edilmektedir.
- Birleştirme işlemi yapıldığında her şeyin düzenli bir şekilde çalışması beklenemez.

BİRİM TEST İÇİN KULLANILAN BAZI UYGULAMALAR / EKLENTİLER

- JUnit
- Spock
- Nunit
- Mocha

BİRİM TEST NASIL YAPILIR?

- Geliştirilecek yazılımları belirlemeden önce birim testlerinin yapılması gerekmektedir.
- Birim testin yapılması için belirli kurallar bulunmaktadır.
- En küçük parçacık test edilmelidir.
- Sadece bir senaryo üzerinde test işlemi gerçekleştirilir.
- Kullanılan adımlar belirlenir.
- Test metodunun ismi test edilen senaryonun yansıması olmalıdır.
- Test edilen kısım diğer kısımlardan bağımsız olmalıdır.
- Testler tam otomatik şekilde çalışmalıdır.
- Hızlı çalışmalı ve çabuk sonuç üretmelidir.
- Okunaklı, anlaşılır ve sürdürülebilir olmalıdır.
- Test başarısız olduğunda durmalı ve iyi bir hata raporu döndürmelidir
- Hata raporunda neyi test ettiğimizi belirtmelidir.
- Beklenen çıktı ile gerçekleşen çıktıyı karşılaştırmalıdır.

# Birim test hakkında

- Birim test işlemleri, Visual Studio tarafından tanımlanmış olan Assert.AreEqual() metodu ile yapılmaktadır.
- Daha çok beyaz kutu yönteminin kullanıldığı birim testi ile düzgün ve hatasız çalıştığına kanaat getirilen bir birim artık tümleştirme testi için hazır hale gelmiş olur.
- Yazılan kodun her satırının başka bir kod tarafından test edilmesini sağlar.
- Kodun anlaşılmasını kolaylaştırır.
- Daha hızlı yazılım geliştirmeyi sağlar.
- Koddaki hata oranını azaltır.
- Kodların kalitesinin artmasını sağlar.
- Hataların çabuk tespit edilip düzenlenmesini sağlar.

**Birimtestdurumları:**

- Arayüz
- Yerelveriyapıları
- Sınırkoşulları
- Bağımsızyollar
- Hatayakalamayolları

**Tümleştirme Testi:** Birden fazla biriminin bir araya getirilerek uyumlu bir şekilde ve hatasız çalışması, her birinin tek tek değil de bir bütün içinde, tasarımda belirtildiği şekilde kendi üzerlerine düşen görevleri yerine getirip getirmediği tümleştirme testi ile kontrol edilir.

Tümleştirme testinin yapılma nedenleri:

- Bir birimin çalışması başka bir birimin çalışmasını etkileyebilir.
- Birimler arasındaki arayüzler arasında verilerin kaybolma olasılığı vardır.
- Bir birim içinde kabul edilebilir sınırlar içinde olan kesinlik değerleri birden fazla birimin devreye girmesi ile kabul edilemeyecek değerlere ulaşabilir.
- Birimler arasında eşzamanlılığın sağlanması gerekir.
- Birimler arasında paylaşılan evrensel veri yapıları sorun çıkarabilir.

Artırımlı tümleştirme yönteminde değişik stratejiler kullanılabilir. Bunlar

- Yukarıdan aşağı =Bu stratejide, önce ana denetim biriminin testi yapılır, sonra ona en yakın düzeydeki birimlerden biri ile beraber test yapılır
- aşağıdan yukarı= Alt düzey birimler birleştirilerek kümeler haline getirilir. Bu kümeler test edilir. Daha sonra bu kümelerin birleştirilmesinden oluşan daha üstü düzeyde kümeler meydana getirilir. Bu şekilde en üstte bulunan ana birime kadar ulaşılır.
- geri çekilme(regresyon)= uygulama ortamındaki yapılan tüm değişikliklerin yeni bir hata üretip üretmediğini kontrol amaçlı olarak yapılan test türüdür.

**Sistem Testi:**

- Performans Testi
- Yük Testi
- Germe(Stres) Testi
- Kurtarma Testi
- Güvenlik Testi
- Taşınabilirlik Testi
- Kullanılabilirlik Testi

**Yükleme testi:**

- Veri hacmi testi
- veri debisi testi
- kapasite testi

**Güvenlik testi:**

- Zafiyet taraması
- Penetrasyon Testi
- Risk Belirleme
- Güvenlik Denetimi
- Şifre Kırma

**Taşınabilirlik Testi:**

- Installability
- Compatibility
- Adaptability
- Replaceability

**Kullanılabilirlik Testi:**

- Öğrenilebilirlik
- Verimlilik
- Memnuniyet
- Hatırlanabilirlik
- Hatalar

**Kabul Testi:**

- Alfa Test Geliştiricinin kendi yerinde müşteri tarafından yapılır.
- Beta Testi, Birçok kullanıcının kendi ortamında yapılır. Geliştirici genellikle testlere katılmaz.

## ✓ Alfa sınaması

- Geliştiricinin kendi yerinde müşteri tarafından yapılır.
  Geliştirici bu testleri gözlemleyerek gerçek kullanım hakkında bilgi sahibi olmaya çalışır; kusur buldukça not alır ve
  düzenleme işlemlerini yürütür. Alfa testlerinin en önemli
  özelliği, denetim altındaki bir ortamda, asıl kullanıcılardan biri tarafından yapılıyor olmasıdır.

## ✓ Beta sınaması

- Birçok kullanıcının kendi ortamında yapılır. Geliştirici
  genellikle bu testlere katılmaz; yalnızca belirli aralıklarla sonuçları ve yorumları alır. Bu testin özelliği de geliştirici tarafından kontrol edilemeyen gerçek uygulama ortamı koşullarında yazılımın denenmesidir. Beta testi sonunda
  geliştirici, bulunan kusurları düzelterek tüm kullanıcılar için yeni bir sürüm çıkartır.

**Web Uygulama Testi:**

# • Web Uygulaması Testi

- **İçerik Testi**
  - ✓ Son kullanıcıya sunulacak içeriğin yapısı veya bütünlüğündeki hataları bulmak için,
  - ✓ Herhangi bir içerikteki sözdizimsel hataları ortaya çıkarmak için,
  - ✓ Herhangi bir içerikteki anlam hatalarını bulmak için yapılır.

İçerik Testi

Veri Tabanı Testi

## Veri Tabanı Testi

- ✓ Veri tabanı kaynakları sorgulanır.
- ✓ İlgili veriler veri tabanından ayıklanır.
- ✓ Ayıklanan veriler içerik nesnesi olarak düzenlenmiş olmalıdır.
- ✓ İçerik nesnesi, görüntülenmek için istemci ortamına aktarılır.

Kullanıcı Ara yüzü Testi

## Kullanıcı Arayüzü Testi

- ✓ Bir kullanıcı, web uygulamasıyla etkileşime geçtiği zaman bir yada daha fazla ara yüzle

etkileşim meydana gelir. Her bir arayüz mekanizması için aşağıdaki denetimler yapılmalıdır.

- ✓ **Linkler:** Her bir link, doğru içerik ve fonksiyona ulaşıp ulaşmadığından emin olmak için test edilir.
- ✓ Formlar:
  - Her bir form alanı uygun genişlik ve veri tipinde oluşturulup oluşturulmadığından emin olmak için test edilir.
  - Formlar, ön tanımlı uzunluktan daha uzun veriler girildiğinde kullanıcıyı uyarmalıdır.
  - Tüm form elemanları kullanıcının anlayabileceği, uygun, kullanışlı bir biçimde sıralanmalıdır.
  - Tarayıcıların otomatik doldurma özelliğiyle yanlış veri girilmesine engel olunmalıdır.
  - Tab tuşu (veya diğer klavye tuşları), form elemanları üzerinde uygun şekilde hareketetmelidir.
  - Form üzerinde hata denetimi yapan scriptler, hata oluştuğunda anlaşılabilir ve yönlendirmeler için hata mesajı yayımlamalıdır.
- ✓ **Uygulamaya Özgü Arayüzler:** Testler arayüz mekanizması tarafından tanımlanan özellikler ve fonksiyonların kontrol listesine göre uygulanır.

Kullanılabilirlik Testi

## Kullanılabilirlik Testi

- ✓ **Kullanılabilirlik:** Bir ürünün potansiyel kullanıcıları tarafından belirli bir kullanım bağlamı içinde, amaçlanan kullanım hedeflerine ulaşmak için, etkin, verimli ve tahmin edici bir şekilde kullanılabilmesi olarak tanımlanır. ISO 9241
- ✓ **Kişiselleştirme**: Kullanıcılar siteyi isteklerine göre kişiselleştirebiliyor mu?
- ✓ **Görüntü Karakteristikleri**: Uygulama uygun boyutta ve uygun çözünürlükte görüntülenebiliyor mu?
- ✓ **Etkileşim**: Menüler, butonlar, işaretleyiciler gibi etkileşim mekanizmalarını anlamak ve kullanmak kolay mı?
- ✓ **Düzen**: Navigasyonlar, içerik ve fonksiyonlar kullanıcının kolayca bulabileceği şekilde yerleştirilmiş mi?
- ✓ **Okunabilirlik:** Yazılar ve Grafik sunumları kolay anlaşılabilir mi?
- ✓ **Estetik:** Resimler, renkler, yazı tipi uyumlu mu? Kullanıcı uygulamaya bakınca iyi hissediyor mu?
- ✓ **Zaman duyarlılığı**: Önemli özellikler, fonksiyonlar ve içerikler kısa zamanda görüntülenebiliyor mu?
- ✓ **5 saniye kuralı:** Kullanıcı ilk 5 saniyede site hakkında bilgi edinir.
- ✓ **2 saniye kuralı**: Kullanıcıların birçoğu siteye girdiklerinde kalma veya gitme kararını ilk 2 saniyede verir.

- ✓ **3 tık kuralı**: Bir çok kullanıcı sitede aradığı bilgiye ortalama 3 tık ile ulaşamazsa siteyi kullanmaktan vazgeçer.
- ✓ **80/20 kuralı**: Tasarımda etkinin %80 i mevcut görsel tasarımın %20 sinden gelir.
- ✓ **7 ± 2 kuralı**: İnsanın yakın zaman hafızası aynı anda 5 ila 9 olguyu hatırlamasına olanak verir. Dolayısıyla 7±2 prensibiyle menülerdeki seçenekler ortalama 7 adet olarak belirlenmelidir.
- ✓ **A/B Testi :** A/B testleri ile bir sayfanın iki (ya da daha fazla) değişik versiyonu test edilir. İki değişik tasarımdan hangisinin daha başarılı olduğu ya da bir sayfa elemanın hangi versiyonunun daha verimli olduğu A/B testleri kullanılarak belirlenebilir.
- ✓ **Ağaç Testi:** Ağaç testi, bir web sitesindeki konuların, başlıkların bulunabilirliğini ölçmek için yapılan bir kullanılabilirlik testidir. Geniş içerikli bir web site, genellikle konular ve alt konuların bulunduğu bir hiyerarşik bir yapıda oluşturulur. Ağaç testi, kullanıcıların bu hiyerarşide aradıkları nesneleri kolayca bulabilmeleri için yapılır.
- ✓ **Gerilla Testi:** Gerilla test, kullanıcıların göz izleme cihazı olmadan ve laboratuvar dışında test edildiği bir metottur. Test kullanıcın yanında bir moderatör tarafından yönetilir ve kullanıcıdan sesli düşünmesi istenir. Test süresince kullanıcının mouse hareketleri, mimik ve sesli düşünceleri kayıt altına alınır. Gerilla test, göz izleme testine kıyasla daha ekonomik bir metottur.
- ✓ **Göz İzleme Cihazı İle Kullanılabilirlik Testi:** Kullanıcının nereye, ne kadar süre ve kaç kere baktığına, anlık ve geçmiş dikkatinin nerede yoğunlaştığına, niyetine, zihinsel durumuna ilişkin bilgi sağlamakta kullanılan bir yöntemdir.

UyumlulukTesti

## Uyumluluk Testi

- ✓ Çözünürlük, bağlantı hızı, istemci işlem hızı gibi faktörlerden dolayı farklı sonuç verebilir. Buradan doğacak istenmeyen sonuçları engellemek için uyumluluk testi yapılır. Her farklı kombinasyon için var olan arayüz, navigasyon, performans ve güvenlik testleri uygulanır.
- ✓ **Donanım:** CPU, memory, storage, and printing devices
- ✓ **İşletim Sistemleri**: Linux, Macintosh OS, Microsoft Windows, a mobile based OS
- ✓ **Tarayıcılar**: Firefox, Safari, Internet Explorer, Opera, Chrome vb.
- ✓ **Kullanıcı Arayüz Bileşenleri**: Active X, Java applets vb.
- ✓ **Plug-ins**: QuickTime, RealPlayer vb.
- ✓ **Bağlantılar**: Cable, DSL, regular modem, T1, WiFi

BileşenSeviyeliTest

## Bileşen Seviyeli Test

- ✓ **Bileşen seviyeli test**, web uygulaması fonksiyonlarındaki hataları ortaya çıkarmak için yapılır. Her bir fonksiyon yazılımın bir bileşenidir ve kara kutu (bazen beyaz kutu) yöntemiyle test edilir. Formlara değerler girilerek doğru çıktının alınması beklenir.

NavigasyonTesti

## Navigasyon Testi

- ✓ **Navigasyonlinkleri**: Bumekanizmalarwebuygulamasıiçindekiiç bağlantıları, diğer web uygulamaları içindeki dış bağlantıları ve spesifik bir web sayfası içindeki **anchor** linkleri oluşturur. Her bir link tıklandığı zaman doğru içerik ve fonksiyona gidiyor mu diye test edilmelidir.
  - **Anchorlink:** Bir web sayfası içinde, aynı sayfanın belirli bir bölümüne yönlendirme yapan linklerdir
- ✓ **Yenidenyönlendirme:** kullanıcı var olmayan bir URL yi açmaya çalıştığında ya da içeriği kaldırılmış bir linke tıkladığında yeniden yönlendirmeler çalışmalıdır. Web sayfasının böyle bir durumda nasıl davrandığı test edilir. Kullanıcıya bir mesaj gösterilip diğer bir sayfaya yönlendirilmelidir.
- ✓ **Yerimleri:** Yerimleri bir tarayıcı fonksiyonu olmasına rağmen, web uygulamasının yer imi oluşturulurken anlamlı bir başlığının olup olmaması test edilmelidir.
- ✓ **Siteharitası:** Site haritası, sitedeki tüm web sayfalarının içeriklerinin haritasıdır. Site haritasındaki her bir elemanın doğru sayfadan yönlendirilip yönlendirilmediği test edilir.
- ✓ **İçaramamotorları:** Kompleks web uygulamaları yüzlerce hatta binlerce içerik nesnesi içerebilir. İç arama motorları kullanıcıların anahtar kelimelerle ihtiyacı olan içeriğe ulaşmasına izin verir. Arama motoru testi, aramanın tamlığını ve doğrulunu, istisna yakalama özelliğini ve gelişmiş arama özelliklerini doğrular.

GüvenlikTesti

## Güvenlik Testi

- ✓ Web uygulaması güvenlik testlerinde aşağıdaki denetimler yapılır.
  - Siteler Arası Komut Çalıştırma (XSS)
  - Enjeksion Açıkları (SQL Injection, Command Injection vs.)
  - Emniyetsiz Doğrudan Nesne Erişimi
  - Siteler Arası İstek Sahteciliği (CSRF)
  - Oturum ve Kimlik Yönetimi Açıkları
  - Mantıksal Saldırılar
  - İstemci Tarafı Saldırıları
  - Bilgi Sızdırma vb.
- ✓ **Authorization (Yetkilendirme):** İstemci yada sunucu tarafında sadece uygun yetkili kişilerin girişine izin veren bir filtreleme mekanizmasıdır. (kullanıcıID ve şifre gibi)
- ✓ **Authentication(Doğrulama):** Tüm sunucu ve istemci kimliklerini doğrulayan bir mekanizmadır, her iki taraf da doğrulandığı zaman iletişime izin verir.
- ✓ **Firewall:** İnternetten gelen bilgileri denetleyen ve ardından güvenlik duvarı ayarlarınıza göre engelleyen veya geçişine izin veren bir yazılım veya donanımdır.

- ✓ **Encryption:** Gizlenmek istenen bir bilginin bir algoritma yardımıyla bir başkası tarafından okunmasını ya da değiştirilmesini engellemek için veri üzerinde yapılan işleme şifreleme denir.

PerformansTesti

- Performans Testi
  - ✓ Performans testleri, sunucu tarafı kaynaklarının eksikliği neden olan performans sorunları ortaya çıkarmak için kullanılır.
  - ✓ Yük testi
    - Yükü çeşitli düzeylerde ve kombinasyonlarda yükleyerek uygulamanın davranışını incelemektedir.
    - Yük testinin amacı, web uygulamasının çeşitli yükleme koşullarında nasıl davrandığını belirlemektir.
  - ✓ **Stres testi**, Web uygulamasının ne kadar kapasitesi olduğunu ve kırılma noktasını belirlemek için yapılır.

**Kullanıcı Arayüzü Testi:**

İstemci taraflı scriptler

Dinamik HTML

Pop-up Pencereleri

CGI Scriptleri

Cookies
Akıcı içerikler

**Kullanılabilirlik Testi:**
Kişiselleştirme
Görüntü Karakteristikleri
Etkileşim
Düzen
Okunabilirlik
Estetik
Zaman Duyarlılığı

**5 saniye kuralı:** Kullanıcı ilk 5 saniyede site hakkında bilgi edinir.

**2 saniye kuralı:** Kullanıcının siteye girdiğinde kalma veya gitme kararını ilk 2 saniyede verir.

**3 tık kuralı:** Kullanıcı aradığı bilgiye ortalama 3 tıkla ulaşamazsa siteyi kullanmaktan vazgeçer.

**80/20 kuralı:** Tasarımda etkinin %80 i mevcut görsel tasarımın %20 sinden gelir.

**7±2 kuralı:** İnsanın yakın zaman hafızası 5 ila 9 olguyu hatırlamasına olanak verir. Bu yüzden 7±2 prensibiyle menülerdeki seçenek ortalama 7 adet olarak belirlenmeli

**Kullanılabilirlik Test Çeşitleri:** A/B Testi, Ağaç Testi, Gerilla Testi, Göz izleme cihazı testi

**A/B Testi:** Bir sayfanın iki farklı versiyonu test edilir.

**Gerilla Testi:** Göz izleme cihazı olmadan ve laboratuvar dışında test edilen bir metottur. Moderatör tarafından yönetilir ve kullanıcıdan sesli düşünmesi istenir. Test esnasında kullanıcının Mouse hareketleri, mimik ve sesli düşünceleri kayıt altına alınır.

**Göz izleme cihazi testi bulguları:** Kullanıcı Videoları, Yol haritaları, Mouse hareketleri, zaman istatistikleri, sıcaklık haritaları.

**Uyumluluk Testi:** Donanım, İşletim Sistemleri, Tarayıcılar, Kullanıcı Arayüz Bileşenleri, Plug-ins, Bağlantılar

**Anchor link:** Bir web sayfası içinde, aynı sayfanın belirli bir bölümüne yönlendirme yapan link.

**Güvenlik Testi:** XSS, Enjeksion Açıkları (SQLi, Command Injection), Emniyetsiz Doğrudan Nesne Erişimi, Siteler Arası İstek Sahteciliği (CSRF), Oturum ve Kimlik Yönetimi Açıkları, Mantıksal Saldırılar, İstemci Tarafı Saldırılar, Bilgi Sızdırma)

**Güvelik açıklarından korumak için:** Authorization(Yetkilendirme), Authentication(Doğrulama), Firewall, Encryption(Verileri şifreleme).

**Authorization(Yetkilendirme):** İstemci yada sunucu tarafında sadece uygun yetkili kişilerin girişine izin veren bir filtreleme mekanizmasıdır.

**Authentication(Doğrulama):** Tüm sunucu ve istemci kimliklerini doğrulayan bir mekanizmadır. Her iki taraf da doğrulandığı zaman iletişime izin verir.

**Performans Testi:** Web uygulama testlerinde Yük ve Stres Testi olmak üzere iki farklı performans testi gerçekleştirilir.

PERFORMANS TESTİ NEDİR?

Performans testi, yazılım uygulamalarının beklenen iş yükü altında iyi performans göstermelerini sağlamak için bir tür yazılım testi olarak tanımlanır.

PERFORMANS TESTİ NEDEN YAPILIR?

Kritik uygulamaların yüksek maliyetli arızlarını önler.

Müşterilerden önce olası sorunları bulur.

Geliştirme süresini azaltır.

Altyapı maliyetlerini azaltır.

PERFORMANS TESTİNİN AMACI NEDİR?

Amacı hataları bulmak değildir. Performans tıkanmalarını ortadan kaldırmaktır. Şu sorulara yanıt arar;

Bir görev tamamlamak ne kadar sürer? (Application Response Time – Uygulama Yanıt Süresi) Hangi yapılandırma en iyi performans seviyesini sağlar? (Configuration Sizing -Yapılandırma Boyutlandırma)

Sistem piyasaya çıkabilecek kadar sağlam mı? (Acceptance - Kabul)

Yazılımın yeni bir sürümü yanıt süresini olumsuz etkiliyor mu? (Regression – Regresyon)

Sistem ağır bir iş yükü altında nasıl çalışıyor? (Reliability – Güvenilirlik)

Performanstaki bozulma hangi noktada meydana geliyor? (Capacity Planning – Kapasite Planlaması)

Performansın düşmesinin nedeni nedir? (Bottleneck Identification – DarboğazTanımlama)

Bir yazılımın sistemini kontrol etmek için;

Hız, ölçeklenebilirlik ve stabilite önem arz etmektedir.

Load Testing (Yük Testi): Uygulamanın beklenen kullanıcı yükleri altında gerçekleştirme yeteneğini kontrol eder. Amacı yazılım uygulaması yayınlanmadan önce performans darboğazlarını tespit etmektir.

Stress Testing (Stres Testi): Yüksek trafik veya veri işlemeyi nasıl işlediğini görmek için bir uygulamayı aşırı iş yükleri altında test etmeyi içerir. Amaç, bir uygulamanın kırılma noktasını belirlemektir.

Endurance Testing (Dayanıklılık Testi): Sistemin sürekli beklenen yükü ne kadar devam ettirebileceğini belirlemek için kullanılır.

Performans test işlemleri, Visual Studio tarafından tanımlanmış olan Stopwatch sınıfının fonksiyonları ile yapılmaktadır.

**Yük Testi:** Yükü çeşitli düzeylerde ve kombinasyonlarda yükleyerek davranış incelemesi.

P=gerekli olan bağlantı hızı, N=eş zamanlı kullanıcı sayısı, T=zaman birimi başına çevrimiçi işlem sayısı, D= sunucu tarafında işlenen işlem başına veri yükü


**Stres Testi:** Uygulamanın ne kadar kapasitesi olduğunu ve kırılma noktasını belirlemek.

- Normal olmayan koşullarda, hem yazılım hem de donanımın ne şekilde davranacağını görmek üzere germe(stress) testleri yapılmalıdır.
- Sistemi normal olmayan miktarda öz kaynak gerektirecek şekilde zorlamak amacıyla yapılan testler.
- Sistemin kaldırabileceği yük durumunda, ani etkilere verilecek tepki süresini ölçmek üzere yapılan testler.

**Mobil Uygulama Türleri:** Native, Hibrid, Web Tabanlı Uygulamalar

**Native Uygulamalar**

- ✓ Native bir uygulama, belli bir platforma özel, genellikle platform sağlayıcısının üretmiş/belirtmiş olduğu platform SDK'sı, araçları ve programlama dili yardımıyla geliştirilmiş uygulamadır. Örneğin Objective-C programlama dilini kullanarak IOS işletim sistemine özel ya da Java programlama dili ile Android işletim sistemine özel geliştirilen bir uygulama Native uygulamadır.

- Web Tabanlı Uygulamalar
  - ✓ Mobil web uygulamaları ise HTML5, CSS3 ve Javascript gibi web teknolojileri kullanılarak gerçekleştirilir ve adından anlaşılacağı üzere "Web" tabanlıdırlar. Mobil işletim sistemlerine özgü üretilmiş mobil web tarayıcıları üzerinde sorunsuz çalışmaları için optimize edilmişlerdir. Mobil Web uygulamaların en öne çıkan özelliği, web tabanlı oldukları için tek seferde geliştirilip, tüm mobil platformda çalışabilir olmalarıdır.
- Hibrid Uygulamalar
  - ✓ Hibrit uygulamalar Native ve HTML5'in karışımından meydana gelen uygulamalardır. Tıpkı Native uygulamalarda olduğu gibi cihaza özel geliştirilir ve HTML 5'de olduğu gibi web teknolojisi kullanılarak yazılır

**Mobil Uygulama Testi:** Kurulum, Fonksiyonel, Performans, Güç Tüketim, Kesme, Kullanılabilirlik

- Mobil Uygulama Testi
  - **Kurulum Testi**
    - ✓ Herhangi bir mobil uygulamanın kurulumu basit olmalı ve kurulumun ilerledurumu hakkında bilgiler vermelidir. Test uzmanları kurulum testi ile bu durumu kontrol eder. Böylece kullanıcı uygulamanın yüklenme zamanını tahmin etmek zorunda kalmaz.
  - Fonksiyonel Test
    - ✓ Uygulama tüm cihazlarda test edilmelidir.
    - ✓ Uygulamanın tüm fonksiyonları test edilmelidir.
    - ✓ Bellek tamamen doluysa da uygulama düzgün çalışmalıdır.
    - ✓ Uygulama, sayfa yüklenirken veya yönlendirilirken zorla durdurulmuş ise bilgilendirici mesaj vermelidir.
    - ✓ Uygulama herhangi e-ödeme sistemi veya bilgi içeriyorsa verilerin son derece güvenli olduğundan emin olunmalıdır.
    - ✓ Monkey testi gerçekleştirilmelidir.
  - Güç Tüketim Testi
    - ✓ Uygulamanın çalışma sürecinde cihazın bataryasını ne oranda kullandığı test edilir. Çok şarj tüketen bir uygulama kullanıcı için verimli olmayacaktır
  - Kullanılabilirlik Testi
    - ✓ **Kullanılabilirlik:** Bir mobil uygulama geliştirilirken ekstra dikkat gerektiren faktör kullanılabilirliktir. Herhangi bir uygulamanın başarısı ya da başarısızlığı kullanıcı ara yüzüne bağlıdır.
    - ✓ Uygulamanın arayüzü her cihaz için uyumlu olmalıdır. Ekrana sığmamış, kaymış kontroller olmamalıdır. Yazılar okunaklı olmalıdır.
    - ✓ Uygulama veya sayfa yüklenirken ilerleme göstergesi bulunmalıdır. Böylece kullanıcı bazı verilerin yüklendiğinin farkında olur.
    - ✓ Buton boyutları düzgün olmalıdır ve geniş parmak tipleri tarafından kolayca

## Performans Testi

- Kullanıcı arayüzü ve fonksiyonel testlerde fark edilemeyen istisnaları bulur.
- Uygulama beklenen maksimum yük taşıma kapasitesinden %20 daha fazlasını taşıma yeteneğine sahip olmalıdır.
- Cihazın bataryası düşük iken uygulama çalıştırıldığında, cihazın işletim sistemi tarafından dayatılan herhangi bir aksaklık veya askıya alınma olmadan uygulamanın problemsiz bir şekilde çalıştığı doğrulanmalıdır.

**Kesme Testi:** Ağ bağlantısı çeşidi, SD kart etkileşimi, Arama ve mesajlar

- ## Kesme Testi

✓ **Ağ bağlantısı çeşidi:** Uygulama Wi-Fi, 4g, 3g ve ya 2g bağlantılarında kolaylıkla çalışmalıdır.

- Cihaz herhangi bir bağlantıdan diğerine geçerse, uygulama bunu otomatik olarak algılayarak diğer ağa bağlanmalıdır.
- Cihazda sim kart takılı olmasa da uygulama düzgün çalışmalıdır.
- Uçak modu aktifken de uygulama düzgün çalışmalıdır.

✓ **SD Kart Etkileşimi**: Bu faktör mobil cihazlarda hayati bir rol oynar.

- Uygulama SD kartın takılı olmaması ve herhangi bir nedenden dolayı çalışmaması durumunda işlevleri yerine getiremediği zaman kullanıcıya bilgilendirme mesajları sunmalıdır.
- Uygulamada, "SD karta taşı" şeklinde seçenek olmalıdır.

✓ **Aramalar Ve Mesajlar:** Mobil cihazın arama, SMS gibi temel işlevlerinin devreye girdiği ve uygulamanın çalışmaya ara verdiği süreç incelenir. Yeniden başlatıldığında uygulamanın işlevini olması gerektiği gibi sürdürüp sürdürmediği test edilir.

**Mobile Yazılım Test Ortamları:** Emülatör, Simülatör, Gerçek Cihazlar, Bulut tabanlı test ortamları

**Emülatör:** Sistemin işleyişini taklit eder ve taklit ettiği sistemin özelliklerini sğalar.

**Simülatör:** Gerçek bir sistemi sadece modeller, yani sadece işleyişini örnekler ve gerçeğe benzer bir ortam oluşturmaya çalışır. Gerçek sistemin yerine geçmez.

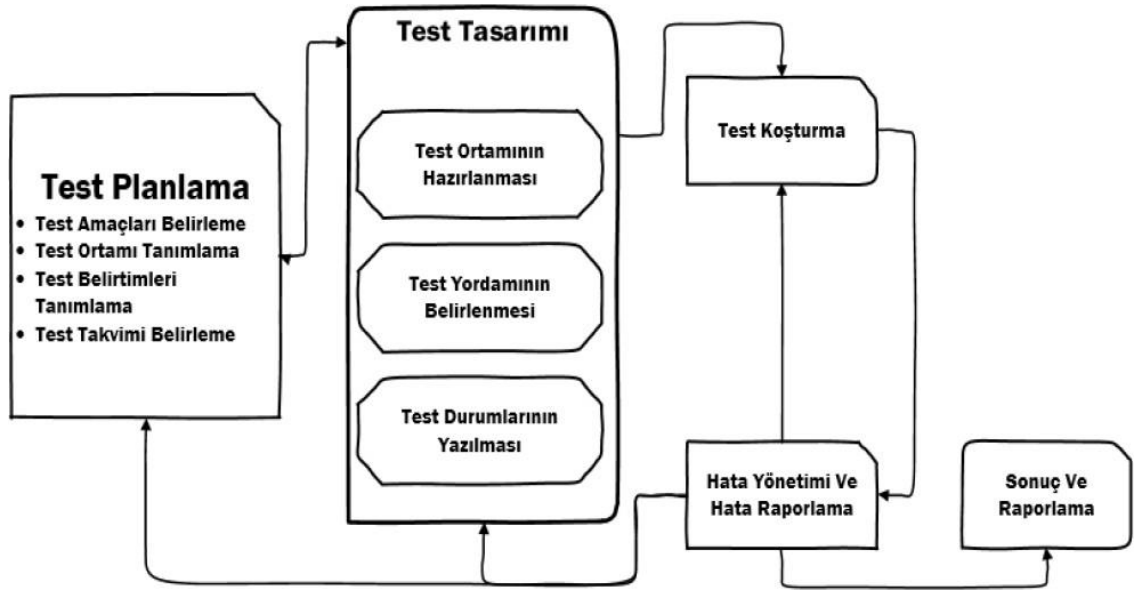
## Mobil Yazılım Test Ortamları

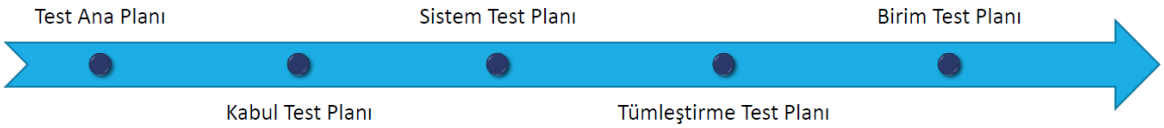
- **Emülatör:** Bir sistemin işleyişini taklit eder ve taklit ettiği sistemin sunduğu özellikleri aynen sağlar. Böylece bu sistemi kullanan diğer sistemler için bunun gerçek sistemden bir farkı yoktur.
- **Simülatör:** Gerçek bir sistemi sadece modeller, yani sadece işleyişini
- örnekler, gerçeğe benzer bir ortam oluşturmaya çalışır. Örneklediği sistemin çalışmasının anlaşılmasına yardımcı olur. Simülatör, gerçek sistemin yerine geçebilecek bir sistem değildir. Kullanım amacı bakımından emülatörden kesin bir şekilde ayrılır.
- **Gerçek Cihazlar:** Gerçek cihazlarda test, bize en doğru test sonuçlarını verir. Test faaliyetlerinin tüm

türleri, donanıma bağlı olanlar da dahil olmak üzere, kolayca yapılabilir. Kullanıcı deneyimini test etmek için kullanılabilir.

- **Bulut tabanlı test ortamları:** Mobil cihazlara web arayüzü yani tarayıcı ile erişilebilir. Test faaliyetlerinin tüm türleri kolayca yapılabilir. Var olan tüm mobil cihazlarda uygulamayı kısa sürede test etmek mümkündür. Bu sayede çok büyük kolaylık sağlar. Bize hafıza kullanımı, işlemci kullanımı da dahil olmak üzere birçok konuda ayrıntılı raporlar sunar. Hataları açıkça bildirir.

**Test Süreci:** Gereksinim Analizi, Tasarım, Kodlama, Test, Bakım




## Yazılım Testlerinde kullanılan yardımcı araçlar:

1) **Koçan(Stub) ve Sürücüler**
2) **Emülatorve Simülatörler**
3) **Test Verisi Üreteçleri**
4) **Hata Ayıklayıcılar(Debugger)**

**Test Durumları:** 3 bölümden oluşur. Giriş/Genel Bakış, Test Durumu Eylemleri, Sonuçlar

1) **Giriş/Genel Bakış:** Tanımlayıcı, Test durumunu yazan, Sürüm, Adı, Gereksinim Tanımlayıcısı, Amaç,

Bağlılıklar

2) **Test Durumu Eylemleri:** Test ortamı, ilklendirme, sonlandırma, eylemler
3) **Sonuçlar:** Beklenen sonuç, gerçek sonuç

**Hata Raporlama:** Hata, hatanın kaynağı, yapılan düzenleme raporlanmalıdır.

## Hata Önem Dereceleri

1) **Ölümcül:** Testin devam etmesini engelleyecek hataları belirtir. Bulunursa test devam ettirilmez.
2) **Kritik:** Test devam eder ama bu hata derecesiyle yazılım teslim edilemez.
3) **Büyük:** Test devam eder. Bu hatayla teslim edilebilir. Telafisi zor sonuçlar doğurabilir.
4) **Orta:** Test devam eder. Bu hatayla teslim edilebilir. Telafisi mümkün sonuçlar çıkabilir.
5) **Küçük:** Test devam eder. Bu hatayla teslim edilebilir. Hatalar önemli bir sonuç doğurmaz.
6) **Kozmetik:** Yazılım üzerindeki görsel (font, büyüklük, renk) hatalardır. Ürün teslimine etki etmez.

**Yazılım Test Riskleri:** Tümleştirmede karmaşa, Sıralama, Paralel Test işlemleri, Yüksek Maliyetli testler, testlerin plan dışı yürütülmesi, test yordamlarının yetersizliği

**Test Süreci:** Planlama > Tasarım > Gerçekleme > Hata raporlama > Test sonuç raporlama > Değerlendirme

**Gözden Geçirmeler:** Eş düşey gözden geçirme, resmi teknik gözden geçirme, birleşik gözden geçirme

**Eş düzey gözden geçirme:** Proje çalışanlarının genellikle aynı düzeyde bulunan personel ile birlikte yürüttükleri gözden geçirme. Tasarımcılar tasarımla ilgili, kodlayıcılar da kodla ilgili katılır.

**Birleşik gözden geçirme:** Geliştirici ile müşterinin, sözleşmede yer alan yönetsel ve teknik işleri, iş adımlarını, aşamaları gözden geçirmek için yapılan toplantı.

**Resmi teknik gözden geçirme:** 3 aşamalıdır. İnceleme, denetleme ve kod geçişleri. Her biri toplantı şeklinde yapılır.

**Denetleme(Audit):** Yazılım ürününün, bir yazılım sürecinin veya bir dizi yazılım süreç faaliyetlerinin belirtimler, standartlar, sözleşme veya diğer unsurlar bakımından uyumunun değerlendirilmesi için yapılan sistematik değerlendirmedir.

**İnceleme(Inspection):** Yazılım ürünündeki hatalar ve standartlardan sapmalara neden olan

anormalliklerin belirlenip tanımlanması için inceleme teknikleri konusunda eğitimli, tarafsız kişilerin rehberliğinde denk kişilerin katılımıyla gerçekleşir.

**Kod geçişleri(Walkthrough):** Yazılım geliştiricisi tarafından diğer geliştirme ekip üyelerine anlatılarak, yazılım ürününün iyileştirilmesine yönelik görüşlerin alınması ve standartların ihlali veya olası hataların belirlenmesidir.

## Resmi teknik gözden geçirme süreci:

Planlama > Bilgilendirme > Bireysel Hazırlık > Grup Toplantısı > Tekrar Çalışma > İzleme

**Yazılım Ürün Metrikleri:** Andaç (Token), Denetim Akış ve Bileşik metrikler küçük projelerde uygulama alanı bulurlar. Büyük projelerde ise sistem metrikleri büyük önem taşır.

**Sistem Metrikleri:** Süreç, Ürün nitelik, boyut, bakım ve okunabilirlik, zamanlama, üretkenlik, maliyet ve kaynak metrikleri

**Geleneksel Metrikler:** Kod Büyüklüğü (Lines of Code – LOC), Yorum Oranı (Comment Percentage – CP), Döngüsel Karmaşıklık(Cyclomatic Complexity – CC)

**Kod Büyüklüğü (Lines of Code – LOC):** Satır Sayısı, Yorum ve boşluk içermeyen satır sayısı, çalıştırılabilir yordam sayısı

**Satır Sayısı(Lines of Code – LOC):** Programın tüm satırlarının sayılması.

**Yorum ve Boşluk İçermeyen Satır Sayısı(Non-comment Non-blank – NCNB):** Programın yorum satırları ve boş satırlardan arındırılmış halidir.

**Çalıştırılabilir Yordam Sayısı(Executable Statements – EXEC):** Program içinde yer alan yordam sayısıdır.

**Yorum Oranı (Comment Percentage – CP):** Program için hazırlanmış yorum satırlarının, toplam programın yorum ve boşluk içermeyen satır sayısına bölümüdür.

## Döngüsel Karmaşıklık(Cyclomatic Complexity – CC): CC= YOLLAR – DÜĞÜMLER + 2

**Nesne Yönelimli Metrikler:** Chidamber ve Kemerer Metrikleri

1- Sınıf Başına Ağırlıklı Method (Weighted Methods per Class – WMC)
2- Kalıtım Ağacının Derinliği (Depth of Inheritance Tree – DIT)

3- Alt Sınıf Sayısı (Number of Children – NOC)
4- Nesneler Arası Eşleme (Coupling Between Object Classes – CBO) 5- Sınıf Yanıt Sayısı (Response for a Class – RFC)
7- Uyumsuzluk (Lack of Cohesion – LCOM)

### 1. Sınıf Başına Ağırlıklı Method (Weighted Methods per Class -WMC)

Döngüsel karmaşıklık değerleri toplamının sınıf sayısına bölümü ile bulunur. WMC değerinin 100'ün altında olması tercih edilmelidir. WMC sınıf karmaşıklığı hakkında fikir veren en önemli donelerden biridir. Bu metriğin faydaları sınıf başına ortalama metot sayısı gibi değerlendirilebilir. WMC değerinin 100'ün altında olması kabul edilebilir bir değerdir. WMC değeri NOM'dan farklı olarak sınıfın karmaşıklığı hakkında daha net bir fikir verir.

### 2. Kalıtım Ağacının Derinliği (Depth of Inheritance Tree – DIT)

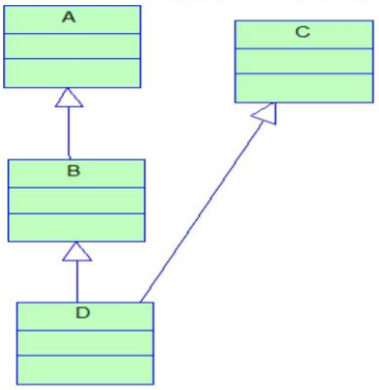
Sınıf hiyerarşisinin derinliği arttıkça daha fazla metot kalıtım alınır ve sınıfın davranışlarını öngörmek; anlamak zorlaşır.

Derin hiyerarşiler daha fazla metot ve sınıfı etkilendiğinden, dizaynı karmaşıklaştırır. Ancak hiyerarşi derinleştikçe kalıtım alınan metotların yeniden kullanım potansiyeli artar.

DIT için önerilen rakam genellikle 5'in altında olmasıdır. 5'in üzerindeki derinlikler oldukça karmaşık yapılar doğurabilir. DIT'in 0 olması sınıfın kök olduğunu gösterir. DIT'in ortalama 2-3 arası bir değerde olması yeniden kullanımın iyi seviyede olduğunu gösterir.2'den küçük derinlikler ise yeniden kullanımın zayıf olduğu alanları işaret eder.

## 2. KALITIM AĞACININ DERINLIĞI (DEPTH OF INHERITANCE TREE - DIT)

- DIT metriği sınıfın ebeveyn sınıflarının sayısını gösterir. Eğer çoklu kalıtım durumu söz konusu ise bu durumda hiyerarşideki en uzun yol kabul edilir. Örneğin yandaki şekilde yer alan D sınıfı için DIT değeri 2 olarak kabul edilir. DIT bize kısaca kaç tane ana sınıfın potansiyel olarak sınıfımızı etkileyebileceğini gösterir.




### 3. Alt Sınıf Sayısı ( Number of Children - NOC)

NOC metriği sınıftan türemiş alt sınıflarının sayısını verir.

Alt sınıf sayısı çoğaldıkça kalıtım özelliğine bağlı olarak yeniden kullanımın arttığı anlaşılır.

Alt sınıf sayısının çokluğu sınıfın hatalı soyutlama yapıldığını, belki de hatalı bir hiyerarşi kurulduğunu gösterebilir.

NOC sınıfın nüfus alanı hakkında bir fikir verir. NOC metriği yüksek sınıflar gözden geçirme, test gibi süreçlerin daha dikkatli ve uzun tutulması gereken yerlerdir.

### 4.Nesne sınıfları arasındaki bağımlılık (Coupling Between Object Classes - CBO)

Bir sınıf içindeki özellik ya da metotların diğer sınıfta kullanılması ve sınıflar arasında kalıtımın olmaması durumunda iki sınıf arasında bağımlılıktan bahsedilebilir. CBO; verimliliği ve yeniden kullanılabilirliği ölçmede kullanılır.

### 5. Sınıfın tetiklediği metot sayısı (Response for a class - RFC)

Bir sınıftan bir nesnenin metotları çağrılması durumunda, bu nesnenin tetikleyebileceği tüm metotların sayısı RFC değerini verir. Yani, bir sınıfta yazılan ve çağırılan toplam metot sayısıdır. Bu metrik; sınıf

seviye tasarım metriklerinden olup; anlaşılabilirliği, dayanıklılığı, karmaşıklığı ve test edilebilirliği ölçmede kullanılır.

6. Metotlardaki uyum eksikliği (Lack of cohesion in methods - LCOM)

LCOM, n adet kümenin kesişiminden oluşan kümelerdeki uyumsuzlukların sayısıdır ve metotlardaki benzerlik derecesini ölçer. Metotlardaki uyum eksikliği; bir sınıfın, iki veya daha fazla alt sınıfa ayrıldığını gösterir ve karmaşıklığı arttırır. Yapılan bir çalışmada, LCOM ölçütünün uyum özelliğini çok da iyi ayırt edemediği ispatlanmıştır. LCOM metriği test ediciye; verimlilik ve yeniden kullanılabilirlik derecesi hakkında bilgi verir.

## C & K METRİK KÜMESİ TABLO

| Kalite Özellikleri | WMC | DIT | NOC | CBO | RFC | LCOM |
|---|---|---|---|---|---|---|
| Verimlilik | | x | x | x | | x |
| Karmaşıklık | x | x | | | x | x |
| Anlaşılırlık | x | x | | | x | |
| Yeniden kullanılırlık | x | x | x | x | | x |
| Testedilebilirlik | | x | x | | x | |
| Dayanıklılık | x | | | | x | |

## ÖRNEK UYGULAMALAR

| Sistem Analizi | Java | C# | C++ |
|---|---|---|---|
| Sınıflar | 46 | 1000 | 1617 |
| Kod Sayısı | 50,000 | 300,000 | 500,000 |
| CBO | 2,48 | 1,25 | 2,09 |
| LCOM | 447,65 | 78,34 | 113,94 |
| RFC | 80,39 | 43,84 | 28,60 |
| NOC | 0,07 | 0,35 | 0,39 |
| DIT | 0,37 | 0,97 | 1,02 |
| WMC | 45,7 | 11,10 | 23,97 |
| Kalite | **Düşük** | **Yüksek** | **Orta** |

## ÖRNEK UYGULAMALAR (LCOM)

LCOM Example 1

```
public class A {
  private int _f1;
  private int _f2;
  private int _f3;
  private int _f4;

  public void method1() {          I1 = { _f1, _f2 }
    // uses _f1
    // uses _f2
  }
  public void method2() {          I2 = { _f2, _f3 }
    // uses _f2
    // uses _f3
  }
  public void method3() {          I3 = { _f3, _f4 }
    // uses _f3
    // uses _f4
  }
}
```

## METOTLARDAKİ UYUM EKSİKLİĞİ

- LCOM formül:
- $P = \{(I_i, I_j) \mid I_i \cap I_j = \varphi\}$ ve $Q = \{(I_i, I_j) \mid I_i \cap I_j \neq \varphi\}$ ise LCOM = $|P| - |Q|$ eğer $|P| > |Q|$
- değilse 0.

SLOC = Sources Lines of Code (Kod Satır Sayısı)

SLOC hesaplanırken yorum satırları göz ardı edilir

LSLOC = Logical Sources Lines of Code (Mantksal Kod Satır Sayısı)

LSLOC = print, for, while, if-else, vb. yapılar dikkate alınır

**Güvenlik Test Araçları**:

Nessus

Nmap

Zmap

Backtrack

Hping

**Nessus**: Güvenlik açığı tarama yazılımıdır. Genel amacı, bilgisayar sistemlerinde ve ağlarında potansiyel güvenlik açıklarını tespit etmektir. Çalışma prensibi istemci/sunucu biçimindedir.

**Nmap:** Gelişmiş bir güvenlik tarayıcısıdır. Taranan ağ üzerinde, bilgi sahibi olunmasında, ağ topolojisinin çıkarılmasında, sızma testlerinin gerçekleştirilmesinde, herhangi bir ağ hazırlanırken gerekli ayarların test edilmesinde, ağ envanterinin tutulması, haritalaması, bakım ve yönetiminde kullanılır.

**Hping:** İsteğe göre düzenlenmiş TCP,UDP,ICMP, Raw-ip paketler üretme, güvenlik duvarı işlevsellik ve performans testleri, DOS engellemesistemi, saldırı tespit ve engellemesistemleri, gelişmiş port tarama, dosya transferi, TCP/IP üzerinden hedef sistemlerden bilgi toplama

**Web Performans Test Araçları:**

GTmetrix GTmetrix, web sitelerinin:

Kod yapısını,

Yer alan resimleri,

Css ve js dosyalarını,

Sayfa boyutunu,

Sorgu sayısını,

Kodlama standartlarına uyumluluğunu,

Açılma hızı gibi arama motorlarındaki sıralaması ile doğrudan orantılı olan düzenlemeleri kontrol edip size rapor sunan bir servistir.

Pingdom Pingdom, sunucunuza bağlı tüm servisleri istediğiniz zaman aralıklarında takip eden ve sunucunun cevap vermediği durumlarda size e-mail ya da SMS ile uyarı gönderen bir servistir.

Google PageSpeed Google PageSpeed, sitenizin daha da hızlanması için site içinde küçük bir araştırma yapan ve size tavsiyelerde bulunan bir uygulamadır. Sitenizin hızını arttırmak için sunulan tavsiyeler arasında CSS ve Javascript dosyalarını sıkıştırma önerileri yer alır. Sitenizin hem SEO bazında, hem

kullanıcı tarafında daha da iyi hizmet vermesini sağlayan uygulama; tarafınıza oldukça gelişmiş ve detaylı bir rapor sunar.

Search Engine Optimization-SEO, arama motorlarının web sayfalarını daha kolay bir şekilde taramasına olanak sağlayan tekniktir

YOTTAA Yottaa'yı diğerlerinden ayıran özellik, aynı anda birçok farklı konumdan sorgulama yapabilmesidir.

SimplyTestable

PageScoring

**Web Güvenliği Test Araçları:**

Acunetix= Dinamik oluşturulan web uygulamalarındaki güvenlik açıklarını tarayarak, çok detaylı analiz yapılmasını ve gelişmiş raporlar alınmasını sağlayan araçtır

Netsparker= Netsparker bir web uygulaması güvenlik tarayıcısıdır. Otomatik olarak bir web sitesini uygulama seviyesindeki güvenlik açıklarına karşı analiz edip güvenlik açıklarını raporlar.

Sucuri= sitenizin iframe virüsü, malware ve kullanıcılar için tehdit oluşturabilecek diğer zararlı kodları içerip içermediğini, kara listelere girip girmediğini tek tıklamayla kontrol etmenize olanak sağlayan güzide güvenlik servislerinden biridir.

Havij= Havij bir web sitedeki Sql injection açıklarını bulmaya yardım eden bir SQL injection otomasyonudur. Web uygulamasının savunmasızlığından faydalanır.

Sqlmap= Saldırı tespiti, korunmasızlık sömürücü ve Sql enjeksiyon açıklarını bulma gibi işlemlerde otomatik olarak işlem yapan çok güçlü bir araçtır.

**PenetrasyonTestAraçları:**

Metasploit= Metasploit içinde pek çok hazır exploit bulunduran aynı zamanda bu exploitleri yerel ve uzak erişim için yapılandırabilen özel bir penetrasyon test yazılımıdır.

Canvas

Armitage

CoreImpact

**Mobil Test Araçları:**

Crittercism= Crittercism, mobil uygulama performansınızı her açıdan değerlendirebilmenize olanak sağlıyor. Bununlar birlikte Crittercism anlık olarak iOS, Android, Windows Phone ve Html5 platformlarındaki uygulamalarınızın hatalarını görmenize olanak sağlayarak detaylı raporlama sunar.

AppThwack= Mobil uygulamaları 100'den fazla gerçek cihazla test etmeye yarayan bulut tabanlı bir test aracıdır. Uygulamanın hatalarını, cihazın hafıza ve işlemci kullanımını ve benzeri birçok özelliği her bir cihaz için ayrıntılı rapor halinde sunar.

Pulse.io= Mobil uygulamaların test aşamasında gözden kaçan hatalarını yakalamak ve kod performansı konusunda ölçümleme ve analiz yapan bir servistir. Uygulamadaki anlık oturumları analiz ederek o andaki sorunlar ve ileriye dönük oluşabilecek hatalar konusunda raporlar çıkartan servis dönüşüm, verimlilik konusunda önemli ölçütleri geliştiriciyle paylaşmaktadır.

Instabug= Instabug'ın SDK'sını uygulamanıza entegre ettiğiniz andan itibaren kullanıcılar uygulamanın açık olduğu anda telefonlarını sağ ve sola sallayarak uygulamanız hakkında size geri bildirimde bulunabiliyorlar.

Bugsense= BugSense mobil uygulama geliştiricilerin tercih edebileceği bir diğer uygulama performansını ölçme (application performance management) araçlarından bir tanesidir. Mobil uygulamaların hata loglarını size bildirerek güncelleme yapmanızı sağlamaktadır.

**Kullanılabilirlik Test Araçları:**

YouEye= Uzaktan kullanıcı testi yapabildiğiniz YouEye'da kullanıcıların nereye baktıklarını takip ederek göz izleme testi yapabilirsiniz. Aynı zamanda kullanıcı ses kaydı da bulunuyor

Userzoom= iPhone, iPad, ve Android telefon ve tabletler için mobil websiteleri ve prototiplerini kullanılabilirlik yönünden test etmemizi sağlayan bir araçtır.

Optimizely= Bağlılık, tıklama, dönüşüm ve kayıt gibi aşama ve kıstaslarda hangi yöntemin daha iyi işe yaradığının ölçmeye yarayan bir araçtır. Hedef takibi ile web sitenin kullanıcı deneyimi açısından daha iyiye ulaşması için A/B testlerinde kullanılır

SessionCam= Kullanıcıların hareketlerini izlemeye yarayan bir kullanılabilirlik test aracı olarak Session Cam, sıcaklık haritası, tıklama haritası, form analizi ve dönüşüm hunisi oluşturmayı sağlıyor.

TrymyUI= Kullanıcıların demografik olarak profillerini seçerek kullanılabilirlik testi uygulanabilir. Ekran ve mouse hareketleri, sesli düşünme testleri ve yazılı geri bildirimler dahil her kullanıcının verileri video halinde sunulur.

Userlytics= Web site, mobil - kullanıcı etkileşimini ölçmek amacıyla uzaktan kullanılabilirlik testi uygulamayı sağlar. Kullanıcı deneyimini pozitifleştirmek için raporlar halinde sunulan veriler kullanıcı kayıtlarından elde edilir

**Pingdom**: Sunucudaki servislerin takip edilmesi (SMS ve E-Mail bildirimi)

**Sucuri**: Sitenin iframe virüsü, malware vb kodları tarar.

**Crittercism:** iOS, Android, Windows Phone, Html5 hatalarını raporlamaya sağlar.

**AppThwack:** Bulut tabanlı mobil test aracı.

**Bugsense:** Mobil uygulama performans ölçmeye yarar. iOS, Android, Windows Phone, Html5 desteği sunar.

**Youeye:** Göz izleme testi yapmaya yarar.

**SessionCam:** Kullanıcı hareketlerini izlemeye yarar. Sıcaklık ve tıklama haritası çıkartır.

**Optimizely:** A/B testleri yapar.

**Userzoom:** İphone, iPad, Android telefon&tablet için mobil website ve prototiplerini test etmeye yarar.