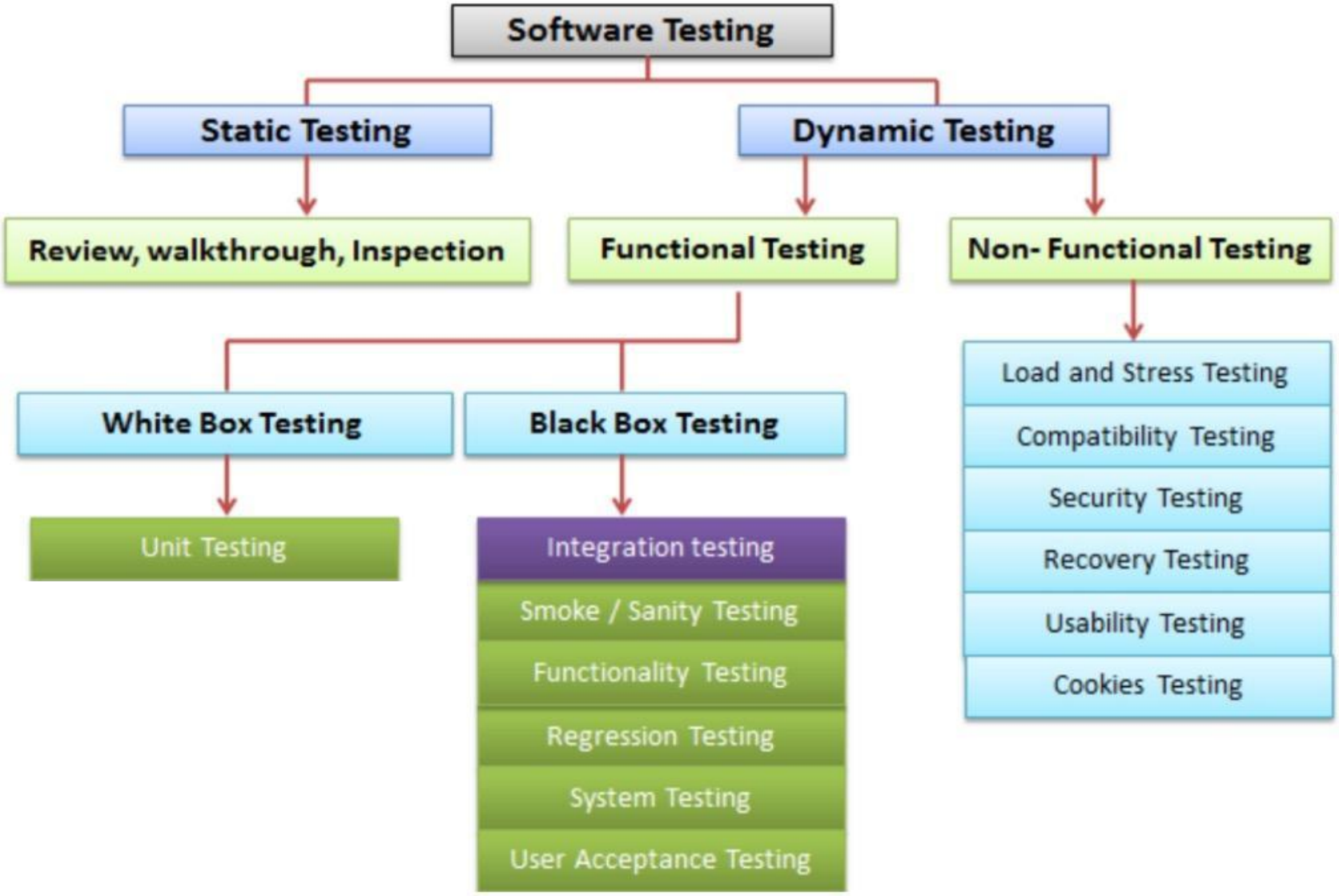
TECHPROEDUCATION

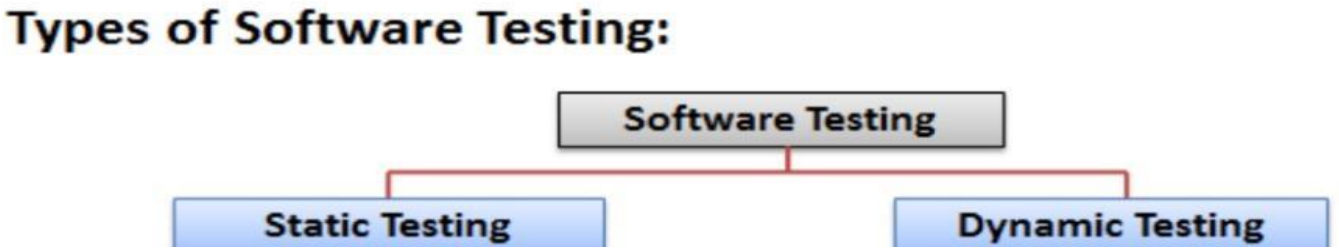
SOFTWARE TEST ENGINEERING COURSE

MANUAL TESTING

TEST ÇEŞİTLERİ

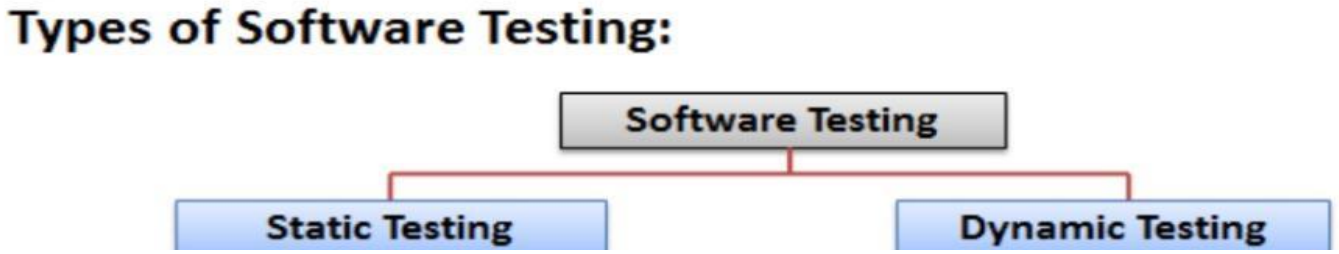


TEST ÇEŞİTLERİ



Statik test kod yürütülmeden kodun veya diğer proje dokümanlarının manual olarak gözden geçirilmesidir. **Statik** testler dinamik testlere geçilmeden önce yapılmalıdır. Projenin başlarında gözden geçirme yoluyla tespit edilen hataların çözülmesi ilerleyen aşamalarda bulunmasından daha az maliyetlidir.

TEST ÇEŞİTLERİ



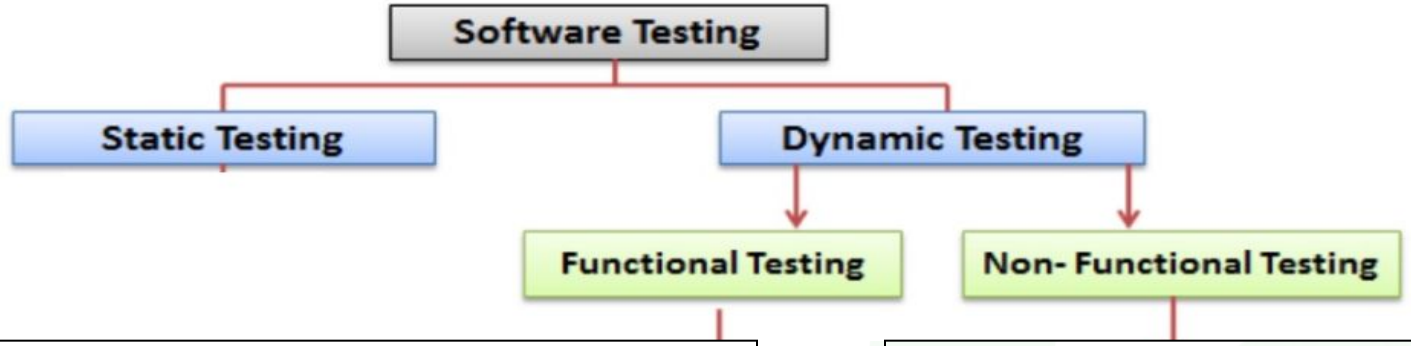
Dinamik Test: Kodun bütününü çeşitli yöntemlerle **test** etmeye yarayan metottur. Bulunan tüm hatalar çözülmeden ve **testin** sonlandırma kriterleri sağlanmadan sona ermez. **Test** edilecek yazılımın türüne göre, uygulanma metotları farklılık gösterebilir.

Verification - Validation

Verification :
Döküman üzerinde yapılan testte (Static Testte), hazırlıklar düzgün yapılmış mı yapılmamış mı, bunun doğrulanması işlevine denir.

Validation :
kodu çalıştırdıktan sonra (Dinamik Testte) ortaya çıkan sonuçta beklenen netice alınıyor mu, alınmıyor mu, bunun doğrulanması işlevine denir)

TEST ÇEŞİTLERİ



Fonksiyonel Test, testini gerçekleştirdiğimiz uygulamanın her bir fonksiyonunun verilen gereksinimlere uygun olarak çalışıp çalışmadığını doğrulayan test türüdür. Fonksiyonel test Black Box test altında kullanıldığı için uygulamanın kaynak kodu ile ilgili değildir. Bu testi gerçekleştirirken odak noktası daima uygulamanın ana işlevlerinin kullanıcı dostu olmasıdır.

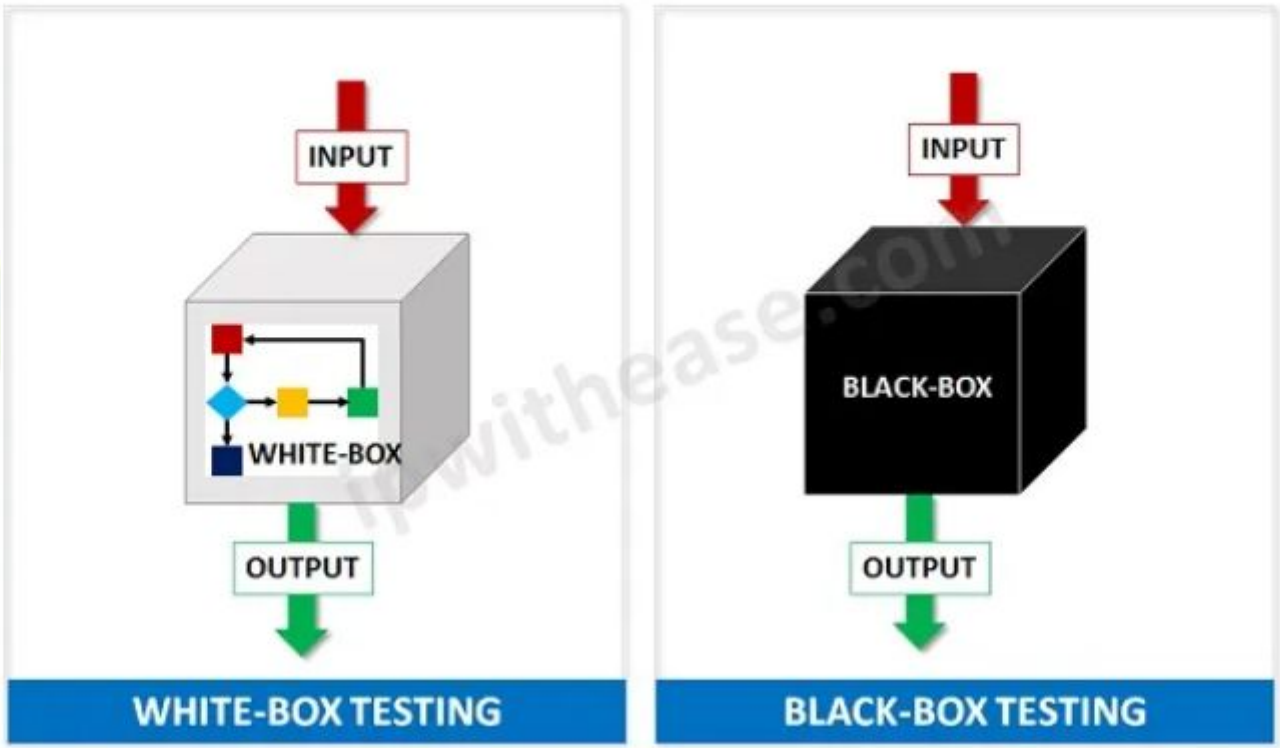
Uygulamanın işlevsel olmayan özelliklerinin test edildiği bir test türüdür. Amaç sistemin hazır olup olmadığını belirlemektir. Bileşenlerin veya sistemin kalite özellikleri test edilir. Yazılımın kalitesinde ve doğru çalışmasında işlevsel testler kadar önemlidir. Örneğin sistemi aynı anda kaç kullanıcı kullanabilir sistem yeterince güvenli midir? Bu gibi soruların karşılığını almak için sistem test edilir.

TEST ÇEŞİTLERİ

Functional Testing

White Box Testing | Black Box Testing

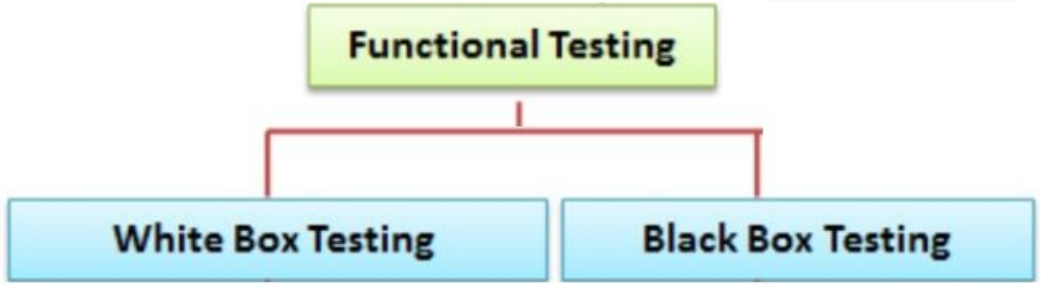
WHIE-BOX TESTING vs BLACK-BOX TESTING



Unit Testing

- Integration testing
- Smoke / Sanity Testing
- Functionality Testing
- Regression Testing
- System Testing
- User Acceptance Testing

TEST ÇEŞİTLERİ



Beyaz kutu testinde, kodun içine girilerek kodun doğruluğu ve kalitesi test edilir. Bu test türünde kod erişimi zorunludur. Kod yapısı ve tasarımına yönelik testler gerçekleştirilir. Örneğin, gereksiz bir kod bloğu tespit edilebilir veya kodun okunabilirliğini arttırmaya yönelik durumlar tespit edilebilir. Kodda erken bulunacak hatalar Kara Kutu(Black Box) testlerini de kolaylaştırmaktadır. Beyaz kutu testleri çoğunlukla geliştiriciler tarafından uygulanır.

Kara kutu testleri; kodun yapısı(structure), tasarımı (design) ve uygulanışı(implementation) ile ilgilenmez. Kara kutu testlerinde girdi ve çıktı değişimine göre sistemin nasıl çalıştığı test edilir. Kara kutu test çeşitleri çoğu yazılım test uzmanı tarafından yaygın olarak kullanılan test çeşitleridir.

Kara kutu test teknikleri:

1- Equivalence Partitioning (Eşit Bölümlere Ayırma Tekniği)
2- Boundary Value Analysis (Sınır Değerleri Analizi)
3- Decision Table (Karar Tablosu)
4- State Transition Table (Durum Geçiş Tablosu)
5- Use Case Testing (Kullanım Durumları Testi)

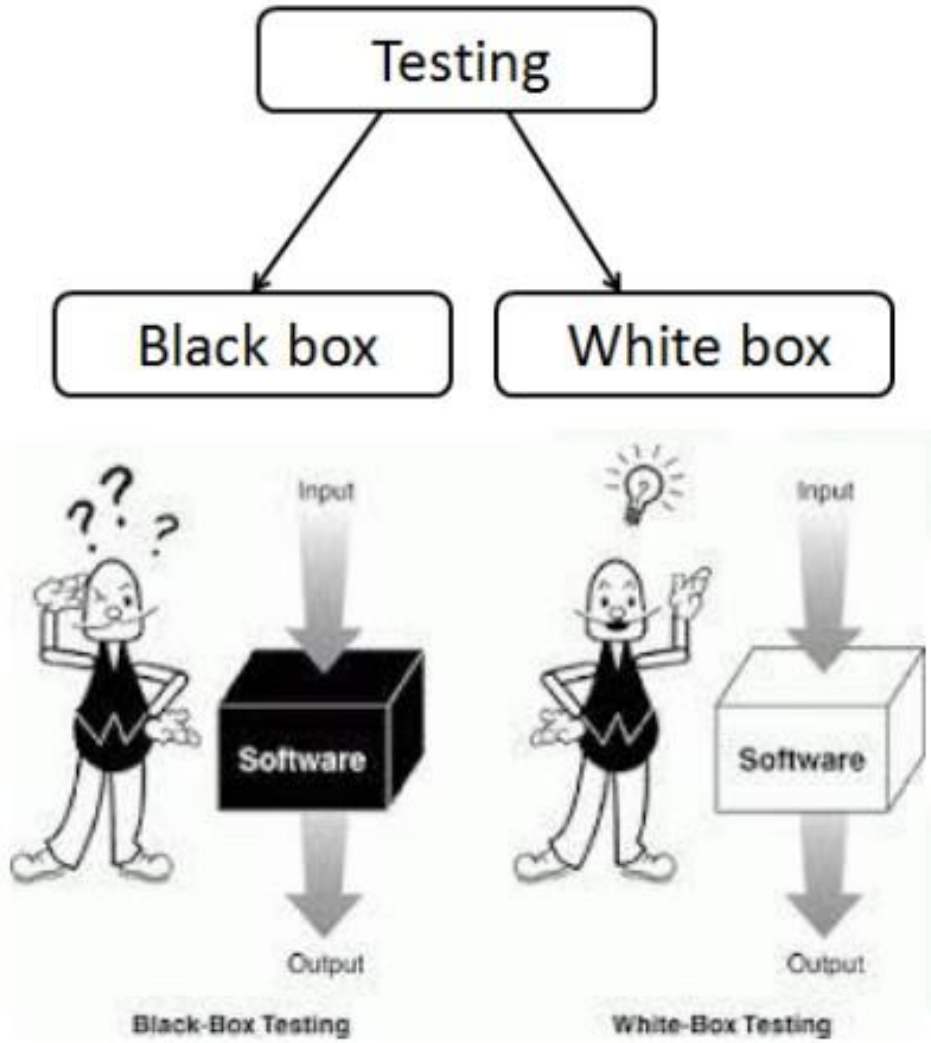
White box avantaj ve dezavantajlar

Beyaz Kutu Testlerinin Avantajları

- Testler, yazılım geliştirilirken tasarlanıp yapıldığı için erken sürelerde müdahale edilebilmektedir.
- Geliştiriciler kendi kod parçacıklarını defalarca gözden geçirdikleri için kodun güvenilirliği yüksektir.
- Kod optimizasyonu fazladır.

Beyaz Kutu Testlerinin Dezavantajları

- Kod okumayı bilen nitelikli bir yazılım test uzmanı bulmak zor ve maliyetlidir.
- Geliştirici kendi testini kendi yaptığı için gözden kaçabilen noktalar olabilir.
- Testler çok kapsamlı ve ayrıntılı olduğu için uzun zaman alabilmektedir



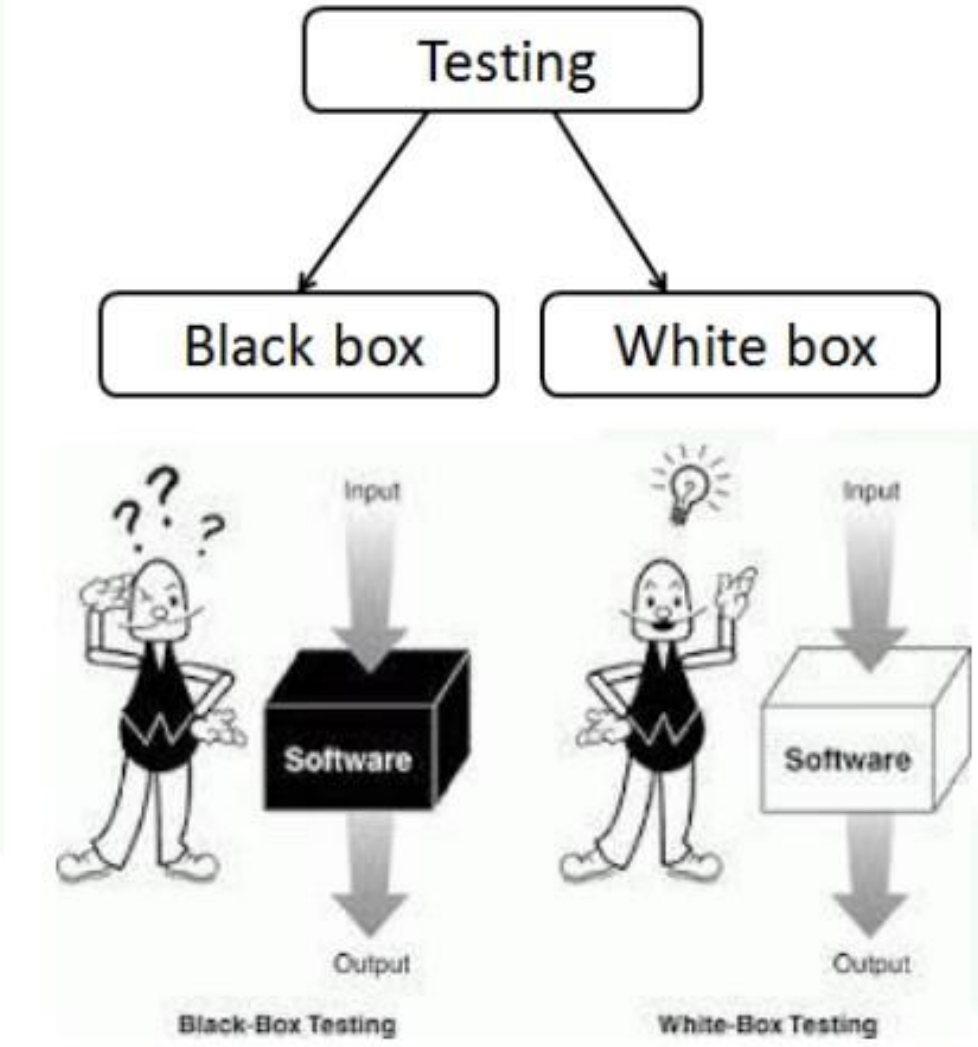
Black box avantaj ve dezavantajlar

Kara Kutu Testlerinin Avantajları

- Test uzmanları tarafından yapıldığı için kod bilgisine ihtiyaç duyulmaz.
- Kodu geliştiren kişi ve test eden kişi farklı olduğu için farklı bakış açılarıyla test edilebilir ve görünmeyen hatalar bile kolaylıkla bulunabilir.
- Testler hızlı ve etkin bir biçimde uygulanabilir.
- Gereksinimlerin belirlenmesinin hemen ardından test senaryoları oluşturulabilir.
- Büyük ölçekli sistemlerde bu yöntem kullanıldığında oldukça yüksek verim alınır.

Kara Kutu Testlerinin Dezavantajları

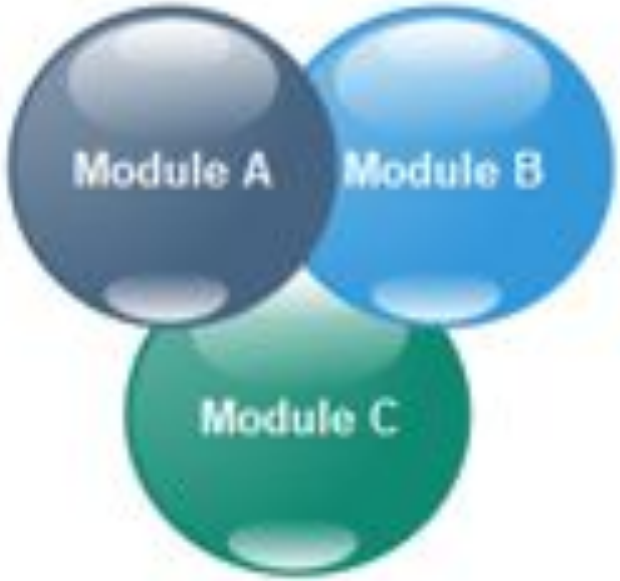
- Sistemin iç yapısı bilinmediği için kaynak kod içerisinde kümelenmiş hataların bulunması zorlaşır.
- Halihazırda geliştiricinin kendi yaptığı testler ile tekrara düşülebilir.
- Karmaşık kod blokları içeren yazılımlarda kullanılamaz.
- Bütün olasılıkları kontrol etmek mümkün değildir bu sebeple test edilmemiş fonksiyonlar olabilir.



UNIT TEST (BİRİM TESTİ)- whitebox

Birimin fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması işlemidir
Unit testing code temelli developerlar tarafından yapılan testlerdir. Her birimi ayrı ayrı test etmek için yapılır.
Bu test genellikle küçük code birimleri için yapılır ve classdan daha büyük değildir.

Tested in Unit Testing

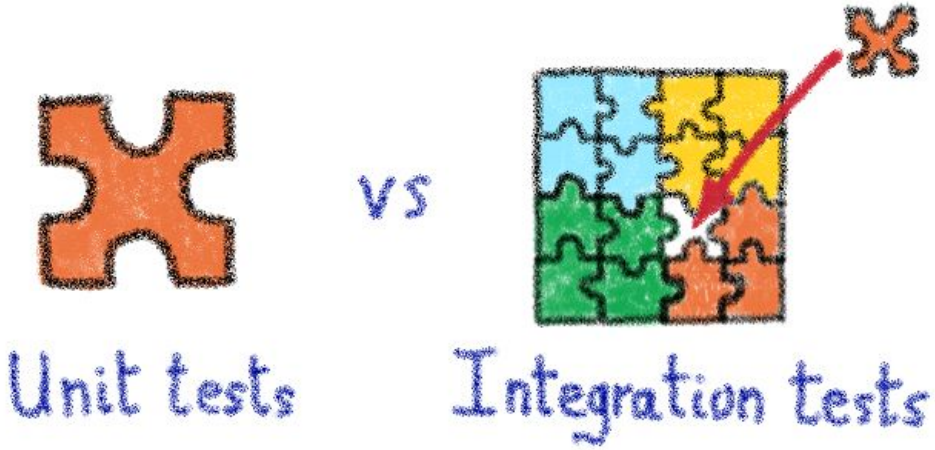


Under Integration Testing

INTEGRATION TEST(Entegrasyon Testi)

- Her modülü test ettikten sonra, entegre modüllerin istenen çıktıyı verdiğinden emin olmak için bir araya getirilir ve test edilir. Tipik olarak her yazılım, farklı yazılım geliştiriciler tarafından kodlanmış farklı yazılım modüllerinden oluşur. Bu test her bir modülün herhangi bir kusur olmadan mükemmel bir şekilde etkileşime girmesini sağlamayı amaçlar. Bu modüllerin verileri arasındaki iletişime odaklanır. Entegrasyon testi, dizi testi ve iş parçacığı testi olarak da bilinir.

INTEGRATION TEST (Entegrasyon Testi)



Birim testi sırasında gözden kaçmış olabilecek kusurları, birim testi yapıldıktan sonra istemcilerin gereksinimlerinde meydana gelen değişiklikler gibi faktörlerden kaynaklanabilecek kusurları, harici donanım arayüzlerinden kaynaklanan hataları, yazılımlar arasındaki etkileşimlerden kaynaklanan hataları ortadan kaldırmak için entegrasyon testi gereklidir.

SMOKE TEST(Duman Testi)

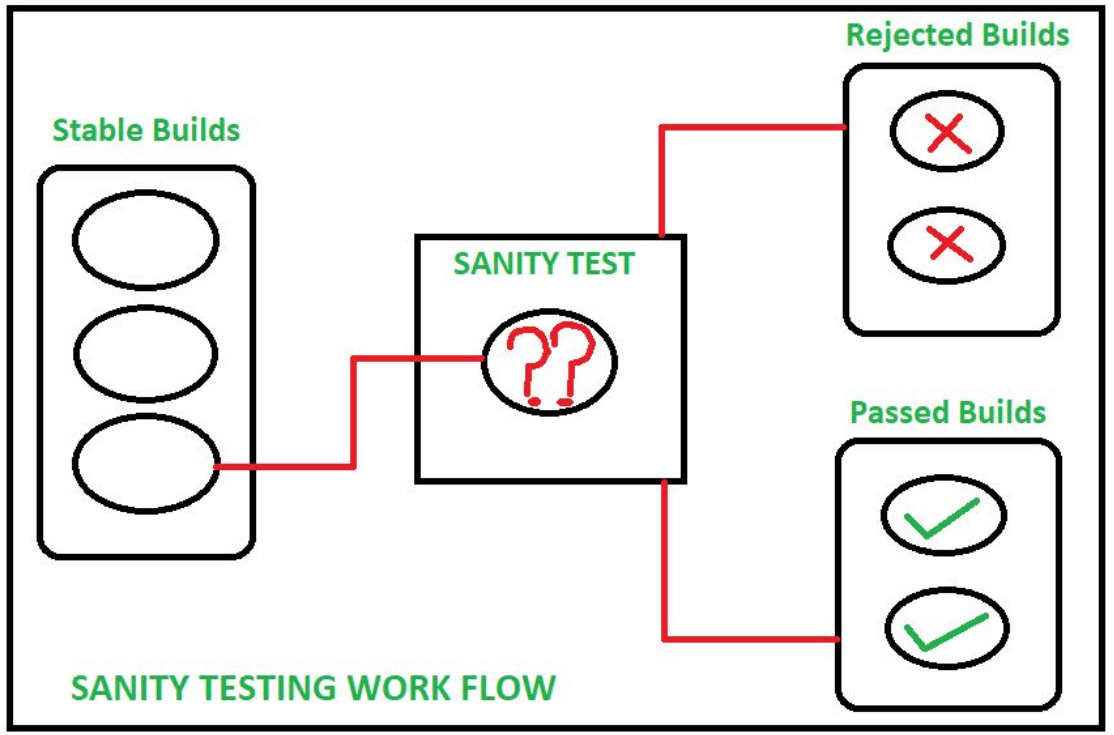
Duman testi projede en önemli işlevlerin çalışmasını sağlamayı amaçlayan kapsamlı olmayan bir test türüdür.
Temel sistemin aynı kalması için tüm sistem birlikte test edilmelidir.
Bu teste önemli olan büyük resmi görmektir.

SMOKE TEST(Duman Testi)

Amaç uygulamanın teste devam edecek düzeye gelip gelmediğini kontrol etmektir. Duman testlerini evreleme ve üretim ortamlarında erken ve sık sık tekrarlamak gerekmektedir.

SANITY TEST

Uygulama üzerinde küçük bir hata giderildiğinde veya küçük değişiklikler ardından yapılan testlerdir. Sanity testi test sürecinde uygulamanın bir iki işlevine odaklanırken duman testi tüm önemli işlevlerin çalıştığından emin olmak için yapılır. Bu test sayesinde hatalar daha erken bir aşamada keşfedilir.

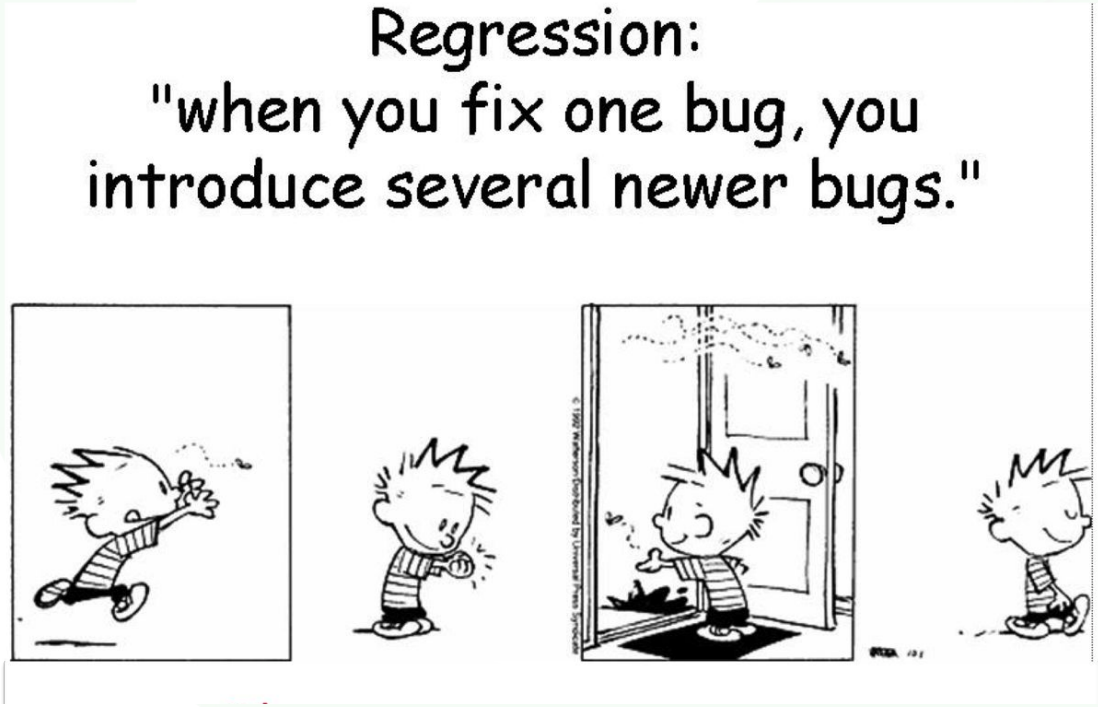


REGRESSION TEST(Regresyon Testi)

Bu test türünde yazılım üzerinde yapılan değişiklik sonucunda uygulama kodunun bozulup bozulmadığını doğrulamak için yapılan test türüdür. Değişiklik sonrasında mevcut özelliklerin çalışıp çalışmadığını kontrol etmek için yapılan test türüdür.

1. Kısmi Regresyon Kısmi Regresyon, kodda değişiklikler yapıldığında ve bu birimin değişmemiş veya halihazırda mevcut olan kodla entegre edildiğinde bile kodun iyi çalıştığını doğrulamak için yapılır.

1. Tam Regresyon Tam Regrasyon, kodda bir dizi modülde bir değişiklik yapıldığında ve ayrıca başka herhangi bir modüldeki bir değişikliğin etkisinin belirsiz olması durumunda yapılır. Değiştirilen kod nedeniyle herhangi bir değişikliği kontrol etmek için ürün bir bütün olarak test edilir



Sistem Testi (E2E – END TO END Test)

- Sistem Testi, yazılım gereksinim testlerinin tamamlanmasından sonra, sistem gereksinimlerine göre oluşturulan testleri kapsar.
- Yazılım tarafında yapılan Birim Testi(Unit Test) ve Entegrasyon Testi(Integration Test) adımlarından sonra yapılan sistem testleri, daha çok işlevi tamamlanan yazılımın, güvenlik, güvenilirlik, performans gibi faktörler altında yapılan test işlemlerini kapsar.
- Sistem testinin amacı, bir uygulamanın tasarlandığı gibi işlevleri gerçekleştirirken doğruluğunu ve eksiksizliğini doğrulamaktır.
- Gerçek sonuçlar ve beklenen sonuçlar sıraya girdiğinde veya farklılıklar, müşteri girdisine göre açıklanabilir veya kabul edilebilir olduğunda sistem testi tamamlanmış olarak kabul edilir.

Sistem Testi (E2E – END TO END Test)

- İki şekilde ifade edilebilir
  - ❑ Farklı fonksiyonların birleştiği uçtan uca senaryolar
  - ❑ Aynı modülün backend ve ui tarafında ele alındığı senaryolar

Ad-hoc Testing (Gecici Test –Maymun Testi)

- Rastgele Test veya Maymun Testi olarak da bilinir, herhangi bir planlama ve belge olmadan bir yazılım testi yöntemidir.

- Testler herhangi bir resmi prosedür veya beklenen sonuç olmadan gayri resmi ve rastgele yapılır.

User Acceptance Testing UAT (Kullanıcı Kabul Testi)

Bu testtin amacı yazılımı iş gereksinimlerine göre doğrulamaktır. Amaç yazılımın kullanıcı ve müşteri tarafından kabul edilip edilmeyeceğini belirlemek için test edilmesine olanak sağlamaktır. **Bu doğrulamalar iş gereksinimlerine aşina olan son kullanıcılar tarafından gerçekleştirilir.** Fonksiyonel, sistem ve regresyon testleri gerçekleştirildikten sonra kullanıcı kabul testleri gerçekleştirilir. Yazılım yayınlanmadan gerçekleştirilen son testtir.

Beta