SOFTWARE TESTING LIFE CYCLE

(Yazılım Testi Yaşam Döngüsü)

1. Ders

25.06.2022

Bu derste neler öğreneceğiz?

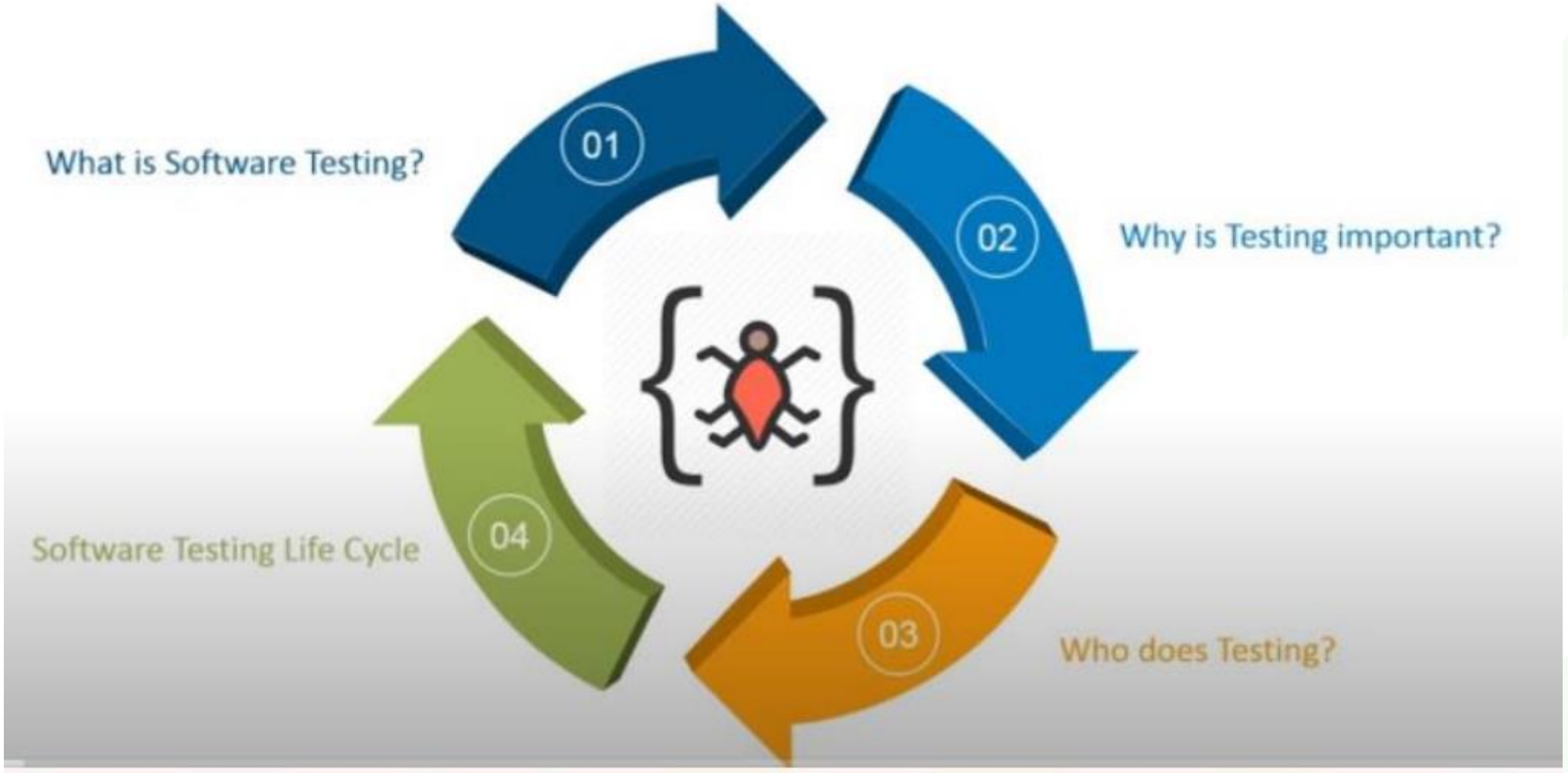
1- STLC NEDİR?

2- STLC AŞAMALARI NELERDİR?

3- TEST ÇEŞİTLERİ NELERDİR?

4- BUG LIFE CYCLE

STLC


What is Software Testing?
01
02
Why is Testing important?
Software Testing Life Cycle
04
03
Who does Testing?

Yazılım Testi Niçin Önemlidir?

- Yazılımın güvenilirliğini kontrol etmek için yazılım testi gereklidir. Yazılım testi olmadan yazılım istenen kaliteye ulaşamaz
- Yazılım testi, sistemin arızaya neden olabilecek herhangi bir hata içermemesini sağlar. Yazılım testi, ürünün müşterinin gereksinimine uygun olmasını sağlar
- Sonunda, yazılım, hepsi farklı bakış açılarına ve yaklaşımlara sahip bir Developer ekibi tarafından geliştirilir. En zeki insan bile hata yapma eğilimindedir. Sıfır hata ile yazılım oluşturmak mümkün değildir dolayisiyla Testing yaşam döngüsü yazilim geliştirme döngüsüne muhakkak dahil edilir.
- En iyi sonuçlar için yazılım testi ve kodlamanın iç içe gitmesi önemlidir.
- Tüm test faaliyetleri planlama gerektirir.
- Test, yazılımın tam olarak gereksinimlerde belirtildiği gibi davranmasını ve uygulanmasini sağlar.
- İhtiyaca(requirement) uygun olmayan her şey bir hatadir. (defect,BUG).

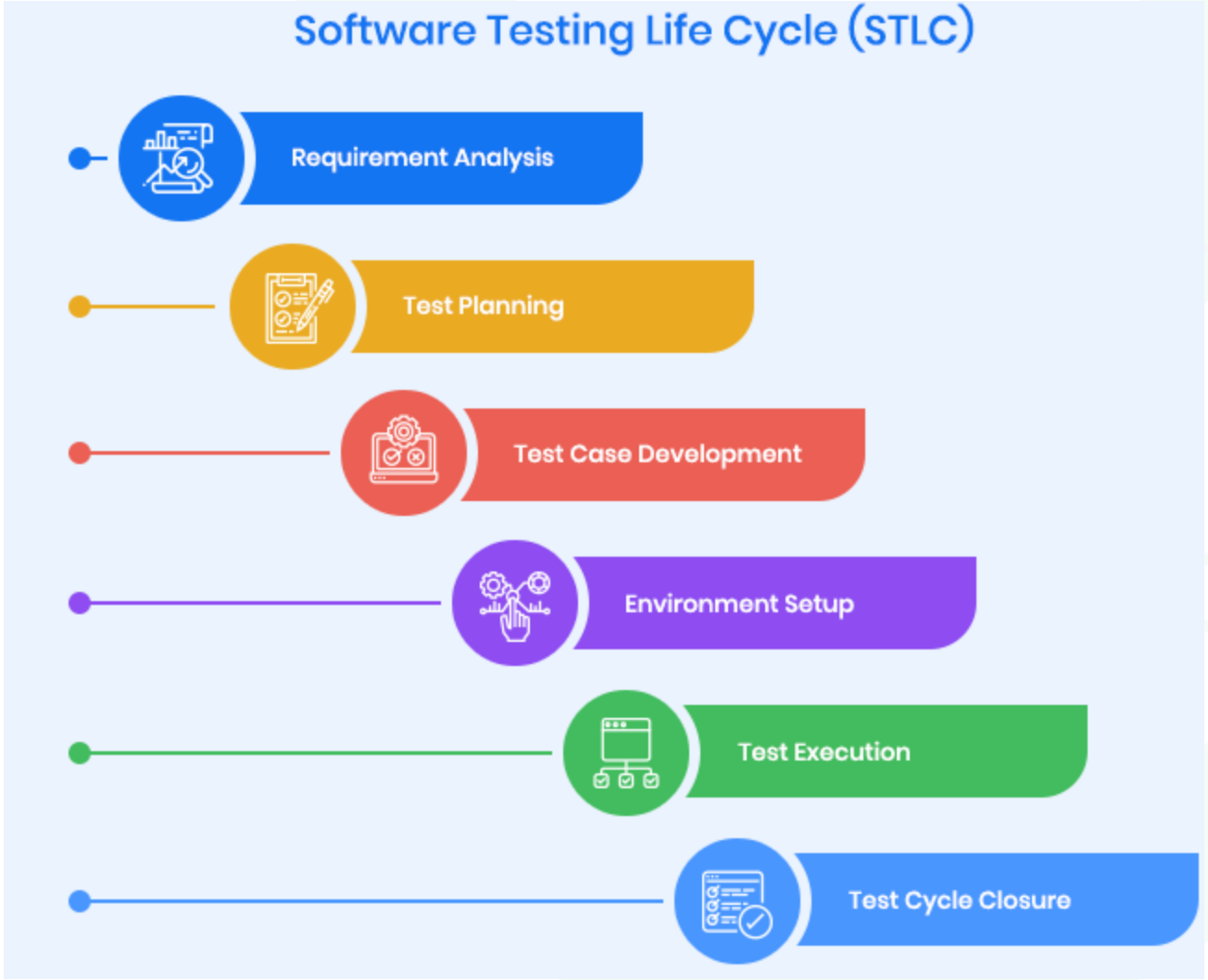
Yazılım Testi Niçin Önemlidir?

Yazılım Kalite Guvencesi, tüm yazılım geliştirme sürecini içerir - sürecin izlenmesi ve iyileştirilmesi, üzerinde anlaşılan süreçlerin, standartların ve prosedürlerin takip edilmesini, sorunların bulunmasını ve ele alınmasını sağlar.

Kalite kontrol (Quality Assurance – QA), bir sürecin kalite etkinliğini azaltacak durumlara karşı önlem alarak kaliteye hakim olma anlamına gelir.

STLC Aşamaları

- Gereksinimlerin Analizi
- Test Planının Hazırlanması
- Test Case Hazırlanması
- Test Ortamlarının Hazırlanması
- Testin Koşulması
- Test İşleminin Tamamlanması



1- Requirements Analysis (Gereksinimlerin Analizi)

Sadece user story de açıklanan yazılım gereksinimlerini gözden geçirirsiniz.

İçecek içmek istiyorum. (User Story – Kullanıcı Hikayesi)

Kabul Kriterleri

- Sunuma önem veren bir insan olarak icecegimin guzel gorunmesi icin semsiye konulmasini istiyorum
- Sicaktan bunaldigim icin suyumla birlikte beni ferahlatacak 1 dilim limon istiyorum
- Cok susadigim icin buyuk bir bardak içecek istiyorum
- Hava sicak oldugu icin içeceğimin soguk olmasini istiyorum
- Kolay içebilmek icin bir pipet istiyorum

2- Test Planning (Test Planlamasi)

Neyin test edilmesi gerektiğine dair genel bir fikir topladıktan sonra, testler için 'plan yapılır’.

Test Planı belgesi, Ürün Açıklaması, Yazılım Gereksinimi Spesifikasyonu (Software Requirement Specification SRS) veya Kullanım Senaryosu Belgelerinden (Use Case Documents) türetilmiştir.

Amaçlanan test faaliyetlerinin kapsamını, yaklaşımını, kaynaklarını ve programını açıklayan bir belgedir.

Test Planı belgesi genellikle Test lead veya Test Manager tarafından hazırlanır ve belgenin odak noktası neyin test edileceğini, nasıl test edileceğini, ne zaman test edileceğini ve hangi testi kimin yapacağını açıklamaktır

3-TEST CASE DEVELOPMENT (TEST KILIFI/SENARYOSU OLUSTURMA)

Test case'ler gereksinimlere göre hazırlanan input'lar, olaylar ya da aksiyonlar ve bunlar sonucu oluşması beklenen sonuçların belirtildiği dokümanlardır.

What is a test case in software testing? (Yazılım testinde test kılıfı nedir?)

- En basit biçimde, bir test case, bir test yazılımının gereksinimleri karşılayıp karşılamadığını ve işlevlerini doğru bir şekilde yerine getirip getirmediğini belirlediği bir dizi koşul veya değişkendir.
- Test case, bir tester' ın gerçekleştirdiği tek bir yürütülebilir testtir. Tek tek aşamalar (step) takip edilerek yapilir.
- Bir test case, bir şeyin davranması gerektiği gibi davrandığını doğrulamak için bir dizi adım talimat olarak düşünebilirsiniz.

3-TEST CASE DEVELOPMENT (TEST KILIFI/SENARYOSU OLUSTURMA)

Bir test senaryosu genellikle şunları içerir

- **Test Case No:** Test Numarası (Kesin Olmalıdır)
- **Priority** (Öncelik)
- **Test Name:** Test Senaryosu Adı (Kesin Olmalıdır)
- **Test Step:** (Test Adımı, Adımları (Kesin Olmalıdır)
- **Result:** Beklenen Sonuç (Kesin Olmalıdır)
- **Status:** Durumu, Done/Undone - Pass/Fail (Kesin Olmalıdır)
- **Test Datası**
- **BUG** (Hata) Sayısı yazılır
- **Notes** (Detay ve Notlar)
- **Sprint No** (Agile & Scrum Çalışmasında Kullanılır)

3-TEST CASE DEVELOPMENT (TEST KILIFI/SENARYOSU OLUSTURMA)

Ornek Test Case

A	B	C
User Story ID	Description	Acceptance Criteria
US 0002	Facebook login (Giris sayfasi) gecersiz kimlik bilgileriyle erisilmemelidir	Gecersiz kullanıcı adı ile erişim sağlanamaz
		Geçersiz şifre ile erişim sağlanamaz
		Geçersiz kullanıcı adı ve şifre ile erişim sağlanamaz

3-TEST CASE DEVELOPMENT (TEST KILIFI/SENARYOSU OLUSTURMA)

Ornek Test Case

A	B	C	D	E	F	G	H	I
User Story ID	Test Case ID	Test Objective	Pre-Condition	Steps	Test Data	Expected Result	Actual Result	Status
US 0002	TC_001	Gecersiz kullanıcı adı ile erişim sağlanamaz	Login erisilebilir olmalidir //www.facebook.com/ da	1_ https://www.facebook.com/ gidi 2_kullanici adi textbox a tiklayiniz 3_yanlis bir kullanici adini giriniz 4_sifre textbox ina tiklayiniz 5_dogru bir kullanici sifresi giriniz 6_login butonuna tiklayiniz	URL 3: https://www.facebook.com kullanici adi ="pes_etmek_yok" sifre ="yapabilirim"	kullanici erisim elde edememelidir	kullanici erisim elde edemedi	Pass
US 0002	TC_002	Geçersiz şifre ile erişim sağlanamaz	Login erisilebilir olmalidir //www.facebook.com/ da	1_ https://www.facebook.com/ gidi 2_ kullanici adi textbox a tiklayiniz 3_dogru kullanici adini giriniz 4_sifre textbox ina tiklayiniz 5_yanlis bir kullanici sifresi giriniz 6_login butonuna tiklayiniz	URL 3: https://www.facebook.com kullanici adi = techproedusa@gmail.com sifresi = "yanlis_sifre"	kullanici sifre hatasi elde etmelidir ve giris izni verilmemelidirr	Email adresi yanlis veerisim elde edilemedi	Fail
US 0003	TC_003	Geçersiz kullanıcı adı ve şifre ile erişim sağlanamaz	Login erisilebilir olmalidir //www.facebook.com/ da	1_ https://www.facebook.com/ gidi 2_ kullanici adi textbox a tiklayiniz 3_yanlis kullanici adi giriniz 4_sifre textbox ina tiklayiniz 5_yanlis sifre giriniz 6_login butonuna tiklayiniz	URL 3: https://www.facebook.com username ="yanlis_username" password ="yanlis_sifre"	kullanici erisim elde edememelidir	kullanici erisim elde edemedi	Pass

4- Test Environment Setup (Test Ortami Kurulumu / Olusturulmasi)

Testinizi nerede ve hangi ortamda gerçekleştireceğinizi bilmelisiniz ve test verilerinize de sahip olmalısınız!

Dev Environment (Development)
Test Environment(Test ortami)
Stage Environment (sahne)
Prod – Canlı (uretim)

5- Test Execution (Testin Uygulanmasi)

Her şey hazır olduğunda, planlanan tüm functionality (ler) test edebilir ve yürütebilirsiniz.

Manuel Testerlar testlerini manuel olarak yapar
Automation (Cross-functional) Tester scriptlerini yazarak test ederler

Test Execution işlemi localde ya da Jenkins gibi programlar aracılığı ile yapılabilir.

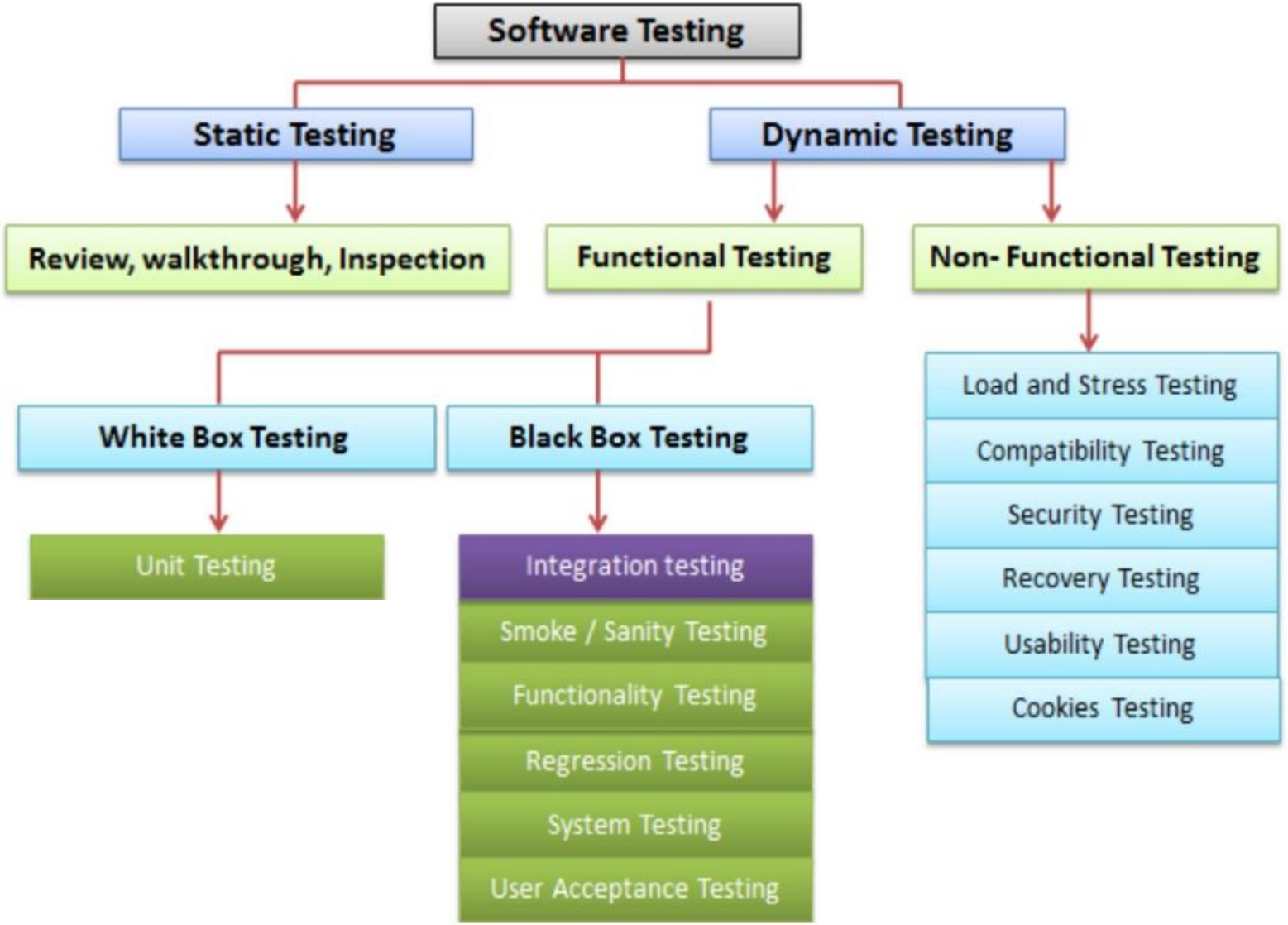
6- Test Cycle Closure (Test Dögüsü Kapanışı)

Definition Of Done (Bitti tanımı) tanımına göre Test Döngüsü sonlandırılır.

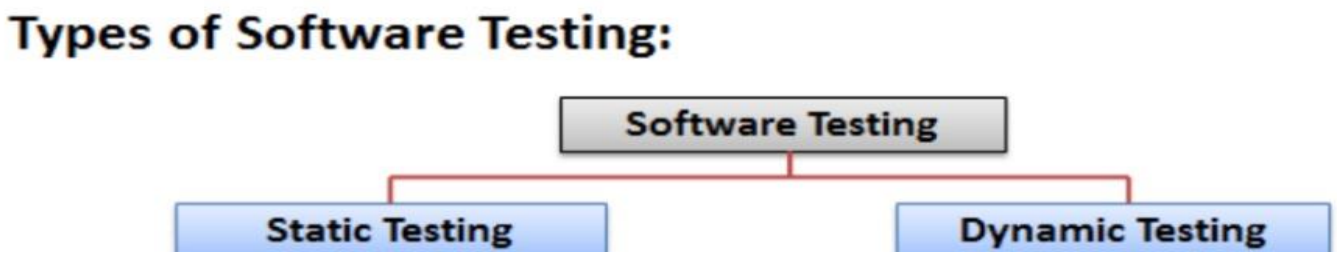
Test planında verilen kriterlere göre karar verilir.

-Koşulan test sayısı 85/100
-Hatasız çalışan test sayısı 95/100

TEST ÇEŞİTLERİ

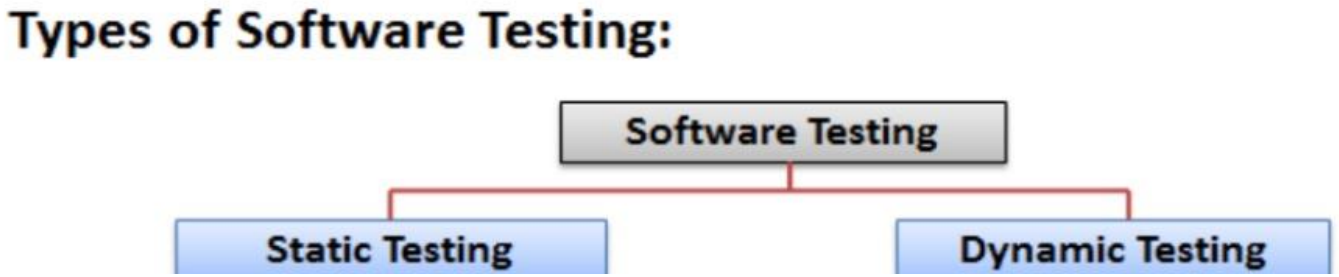


TEST ÇEŞİTLERİ



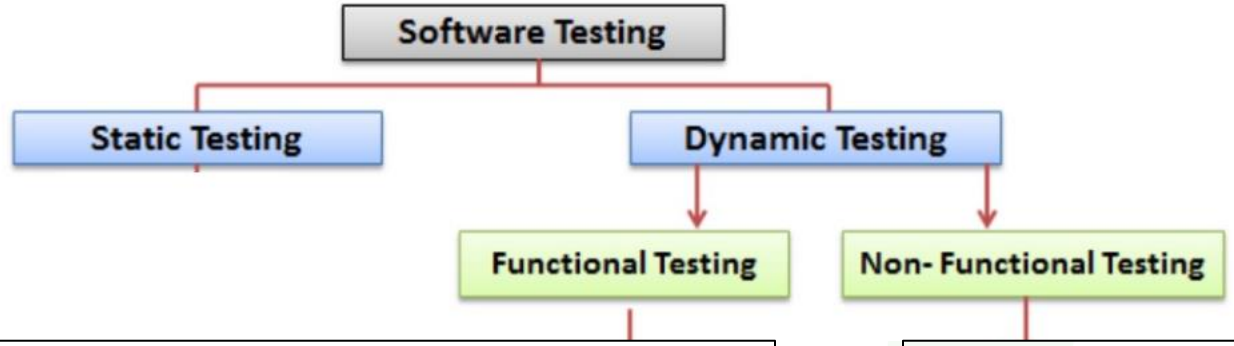
Statik test kod yürütülmeden kodun veya diğer proje dokümanlarının manual olarak gözden geçirilmesidir. **Statik** testler dinamik testlere geçilmeden önce yapılmalıdır. Projenin başlarında gözden geçirme yoluyla tespit edilen hataların çözülmesi ilerleyen aşamalarda bulunmasından daha az maliyetlidir.

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.

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

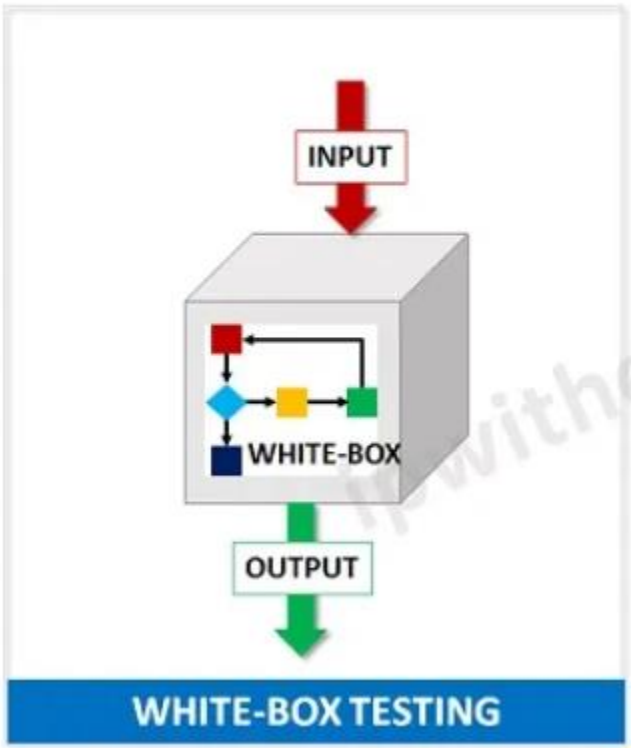
Unit Testing

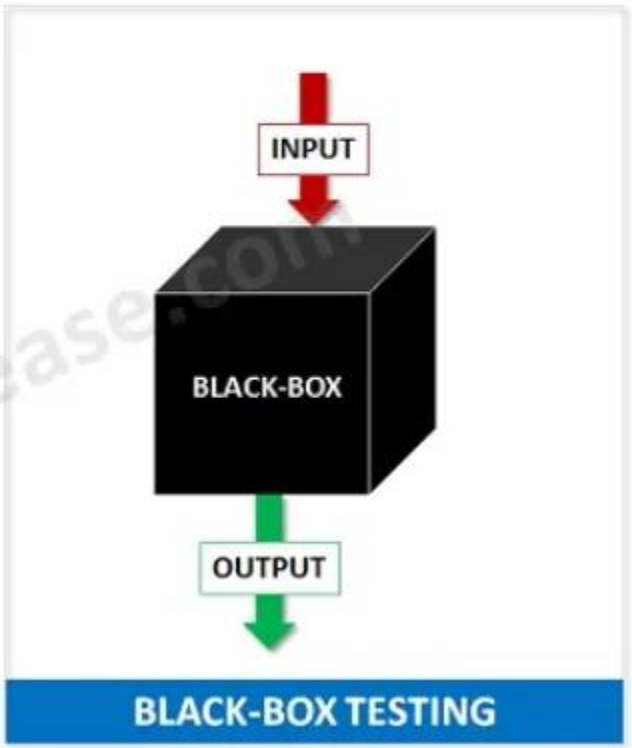
Functional Testing

White Box Testing

Black Box Testing

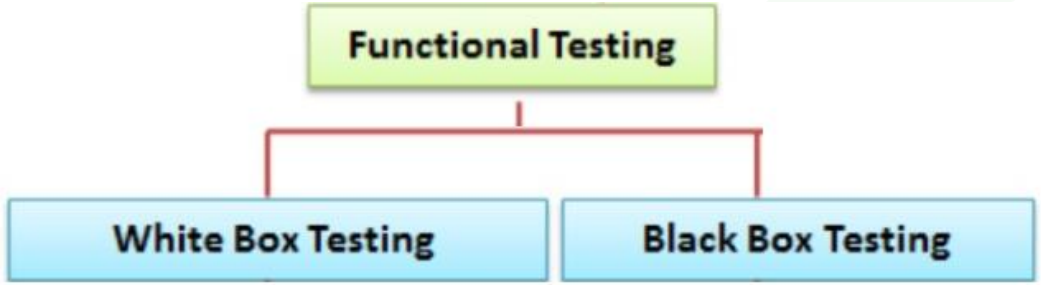
WHIE-BOX TESTING vs BLACK-BOX TESTING





- 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)

UNIT TEST (BİRİM TESTİ)- whitebox

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.

Birim testi sırasında gözden kaçmış

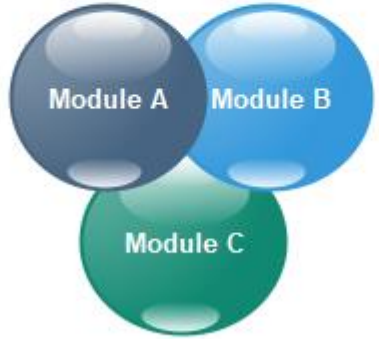
INTEGRATION TEST(Entegrasyon Testi)

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.

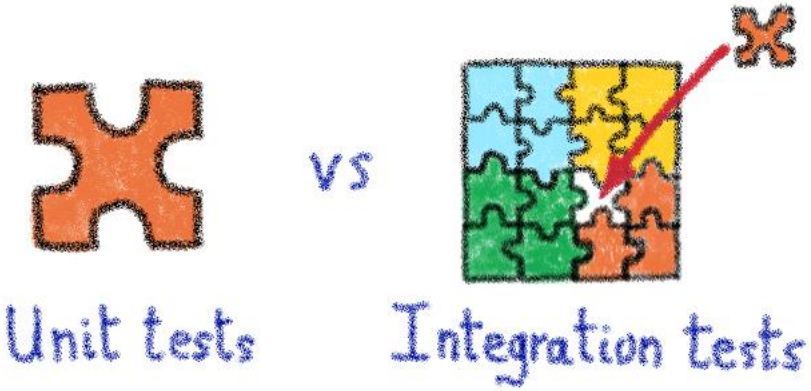
Birim testi sırasında gözden kaçmış olabilecek kusurları, birim testi yapıldıktan sonra istemcilerin gereksinimlerinde meydana gelen değişiklikler gibi faktörlerden kaynaklanabilecek kusurları, harici donanım arayüzlerinden kaynaklanan hataları, yazılımlar arasındaki etkileşimlerden kaynaklanan hataları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

INTEGRATION TEST(Entegrasyon Testi)

Tested in Unit Testing

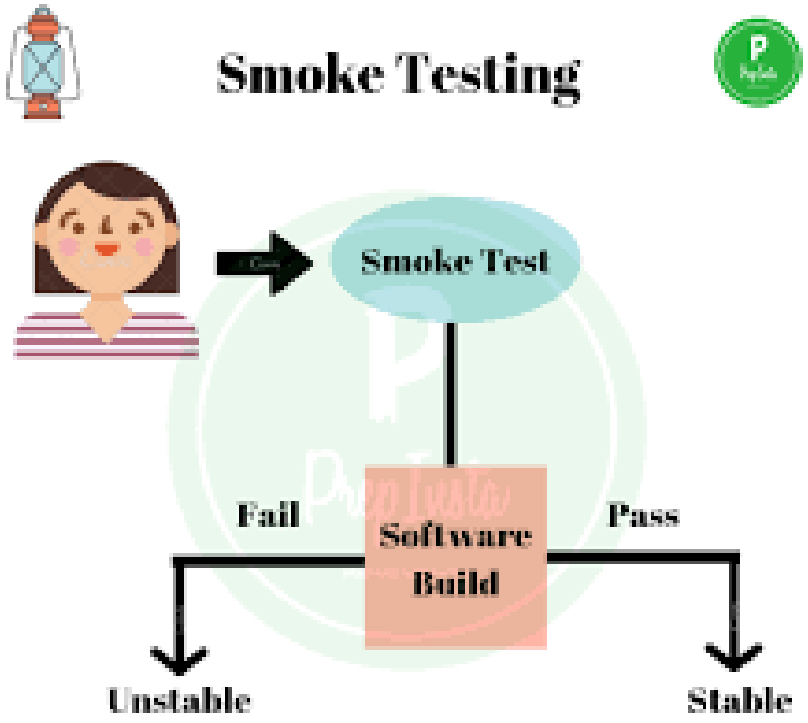


Under Integration Testing



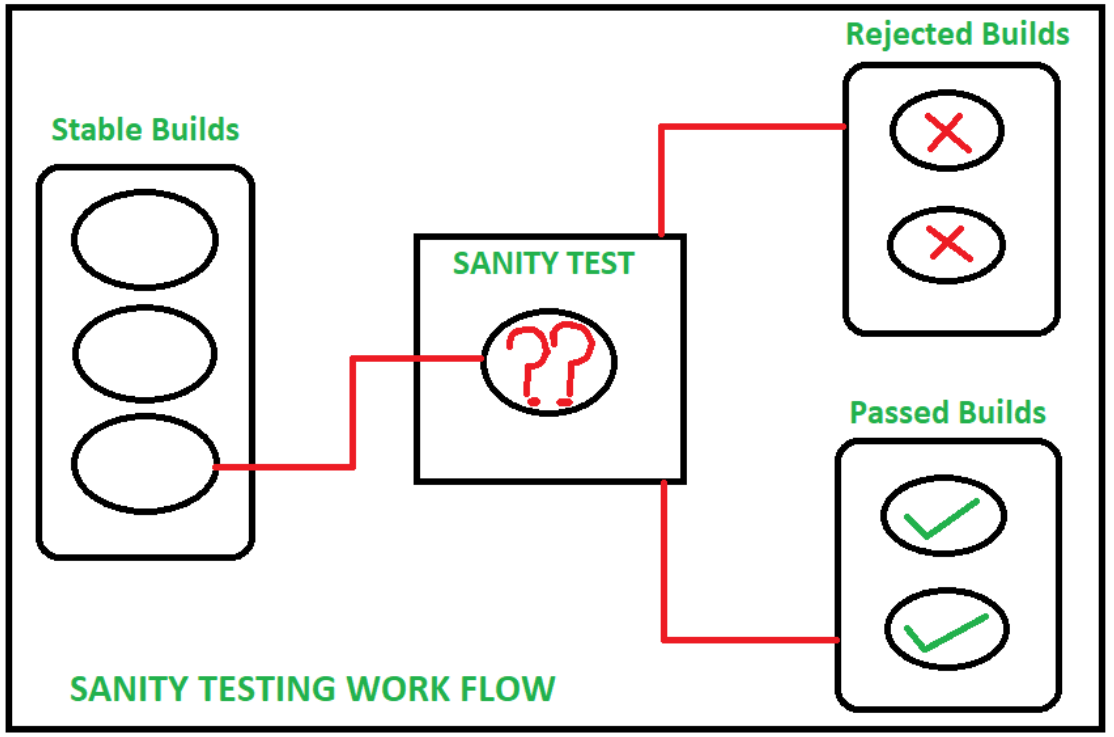
SMOKE TEST(Duman Testi)

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



SANITY TEST(Duman Testi)

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.

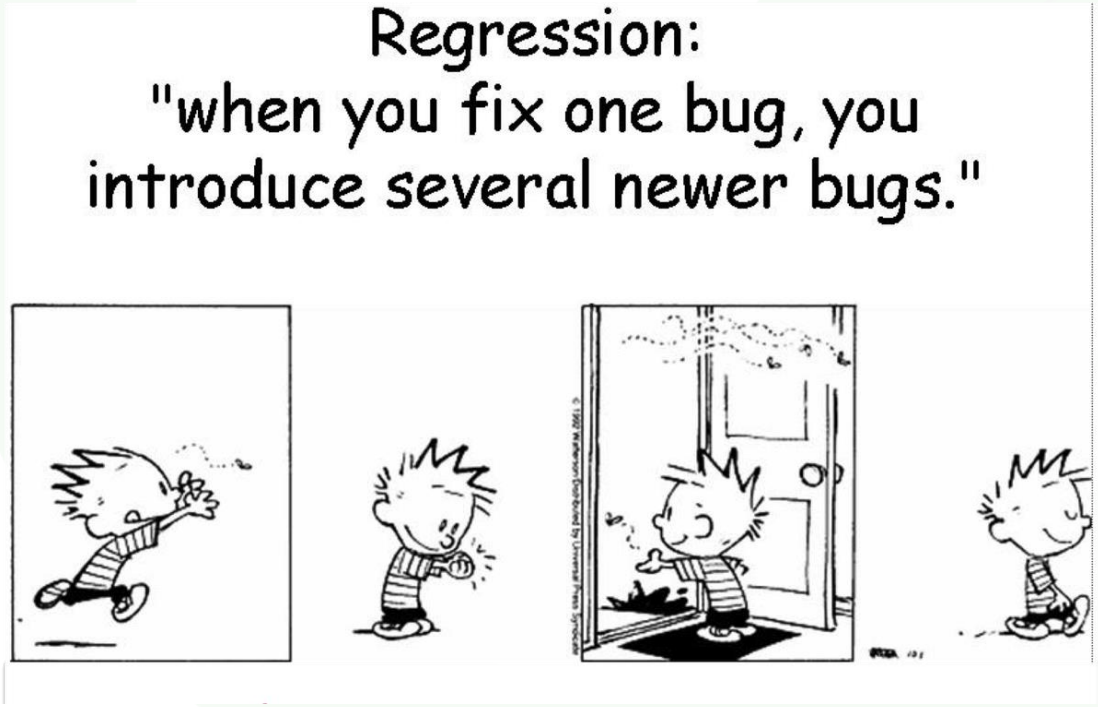


REGRESSION TEST(Regresyon Testi)

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.

2. 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)

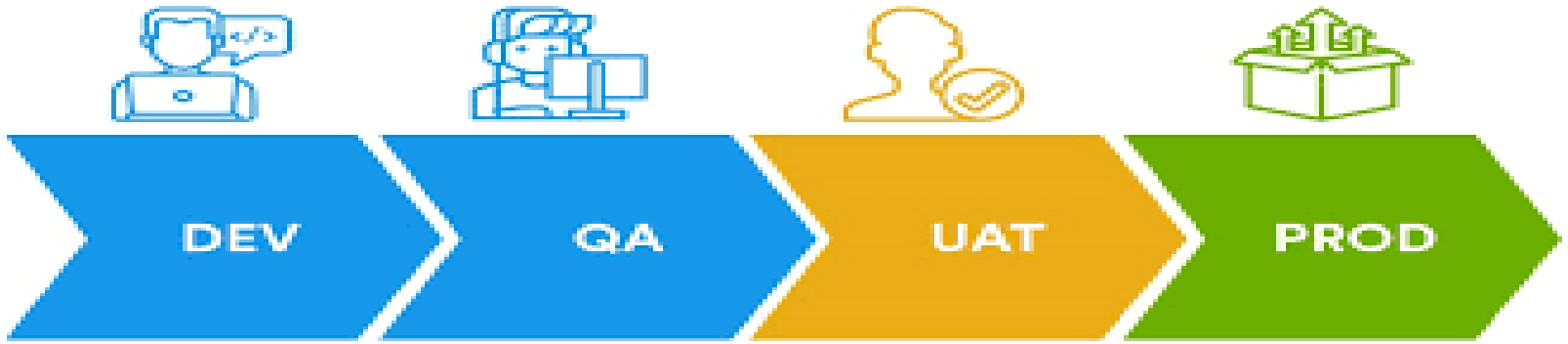
- İki şekilde ifade edilebilir
  - ❑ Farklı fonksiyonların birleştiği uçtan uca senaryolar
  - ❑ Aynı modülün backend ve ui tarafında ele alındığı senaryolar

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 testi olarak ta adlandırılır.



Test Piramidi – Test Hiyerarşisi