



BATCH : BATCH 59

LESSON : ISTQB

DATE : 23.05.2022

SUBJECT : Foundation Level

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



ISTQB HAKKINDA...



ISTQB NEDİR ?

International Software Testing Qualifications Board

Uluslararası Yazılım Testi Yeterlilikler Kurulu



ISTQB, yazılım test uzmanlarının sertifikalandırılması konusunda dünyanın en önde gelen kuruluşudur.

ISTQB terminolojisi, endüstride **yazılım testi alanında** **fiili dil** olarak kabul edilir ve dünya çapındaki profesyonelleri birbirine bağlar.



ISTQB SERTİFiKASI NE İSE YARAR?

- ISTQB sertifikası ile kariyerinizi bir adım ileriye taşıyarak, **test uzmanlığı** alanında kendinizi geliştirebilirsiniz.
- ISTQB ; içlerinde ABD'nin ve AB üyelerinin de bulunduğu 70'in üzerinde ülkede yazılım test ve kalitesi konusunda faaliyet süren **uluslararası bir organizasyondur.**





ISTQB SERTİFiKASININ ONEMİ



ISTQB uluslararası geçerliliği olan test mühendisliği alanındaki en **önemli** sertifikasyon programıdır.

ISTQB' nin çeşitli **sertifikaları** var ancak bütün yollar ilk **sertifika** olan Foundation Level sertifikasından geçiyor.

Yazılım testi sertifikasyonu alanında **uluslararası standarttır**. İster test etme kariyerine yeni başlıyor olun, ister birkaç yıldır bu alanda çalışıyor olun, bir **ISTQB sertifikası kazanmak** önemlidir.



NEDEN ISTQB?

- High quality syllab
- Global recognition
- Common language
- Objectivity
- Encourage adherence to a Code of Ethics
- Public Availability
- Openness
- Independence
- Continuous improvement
- Professional standing





ISTQB SERTİFiKASININ AVANTAJLARI

- Profesyoneller için Faydaları
- İşverenler için Faydaları
- Eğitim Sağlayıcıları için Faydaları





Yazılım Testinde Sertifikalı Test Uzmanı - Temel Seviye



- ✓ Temel seviye ders programı yazılım testine dahil veya yazılım testi ile uğraşmış olan herkese yönelikir.
- ✓ Bunlar arasında, test uzmanları, test analistleri, test mühendisleri, test danışmanları, test yöneticileri, kullanıcı kabul testi uzmanları ve yazılımcılar bulunur.
- ✓ Temel seviye nitelik olarak aynı zamanda, yazılım testinin temel kavramlarını anlamak isteyen ve yazılım geliştirme projelerine dahil olan proje yöneticileri, kalite güvence uzmanları, yazılım geliştirme yöneticileri, iş analistleri, sistem analistleri, kullanılabilirlik uzmanları ve yönetim danışmanları için de uygundur.
- ✓ Bu ders programına ilişkin sınava girerek temel seviye sertifikaya sahip olan kişiler, ileri seviye test uzmanı ders programlarına devam edebilirler.



ISTQB- Temel Seviye-Sınav

- Temel seviye sertifika sınavı, bu ders programını baz alacaktır. Sınav sorularına verilecek yanıtlar, bu ders programının birden fazla kısmını ilgilendirebilir. Ders programının tüm bölümlerinden sınav yapılabilir.
- Sınav çöktan seçmelidir.
- Test uzmanı adayları sınavlara, akredite bir eğitimin bir parçası olarak veya bağımsız bir şekilde (örneğin bir sınav merkezinde veya genel bir sınav şeklinde) girebilirler. Akredite bir eğitimin tamamlanması, sınav için ön şart değildir.



ctfl: certified tester foundation level - sınav 40 soru, sınav süresi 1 saatdir.
sertifikayı almak için %65'in üzerinde başarı gereklidir.) %65 başarı- (26 soru)



Ders Programının Düzeni

Test ile İlgili
Temel Bilgiler

Yazılım Yaşam
Döngüsü
Boyunca Test

Statik
Teknikler

Test Tasarım
Teknikleri

Test Yönetimi

Test Aracı
Çeşitleri





Test ile ilgili Temel Bilgiler



Test ile İlgili Temel Bilgiler

Keywords





Test ile İlgili Temel Bilgiler

Test Neden Gerekli?

Test Nedir?

Testin Yedi İlkesi

Temel Test Süreçleri

Test Psikolojisi



Test Neden Gerekli?



YAZILIM NEDİR ?

Yazılım

Tanımlanmış bir işlevi yerine getiren,

- Girdi ve çıktıları olan,
- Herhangi bir donanım üzerinde çalışan,
- Bilgisayar **programı veya programlarından ve kullanım ve bakım rehberleri gibi belgelerden** oluşan bir ürünüdür.





Yazılım Projelerinde Hedef ?

- Müşteri gereksinimlerini karşılayan,
- Hatalardan arındırılmış,
- Müşterinin piyasadaki rekabet gücünü kuvvetlendirecek,
- Belirlenen bütçe ve zaman dahilinde tamamlanacak bir yazılım geliştirmektir.





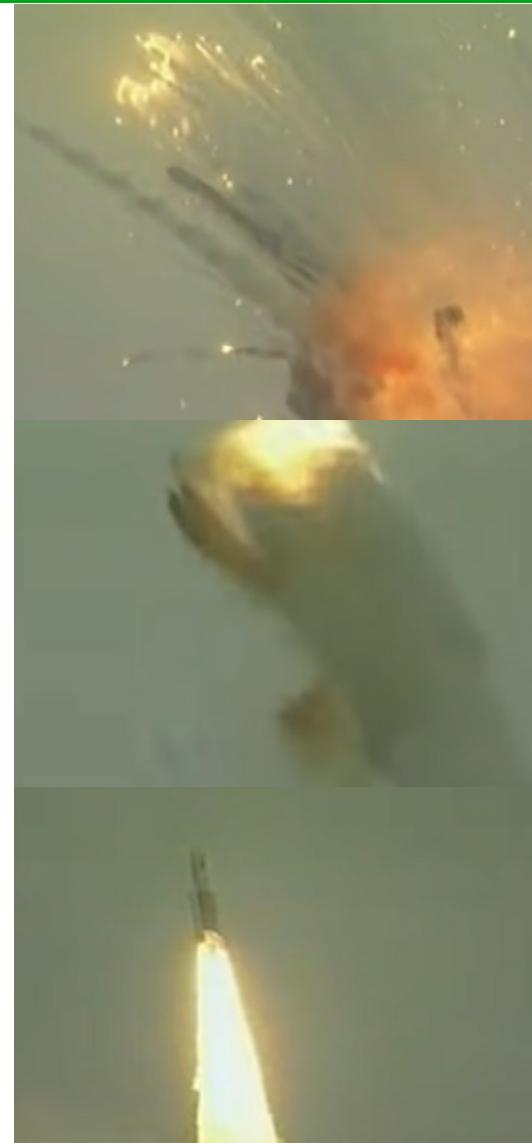
Yazılım Felaketleri

- **Ariane 5 roketinin havaya uçması**

4 Haziran 1996'da Avrupa Hava Ajansının (ESA) ilk roket havalandıktan kısa bir süre sonra düştü. Düşme sebebinin daha sonradan **yazılımdaki bir hatadan kaynaklandığı** anlaşıldı.

Hatanın 64 bit ondalıklı integerin 16 bit işaretli integere dönüştürülürken ondalık kısımda oluşan hatayı kontrol eden exception handling yapısının olmamasından kaynaklandığı anlaşıldı.

Bu kaza, NASA'ya aşağı yukarı 370 milyon dolara mal oldu.





Yazılım Felaketleri

- **Aşırı dozda radyasyon yayan Therac-25**

Therac-25 olarak bahsedilen makine, kanser hastalarının tedavisinde kullanılmak yapılması için tasarlanmıştı. Önceki modelini temel alan Therac-25'te "geliştirilmiş" bir terapi sistemi bulunuyordu ve bu sistem iki farklı türde radyasyon yayabiliyordu. Birincisi, düşük güçte elektron ışını (beta parçacıkları), bir diğeri ise **çok daha güçlü** elektronlar yayan X ışınları.

Therac-25, X ışınlarını elektron tabancası ve hastanın arasında bulunan metal bir plakayla parçacıkları çarpıştırarak tedaviyi uyguluyordu. Bir diğer geliştirme ise Therac-20'de güvenlik önlemi olarak bulunan elektromekanik güvenlik kilitlerinin **yazılımsal güvenlik önlemleriyle değiştirilmesiydi**. Ne yazık ki gelişim olarak görülen bu hata, Therac-20'nin kodlarında bulunan ancak fark edilemeyen hatanın Therac-25'te ortayamasına sebep oldu.





Yazılım Felaketleri

- **Dakikalar içinde kaybedilen 460 milyon dolar**

Knight Capital Group, 1 Ağustos 2012'de yeni bir yazılım güncellemesi yapma kararı aldı. Yaptıkları yazılım güncellemesiyle otomatik olarak stok alım satımı yapmayı planlayan şirket bir anda kendisini hiç ummadığı şekilde **iflasın eşiğinde** buldu.

Saat 09.00 sularında New York Borsası işlemler için açıldı ve Knight Capital'ın ilk perakende yatırımcısı varlıklarını satın almak veya satmak için talimat verdi. Yalnızca 45 dakika sonra Knight Capital'ın sunucuları 4 milyon işlem gerçekleştirdi ve şirkete **460 milyon dolar kaybettirerek** iflasın eşiğine getirdi. NYSE'deki bazı hisseler %300'ün üzerinde arattı. Bunun sebebi, diğer firmaların algoritmaları Knight Capital'ın bu hatasını sömürmeye başladı.





Yazılım Felaketleri

- **Amerikan üssünde patlayan füze**

Yapılan sorgulamaların ve araştırmaların sonucunda patlama sebebinin üstे bulunan anti balistik füze sisteminin bir **yazılım hatası yüzünden ateşlenmemesi** olduğu anlaşıldı. Orta mesafe havadan gelen füzeleri yok etmesi gereken MIM-104 Patriot, 100 saat aşıkın süredir çalışıyordu ve geçen her saatte dahili saat **birkaç milisaniye** ileri gidiyordu.

Bir insan için inanılmaz küçük olan 0,33 saniye, Al Hussein füzesini takip etmek için yapılan bir sistem için **inanılmaz büyük** bir hataydı. MIM-104 Patriot, havada bir cisim olduğunu algılamayı başardı ancak bug yüzünden cismi **takip edemedi** ve bunun bir füze olduğunu anlayamadı. Engellenemeyen füze yüzünden üste bulunan 28 asker hayatını kaybetti.



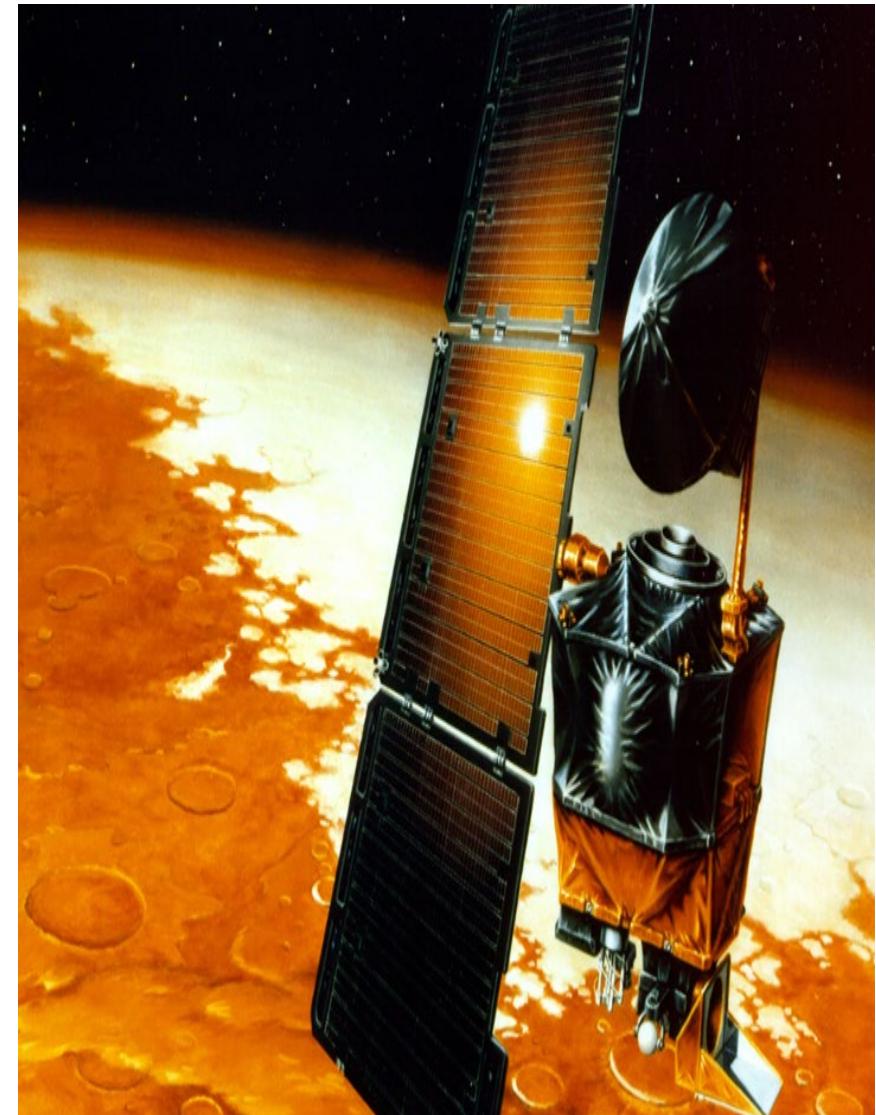


Yazılım Felaketleri

- **Mars Climate Orbiter kazası**

1998 yılında fırlatılan Mars Climate Orbiter uzay aracının amacı, Mars'taki iklimi kontrol etmek ve dünyaya iletmekti ancak ne yazık ki kodlarındaki bir **bug** yüzünden bu görevi hiç tamamlayamadı. Birkaç ay boyunca uzayda yoluna devam eden Mars Climate Orbiter, **yönlendirme hatası** yüzünden tahrip oldu.

Uzay aracını Dünya üzerinde kontrol eden takımlar, **imperial birimi parametrelerini** kullanıyorlardı. Uzay aracının yazılımı ise hesaplamaları **metrik sisteme** göre yapıyordu. Bu yanlış kodlama yüzünden Mars Climate Orbiter olması gerektiği rotadan saptı ve çarpışma yaşadı. Bunun sonucunda ise Mars atmosferinde çok fazla sürtünme yaşayan araç tahrip oldu.





Yazılım Projelerindeki Başarısızlık

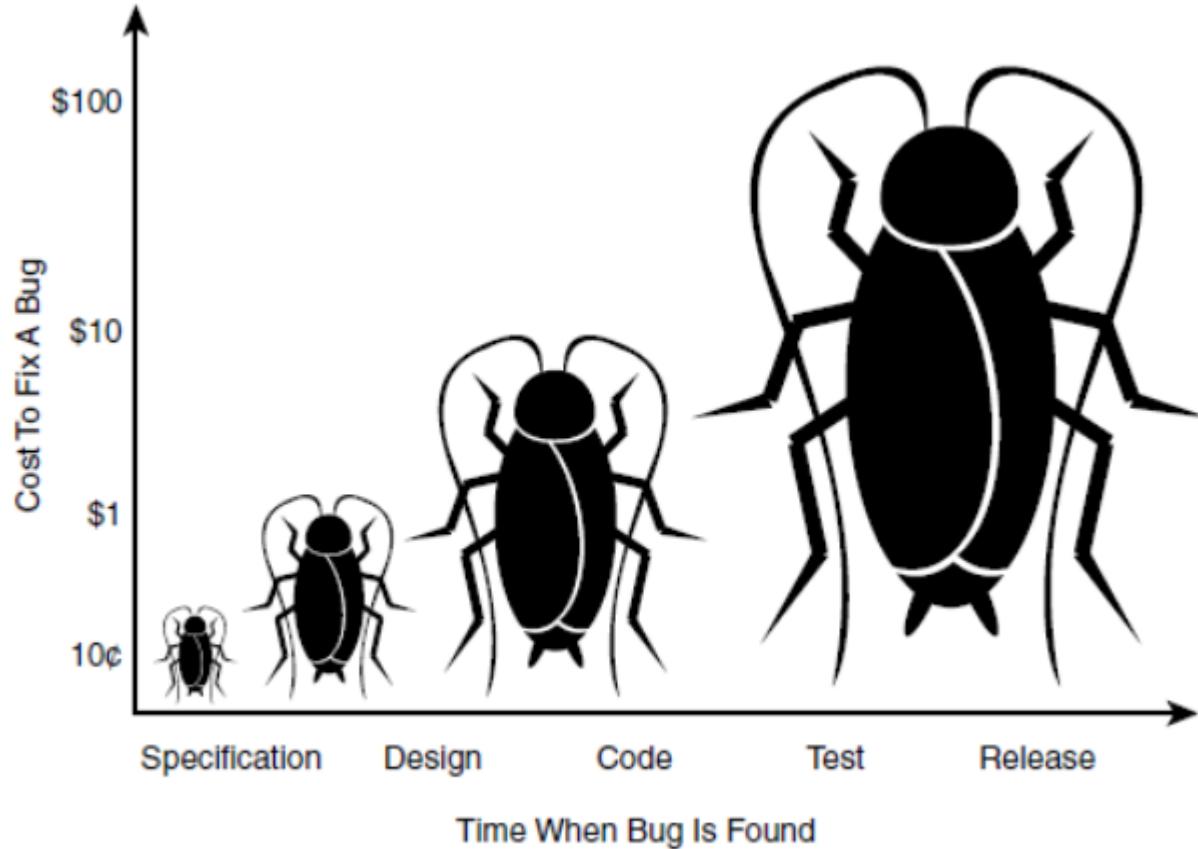
Standish Group' un 2009 yılı için yayımladığı CHAOS raporuna göre bu yıl içinde gerçekleştirilen yazılım projelerinin yaklaşık %68'i başarısızlıkla sonuçlanmıştır (%48'i isterler(gereksinimler) açısından başarısız)

2012 de -> %39 başarılı

Kullanıcı girdisinin olmaması	%12.8
Tamamlanmamış isterler ve Belirtimler	%12.3
Değişen isterler ve Belirtimler	%11.8
Yönetim desteğinin olmaması	%7.5
Teknolojik yetersizlik	%7.0
Kaynak yetersizliği	%6.4
Gerçekçi olmayan bekłentiler	%5.9
Açık olmayan hedefler	%5.3
Gerçekçi olmayan zamanlama tahminleri	%4.3
Yeni Teknoloji	%3.7
Toplam	% 77



Hata Düzeltme Maliyetleri



Yazılım hatasının bulunma zamanının maliyetle ilişkisi.

Hatalar projenin erken safhalarında bulunursa, o hatayı düzeltmek daha kolay ve hatanın düzeltilme maliyeti o kadar az olur

Hatanın geç bulunması...daha fazla belgeye, koda, teste neden olur, bu yüzden maliyet artar



Yazılım Buzdağı





Verification & Validation

YAZILIM

Verification

Doğrulama

Onaylama

Validation

- **Doğrulama:** Yazılımın doğru şekilde üretilmesini sağlamaktır. Geliştirme sürecinde her aşamanın çıktısı, o aşamanın gereklerine göre kontrol edilir. Doğrulama ile «Ürün doğru mu geliştirildi?» sorusuna cevap aranır.
- **Onaylama:** Geliştirilen yazılımın kullanım amacına uygunun gösterilmesidir. Onaylama ile «Doğru ürün mü geliştirildi?» sorusuna cevap aranır.



Verification & Validation

YAZILIM

Verification

Doğrulama

Onaylama

Validation

Gözden geçirme

Test



Test Nedir?



Yazılım Testi Nedir ?

Keywords





Yazılım Testi Nedir ?



Testing, sistemin belirtilen gereksinimleri karşılayıp karşılamadığını kontrol etmenin yanında, sistemin çalışma ortamında kullanıcı (user) ve diğer paydaşların(stakeholder) ihtiyaçlarını karşılayıp karşılamayacağını kontrol etmeyi de içerir.

- Bazı testler, testin isleyişine test edilen bileşen(component) ve testin gerçekleştiği sistemi de dahil eder; bu tür testlere **dinamik test** denir. Bazı testler ise, test edilen bileşen ve sistemi çalıştırılan testlere dahil etmez; bu tür testlere **statik test** denir. Bu nedenle, testing aynı zamanda gereksinimler(requirements), user story ve kaynak kodu(source code) gibi çalışma ürünlerinin(work product) gözden geçirilmesini de içerir.



Yazılım Testi

Statik

Kod çalıştırılmadan

Dinamik

- Kod çalıştırılarak

Sınırlı

- Yeterli sayıda

Seçilen

- Uygun test durumları

Beklenen

- Beklenen ve tanımlı özelliklere uyan

Test; hata bulma amaçlı
planlı bir şekilde gerçekleştirilen eylemler dizisi, bir doğulama methodudur.



Hata - Bug

Yanlış/Hata (Error)

- Bir insan tarafından gerçekleştirilen ve doğru olmayan sonuç üreten bir eylemdir

Kusur/Hata (Fault-Defect-Bug)

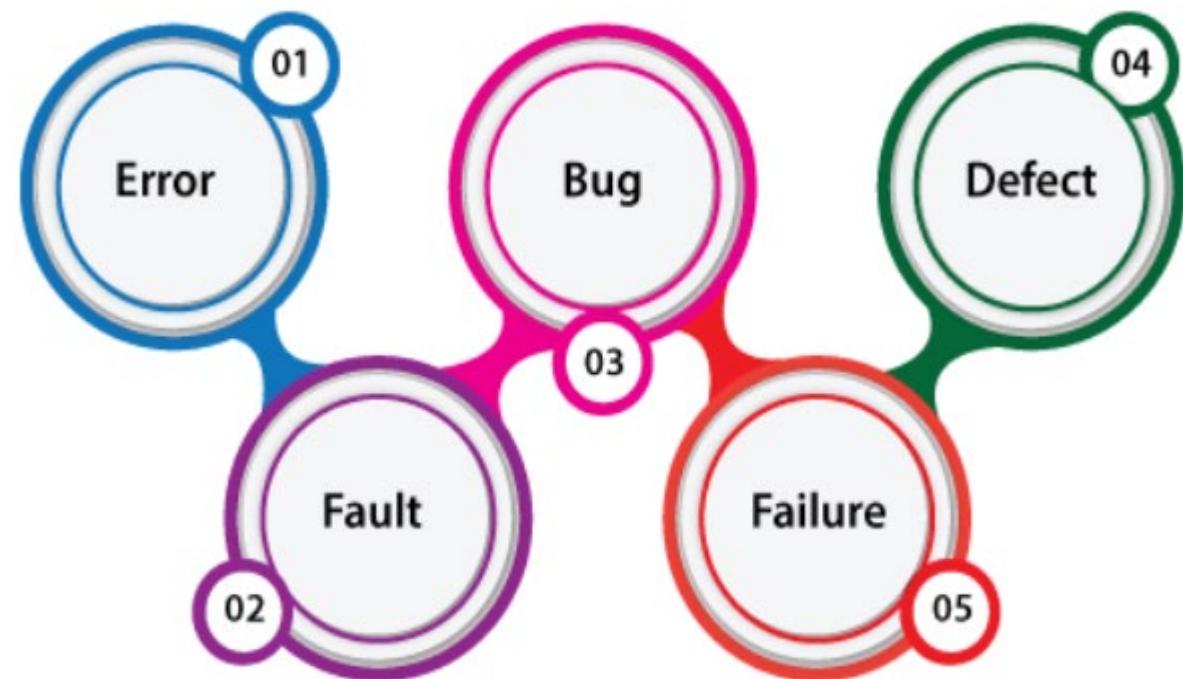
- Yazılımdaki bir yanlışın ortaya çıkmasıdır, ancak eğer çalıştırılırsa; kusur arızaya sebep olur

Arıza (Failure)

- Bileşen component) veya sistemden beklenen teslimat, servis veya sonuçtan sapmasıdır



Hata - Bug



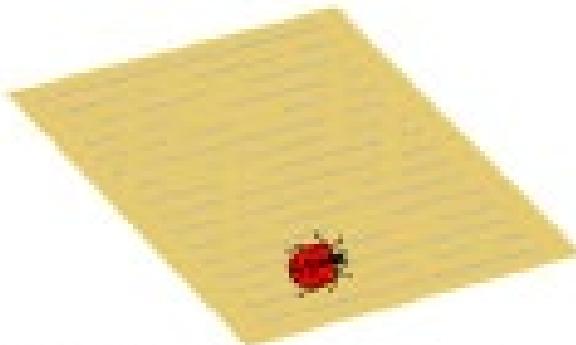


Error-Fault-Failure

A person makes
an error ...



... that creates a
fault in the
software ...



... that can cause
a failure
in operation



Örnek...

tek bir yanlış kod satırı

- Müşteri şikayetleri etkileridir(effect)
- Yanlış faiz ödemeleri başarısızlığıdır(failure)
- Koddaki yanlış hesaplama bir kusurdur(defect)
- Orijinal kusurun temel nedeni ise, product owner' in user story yazarken bir hata yapması- bilgi eksikliği





Örnek...



Yazılımcı yanlış
(error) yapar (yanlış
kodlar)

```
newUsernames = new ArrayList<String>();
newPasswords = new ArrayList<String>();
newUsernames.add("newUser");
newPasswords.add("newPass");
if(newUserName != null &
    newPassword != null) {
    _userName = newUserName;
    _password = newPassword;
    return true;
} else {
    return false;
}
```

... Bu yazılım için bir
kusur (fault) demektir....

**İKİ UÇAĞIN ÇARPIŞMASI
SON ANDA ÖNLENDİ!**



... Bu kusurlar çalışma
sırاسında arızaya
(failure) neden olur.



Hata-Bug

1947 yılında Harvard'da Hesaplama Laboratuar'ında Mark II isimli bilgisayarda iki elektrikli röle arasında gerçek bir böceğin -bir güve olduğu söylenmektedir- hataya yol açmasından sonra bilgisayar dünyasında **ilk “bug”**ın gerçek bir böcek tarafından kaynaklanmasıdan sonra kullanılmaya başlanmıştır.

A cartoon illustration of a small, round, orange alien with a single antenna and two large eyes. It is standing on a grid of binary digits (0s and 1s) arranged in a 10x10 pattern. The alien has its arms raised in a joyful or excited pose.

9/9

0800 Autan started
 1000 stopped - autan ✓ $\left\{ \begin{array}{l} 1.2700 \quad 9.037842.025 \\ 9.037846.745 \end{array} \right.$
 1300 (2nd) MP - nc ~~1.2700~~ 4.615925059(-4)
 023 PRO 2 2.130476405
 Convict 2.130476405
 Relays 6-2 ~ 033 failed special speed test
 in Relay 10.00 test.
 (Relay) changed

1100 Started Casino Tape (Sine check)

1515 Started Multi Adder Test.

1545  Relay #70 Panel F
 (Moth) in relay.

1600 First actual case of bug being found.
 Autan started.

1700 closed down.



Hatalar(Error) Oluşma Nedenleri

- Zaman baskısı
- İnsan hataları
- Deneyimsiz veya yetersiz vasıflı proje çalışanları
- Gereksinimler ve tasarım hakkında yanlış bilgilendirme dahil olmak üzere proje çalışanlar arasındaki iletişimsızlık
- Kodun, tasarımın, mimarinin, çözülmesi gereken temel sorunun ve/veya kullanılan teknolojilerin karmaşıklığı
- Dahili sistem(intra-system) ve harici sistemlerin(inter-system) ara yuzleri arasında yaşanabilecek yanlış anlamalar (özellikle bu tür sistem içi ve sistemler arası etkileşimlerin sayıca fazla olduğu durumlarda)
- Yeni, tam bilinmeyen teknolojiler





Unexpected- False positives- False negatives

- ✓ Beklenmeyen(**unexpected**) test sonuçlarının tümü başarısızlık(failure) değildir.
- ✓ **Yanlış pozitifler**, testlerin uygulanma şeklindeki hatalardan veya test verilerindeki, test ortamındaki veya diğer test yazılımındaki hatalar nedeniyle veya diğer nedenlerden dolayı ortaya çıkabilir.
- ✓ Benzer hataların veya insan hatalarının **yanlış negatiflere** neden olduğu bunun tersi durumlar da ortaya çıkabilir. Yanlış negatifler, bulunması gereken hataları bulamayan testlerin sonuçlarıdır; yanlış pozitifler hata olarak rapor edilmesine rağmen,其实 hata değildir.





Testing Niçin Gereklidir, Neden Yapılır ?

- Yazılımda hata olabilir
- Yazılımın güvenirliliği öğrenilmek istenebilir
- Yüksek arıza maliyeti önlemek istenebilir
- Arızaların telafi edilemeyecek sonuçları olabilir
- Piyasada güvenilir ve başarılı etiketi tescil edilmek istenebilir
- Olumsuz hukuki sonuçları önlemek için





Testing Niçin Gereklidir, Neden Yapılır ?

Neler test edilir?

- ✓ Yazılımdan istenilenler yerinde ve yapılmış mı? **validation**
- ✓ Yazılım istenilen işlevleri yerine getiriyor mu? **verification**
- ✓ Yazılım işlevleri yaparken hata veriyor mu? **reliability**
- ✓ Yazılım istenilen hızda yapıyor mu? **performance**
- ✓ Yazılım istenilen kadar işlev yapabiliyor mu? **load**
- ✓ Yazılım istenilen işlevleri en çok ne kadar yapıyor? **stress**
- ✓ Yazılım istenilen kolay yapıyor mu? **usability**
- ✓ Yazılım istenilen işlevleri güvenli yapıyor mu? **security**
- ✓ Yazılım işlevleri her zaman yapabiliyor mu? **compatibility**



Testing Neden Yapılmaz ?

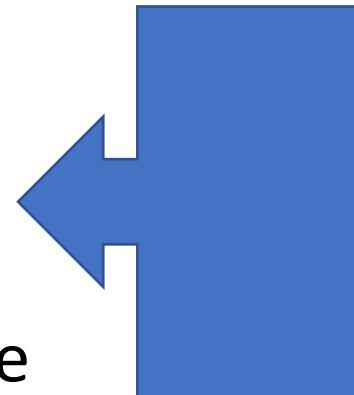
- Yazılımı hatasız olduğunu göstermek
- Proje planında olduğu için
- Proje takviminde boşluk olduğu için
- İmaj için





Ne Kadar Test ???

- Sürekli, her zaman
- Planlanan süre bittiği zaman
- Kullanıcı yeterli dediğinde
- Yazılımın doğru çalıştığı ispat edildiğinde



- Yazılımın doğru çalıştığından emin olunduğunda
- Yazılımın risk değerlendirmesine bağlı olarak

şartlara göre
doğru kabul edilir

Kesin doğru....



Testing' in Temel Amaçları

- Gereksinimler(isterler), user story, tasarım ve kodlar gibi çalışma ürünlerini(work product) değerlendirerek kusurları önlemek
- Belirtilen tüm gereksinimlerin karşılanıp karşılanmadığını doğrulamak
- Testin tamamlanıp tamamlanmadığını kontrol etmek ve kullanıcılar ve stakeholder'ların beklediği gibi çalışıp çalışmadığını doğrulamak
- Testimizin kalite düzeyine güven oluşturmak





Testing' in Temel Amaçları

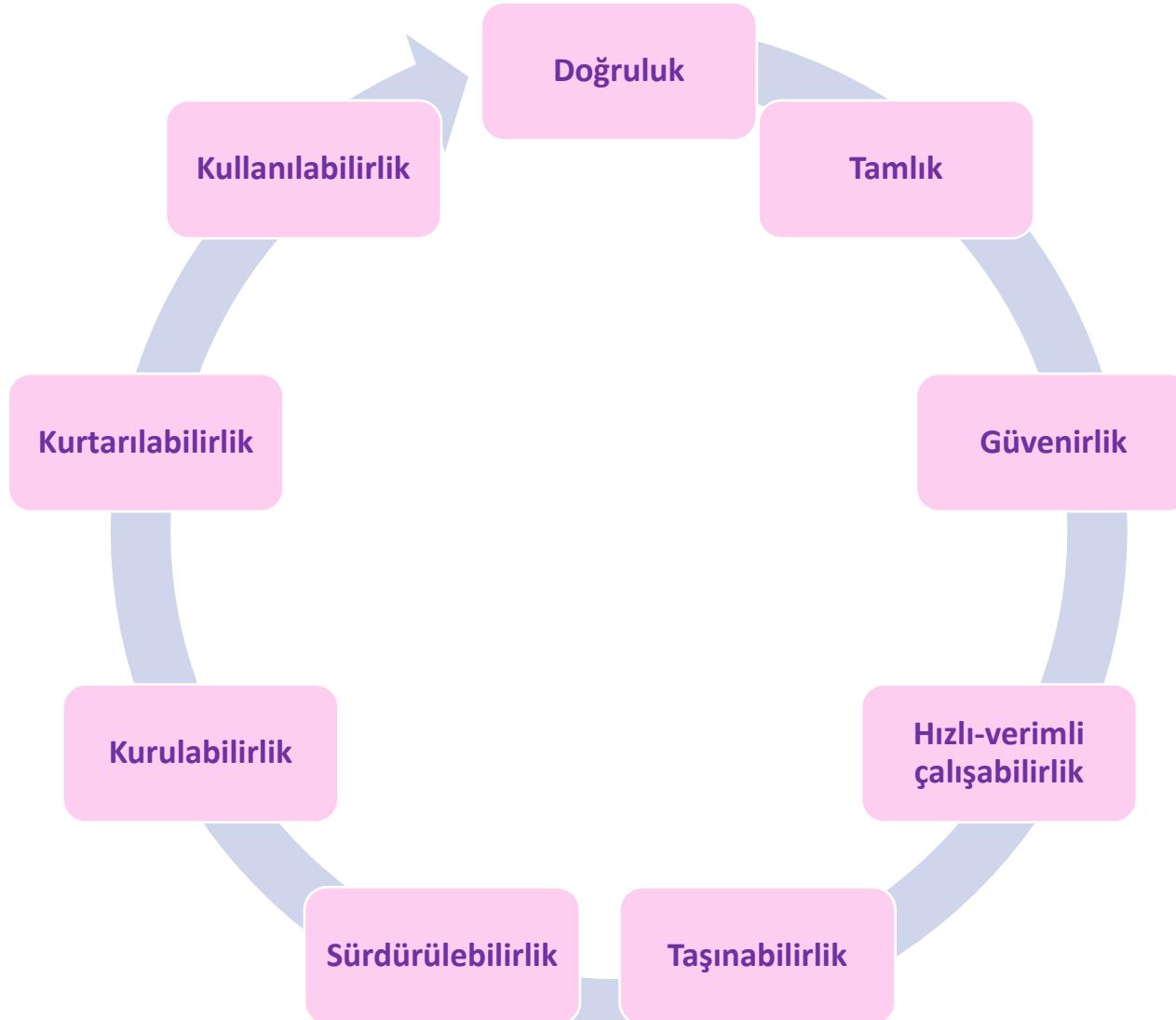
- Kusurları(defects) ve arızaları(failures) bulmak, böylece yazılımın “yetersiz yazılım kalitesine (inadequate software quality)” düşme riskini azaltmak
- Paydaşlara(stakeholder), bilinçli kararlar vermelerini sağlamak için yeterli bilgi vermek, özellikle de testimizin kalite düzeyi ile ilgili bilgiler sağlamak
- Sözleşmeye dayalı, yasal veya düzenleyici gereksinimlere veya standartlara uymak ve/veya testimizin bu tür gereksinimler veya standartlara uygunluğunu doğrulamak





Testing' in Temel Amaçları

Yazılım Testi
Neyi
Amaçlar ???





BATCH : **BATCH 59**
LESSON : **ISTQB-02**
DATE : **30.05.2022**
SUBJECT : **ISTQB**

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



Testin Yedi İlkesi



Testing' in Yedi Prensibi

1. Test, kusurların varlığını gösterir, yokluğunu değil

- ✓ Testing, kusurların mevcut olduğunu **gösterebilir**, ancak **kusur olmadığını kanıtlayamaz**.
- ✓ Test etme, yazılımda kalan keşfedilmemiş hataların olasılığını azaltır, ancak hiçbir hata kalmasa bile testing, yazılımın tamamen doğru olduğunun bir **kanıtı** olamaz.





Testing' in Yedi Prensibi

2. Yazılımı bütünüyle test etmek imkansızdır

- ✓ Bir yazılımın tamamını **test etmek** (tüm girdi ve önkoşul kombinasyonları dahil) küçük projeler dışında **mümkün değildir**.
- ✓ Bir yazılımın tamamını test etmeye çalışmak yerine, risk analizi, test teknikleri ve öncelikler kullanılarak test faaliyetlerini gerekli yerlere yoğunlaştırmak gereklidir.





Testing' in Yedi Prensibi

3. Testlerin erken başlaması zaman ve para kazandırır

- ✓ Hataları erken bulmak için, yazılım geliştirme yaşam döngüsünde statik ve dinamik test faaliyetleri mümkün olduğunca erken başlatılmalıdır.
- ✓ Testing' e erken başlamaya bazen **sola kaydırma(shift left)** denir.
- ✓ Yazılım geliştirme yaşam döngüsünün başlarında test yapmak, geç fark edildiği için maliyeti artacak değişiklikleri azaltmaya veya ortadan kaldırmaya yardımcı olur.





Testing' in Yedi Prensibi

4. Kusurlar belirli alanlarda yoğunlaşır (cluster together/hata kümeleşmesi)

- ✓ Bir uygulama içerisindeki az sayıda modül release(sürüm öncesi testler sırasında keşfedilen **kusurların çoğunu içerir** veya operasyonel arızaların çoğundan sorumludur.
- !!! (Pareto Yasası)**
- ✓ Öngörülen hata kümeleri ve test veya operasyonda gerçek gözlemlenen kusur kümeleri, test çabasına odaklanmak için kullanılan bir risk analizine önemli bir girdidir (ilke 2'de belirtildiği gibi).





Testing' in Yedi Prensibi

5. Pestisit paradoksuna dikkat edin

(Türkçede antibiyotik direnci tabiri uygun olabilir)

- ✓ Aynı testler defalarca tekrarlanırsa, bu testler artık yeni bir kusur bulamaz. Yeni kusurları tespit etmek için mevcut testlerin, test verilerinin değiştirilmesi veya yeni testlerin yazılması gerekebilir.(Tıpkı pestisitlerin bir süre sonra böcekleri öldürmede artık etkili olmadığı gibi, testler de artık kusurları bulmada etkili olmazlar.)
- ✓ Standart tekrarlanan(automated) regresyon testi gibi bazı durumlarda, pestisit paradoksuna dikkat edilmesi, nispeten daha düşük sayıda regresyon kusuru oluşması gibi faydalı bir sonuca sahiptir.





Testing' in Yedi Prensibi

6. Testing, koşullara(context) bağımlıdır

(Test yaklaşımı ve aktiviteleri yazılım projesinin koşullarına göre değişiklik gösterir)

- ✓ Test aktiviteleri, yazılımın özelliklerine, bağlamına ve içeriğine göre farklı biçimlerde ele alınmalıdır.

Örnek olarak, bir e-ticaret yazılımı ile nükleer santral için yazılmış güvenlik tehlikesi taşıyan bir uygulama farklı şekillerde, farklı test teknikleri ve metodolojileri kullanılarak test edilmelidir.





Testing' in Yedi Prensibi

7. Hataların olmaması bir yanlışıdır (hata yokluğu yanlışsı)

- ✓ Bazı kuruluşlar, tester'ların tüm olası testleri çalıştırmasını ve olası tüm kusurları bulmasını bekler, ancak sırasıyla 2 ve 1 numaralı ilkeler bize bunun imkansız olduğunu söyler.
- ✓ Ayrıca, sadece çok sayıda kusuru bulup düzeltmenin bir sistemin başarısını garantiyeceğini beklemek bir yanlışıdır (yanlış bir inançtır).

Örneğin, belirtilen tüm gereksinimlerin kapsamlı bir şekilde test edilmiş ve bulunan tüm kusurlar düzeltilmiş bir yazılım, hatasız olmasına karşın kullanımı zor, kullanıcıların ihtiyaç ve bekleyenlerini karşılamayan veya diğer rakip sistemlere kıyasla daha düşük seviyede olan bir sistem üretebilir.





Temel Test Süreçleri



Yazılım Test Süreci Nedir?

Keywords

- Conformance Testing
- Exit Criteria
- Incident
- Regression Testing
- Test Condition
- Test Date
- Test Execution
- Test Log
- Test Summary Report
- Testware
- Test Plan



Yazılım Test Süreci Nedir?

- **Yazılım testleri ne zaman başlar ?**

Yazılım geliştirildikten sonra başlar..



Test eylemi, yazılım geliştirme yaşam döngüsü içerisinde, yazılım geliştirme basamaklarında, en erken safhada başlar..





Yazılım Test Süreci

Planlama ve Kontrol

Analiz ve Tasarım

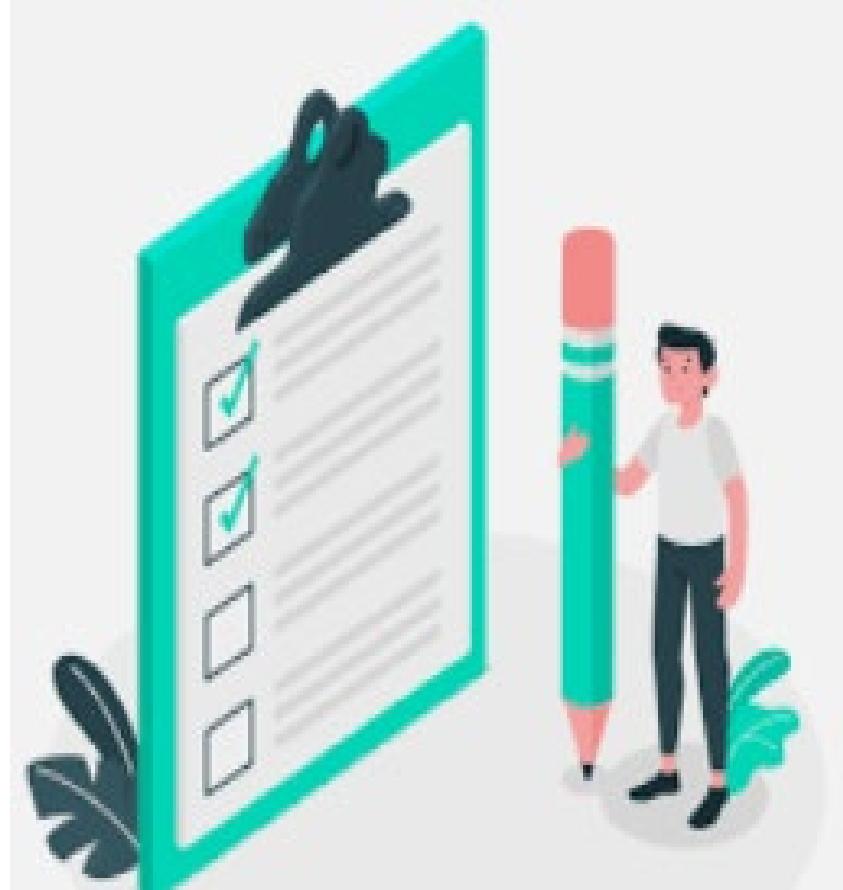
Oluşturma ve Yürütme

Değerlendirme

Testi Sonlandırma



Planlama ve Kontrol



- Tüm planlar bu aşamada yapılır
- **Master test planı yazılır**
- Test Stratejisi belirlenir
- Test yaklaşımına karar verilir
- **Test başlama (entry criteria) ve Test sonlandırma (exit criteria) kriterleri belirlenir**

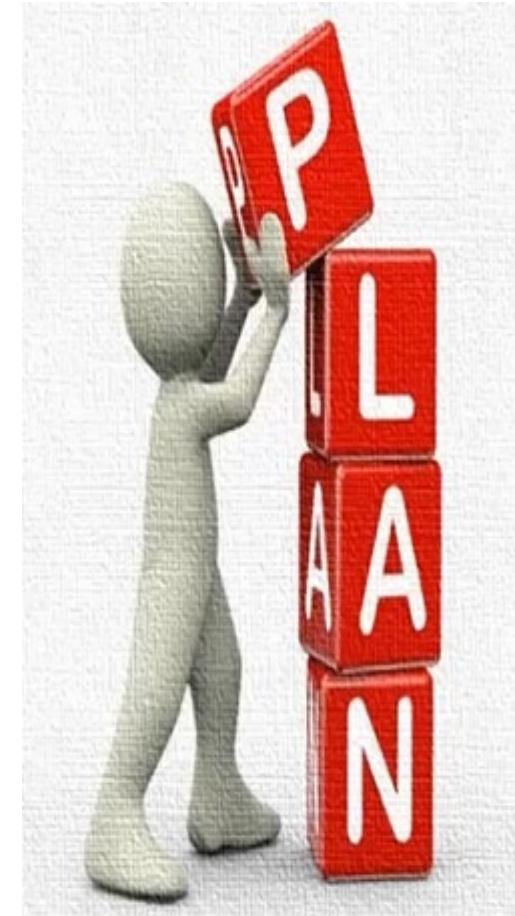


Planlama ve Kontrol

- **MTP - Master Test Plan**

Master Test Planı içerisinde test yönetimine ait tüm bilgileri içeren dökümandır. Bu döküman planlama aşamasında hazırlanır ve tamamlandıktan sonra tüm ilgili paydaşların (yazılımcı, analist, testçi, proje yöneticisi, ...) her zaman ulaşabileceği şekilde yayınlanır.

MTP daha sonraki aşamalarda ikilik çıkması durumunda başvurulacak kaynak niteliğindedir bu yüzden özenle hazırlanmalı ilgili kişilerin onayları alınmalıdır.





Planlama ve Kontrol

Test- Approach(Yaklaşımı)- Test Stratejisi

- **Test Stratejisi**

Test yaklaşımının ana hatlarını çizen ve projeye özgü hazırlanan kurallar bütünüdür.

Organizasyondaki herkesin bilgilenmesi ve bu doğrultuda hareket etmesini sağlar.

Test caselerin tasarılanması, test süreçlerin belirlenmesi, teste başlama ve testi sonlandırma kriterlerin belirlenmesinde ana rol oynar.

- **Test Yaklaşımı**

Projeyi oluşturulan bütün bileşenleri içeresine katarak daha başarılı test sonuçları yakalamak için kullanılan test tiplerinin gruplandırıldığı test anlayışıdır.

Seçilen test yaklaşımı **bağlama(context) bağlıdır ve riskler, güvenlik, mevcut kaynaklar ve beceriler, teknoloji, sistemin doğası, test hedefleri ve düzenlemeler** gibi faktörleri dikkate alabilir.

- **Değerlendirilmesi gereken bileşenler**

Test stratejisini ve yaklaşımını belirlemek için bu kriterler göz önüne alınmalıdır





Planlama ve Kontrol

• Entry Criteria- Teste Başlama Kriteri

Test başlama kriterleri, herhangi bir yazılım çıktılarının testine başlamadan önce sahip olması gereklili en temel özelliklerin listesidir. Bu liste SDLC göz önüne alındığında çeşitli kriterlerden söz edilebilir. Örneğin, geliştirme tamamlandı ve yazılım üzerinde fonksiyonel testlere başlanacak, yazılımın hazır olup olmadığını kontrol etmek gerekiyor.

Bu durumda uzman ya da moderator tarafından şu kontroller yapılabilir:

- 30 dk boyunca kontrol edildi ve bir sayfada 3 büyük defect' ten başka bir şey bulunamıyor ise,
- 5 değişik sayfada 10





Planlama ve Kontrol

- **Exit Criteria- Testi Sonlandırma Kriterleri**

Test için sonsuz zaman ve kaynak olmadığından bir süre sonra test sonlandırmak gereklidir.

Genelde risk analizi yapılarak ne kadar test yapılacağına karar verilir. Test sonlandırmak için belirlenmiş kriterlere **exit criteria** denir.





Planlama ve Kontrol



- Göz önünde tutulması gereken etkenler
- **Yazılım geliştirme stratejisi**
- Müşteri gereksinimleri
- Test Amacı
- Proje alanı ve konusu





Planlama ve Kontrol

Örnek bir değerlendirme yapılır ise;

Bir e-ticaret firmasını örnek olarak aldığımızda 2 Alanda inceleme yapılabilir:

Proje konusu ve Geliştirme Ortamı

✓ **Analitik Yaklaşım:** İnternet üzerinden satış yaptığı için bazı riskler vardır.

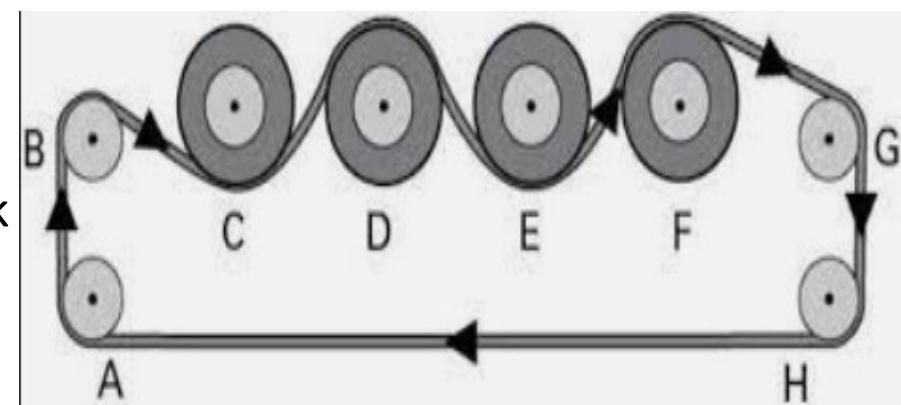
Örneğin; güvenlik problemleri, firma imajını zedeyecek görsel ve yazım hataları, ... olmasından dolayı test caselerin risklerinin sınıflandırılmasında kullanılabilir.

✓ **Dinamik Yaklaşım:** Gereksinimlerin olmadığı, planlamanın tam olarak yapılmadığı hızlı bir yazılım geliştirme ortamı olmasından ötürü de dinamik test yaklaşımı kullanılabilir.

Analitik ve Dinamik test stratejilerinin bileşiminden oluşan;

Risk Temelli - Dinamik test yaklaşımı belirlenebilir

- Hata verme olasılığı ve daha yüksek alanlara daha fazla test yapmak
- Keşif tabanlı testler yaparak hata tahminlerinde bulunarak, ilgili alanlara yoğunlaşmak





Planlama ve Kontrol

Yazılımın doğru çalıştığınından emin olduğunuzda

Yazılım gereksinimleri göz önüne alındığında çok fazla olmayabilir veya uzun bir zaman aralığında testlerinizi başarılı bir şekilde tamamlamış ve daha fazla hata bulamıyor olabilirsiniz bu durumda yazılımın doğru çalıştığı konusunda kendinizi emin hissediyorsanız testleri durdurarak **yayına almak (go live)** gerekir. Fakat emin olma durumu kişiden kişiye göre ve deneyimlere göre değişiklik gösterdiği için bu madde her zaman geçerli değildir.

GOLIVE ((•))



Planlama ve Kontrol



Yazılımın risk değerlendirmesine bağlı olarak

Test için zaman her zaman kısıtlıdır. Her şeyi planlandığı gibi test etmek genelde mümkün olmaz bu durumda riskleri göz önüne alarak:

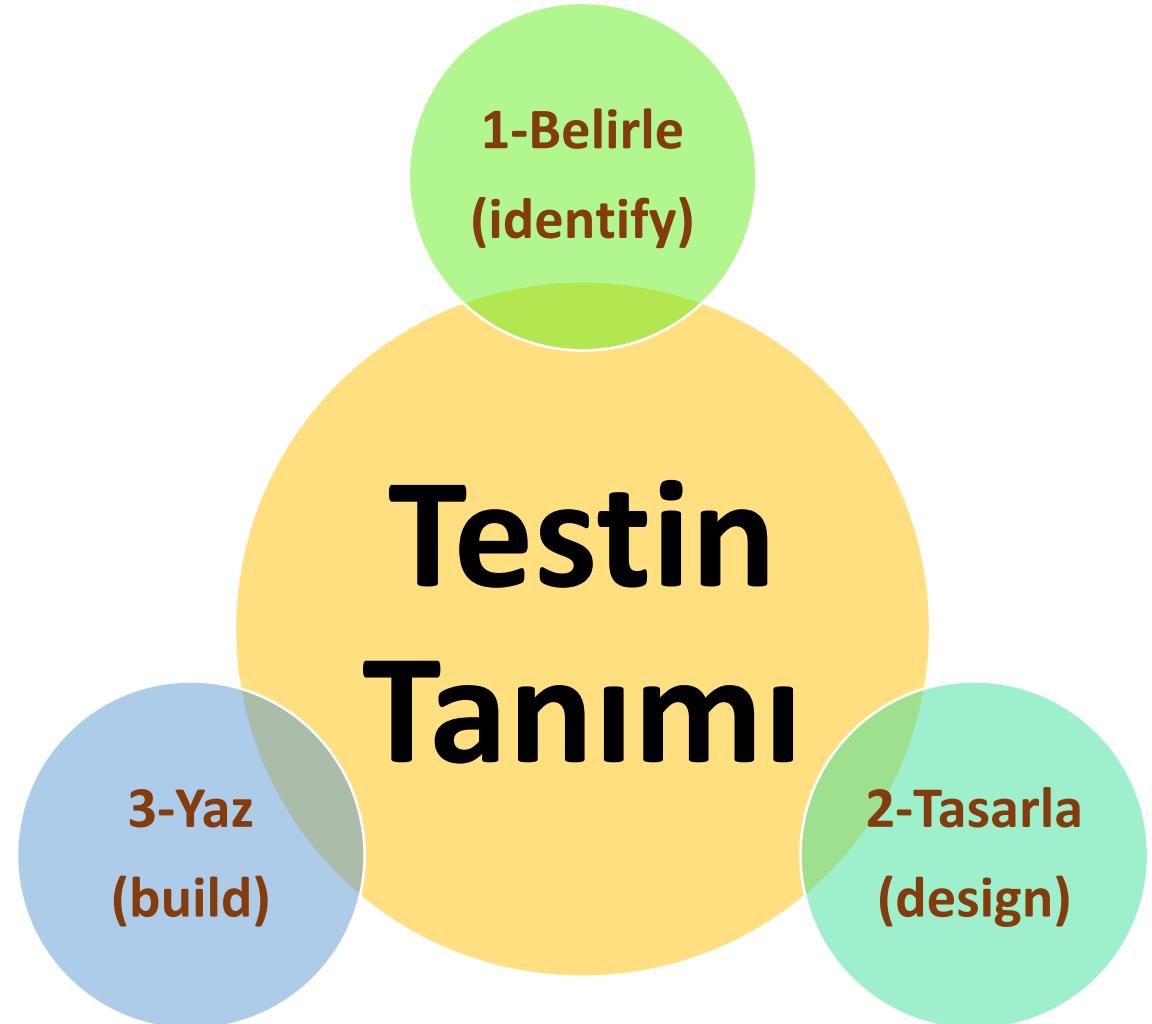
- Nelerin en çok,
- Nelerin ilk,
- Hangi bölümün kısmı veya tamamını,
- Test edilecek bölümlerin ne kadarlık zaman içinde,
- Nelerin test edilmeyeceğini

Belirleyerek test zamanını en etkin nasıl kullanılması gerekiğinin analizi yapıldığında ve bu zamanın sonlanması ile test bitirilebilir.



Analiz ve Tasarım

- Testlerin altyapısı belirlenmelidir
- Testlerin tasarlanmasıdır
- Test ortamının tasarlanmasıdır
- Test edilmesi planlanan sistemlerin ihtiyaçlar kapsamında test edilebilirliğinin belirlenmesidir





Analiz ve Tasarım

identify

Neyin test edileceğini belirlenmeli **ve test edilebilecek (testability)** konular bir liste haline getirilerek önem sırasına göre sıralanmalıdır. Testin başarılı sonuçlanması için gerekli yeter koşullar belirlenmeli ve oluşabilecek en kötü durumlar ve bu durumda yazılım kararlılık tablosu çıkartılmalı. Yazılımın yapısı hakkında bilgiler edinilir.

design

Üst düzey test caseler çıkartılır. Belirlenen konuların nasıl test edileceği belirlenmeli ve test caselerin yazımı için gerekli çalışmalar yapılmalıdır. Testte kullanılacak veriler belirlenmeli, veriler (input) sonucunda yazılımın verdiği cevap (output) belirlenmelidir.

build

Test caseleri yazılmalıdır, Scriptler yazılarak test araçları üzerinden testlerde yapılabilir.

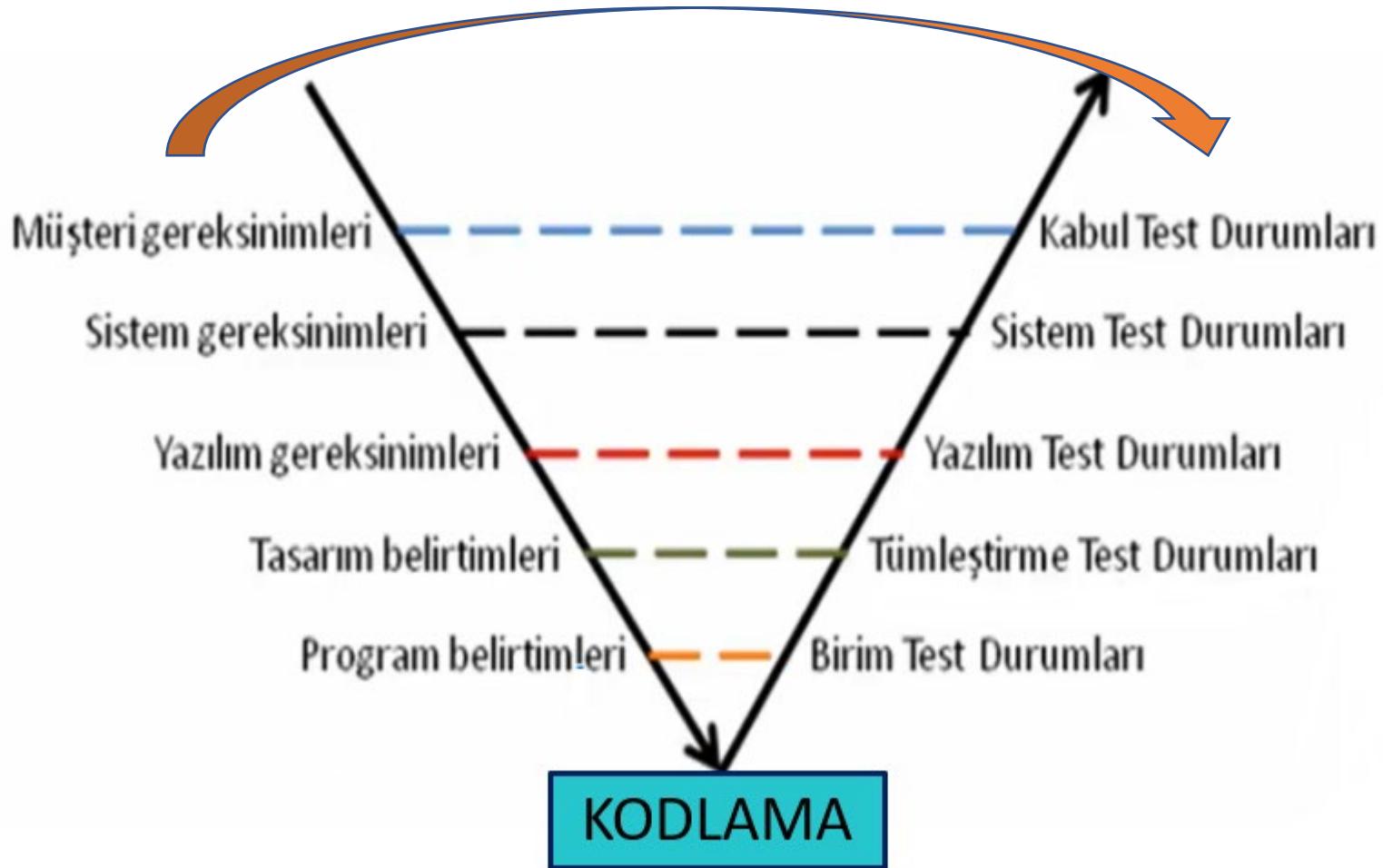


Test Ortamının Hazırlanması





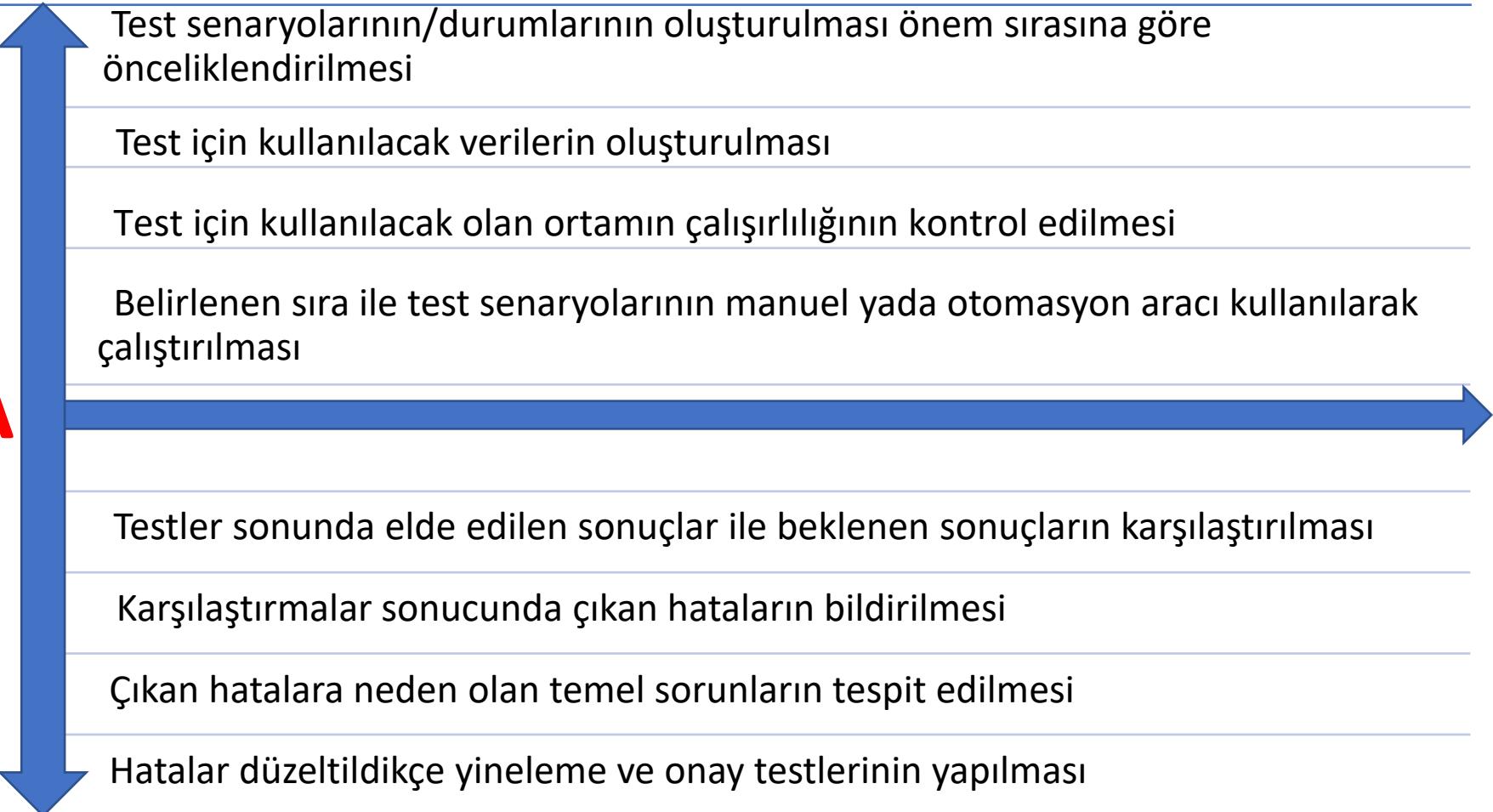
Test Durumlarının Yazılması





Oluşturma ve Yürütme

UYGULAMA





Değerlendirme

DEĞERLENDİRME

- ↑ Test loglarının ve raporlarının incelenmesi
Raporların ve logların planlamada belirlenen sonlandırma kriterlerinin yakalanıp yakalanmadığının kontrol edilmesi
Kontroller sonucunda "Daha fazla yapılmalı mı, yoksa test sonlandırılmalı mı" kararının verilmesi
↓ Test projesi özetinin, üst yöneticiye raporlanması



Test Sonlandırma

**TEST
SONLANDIRMA**



Planlanan çıktıların üretildiğinin, açılan önemli hataların düzeltildiğinin ve test projesinin istenilen şekilde dökümante edildiğinin kontrol edilmesi

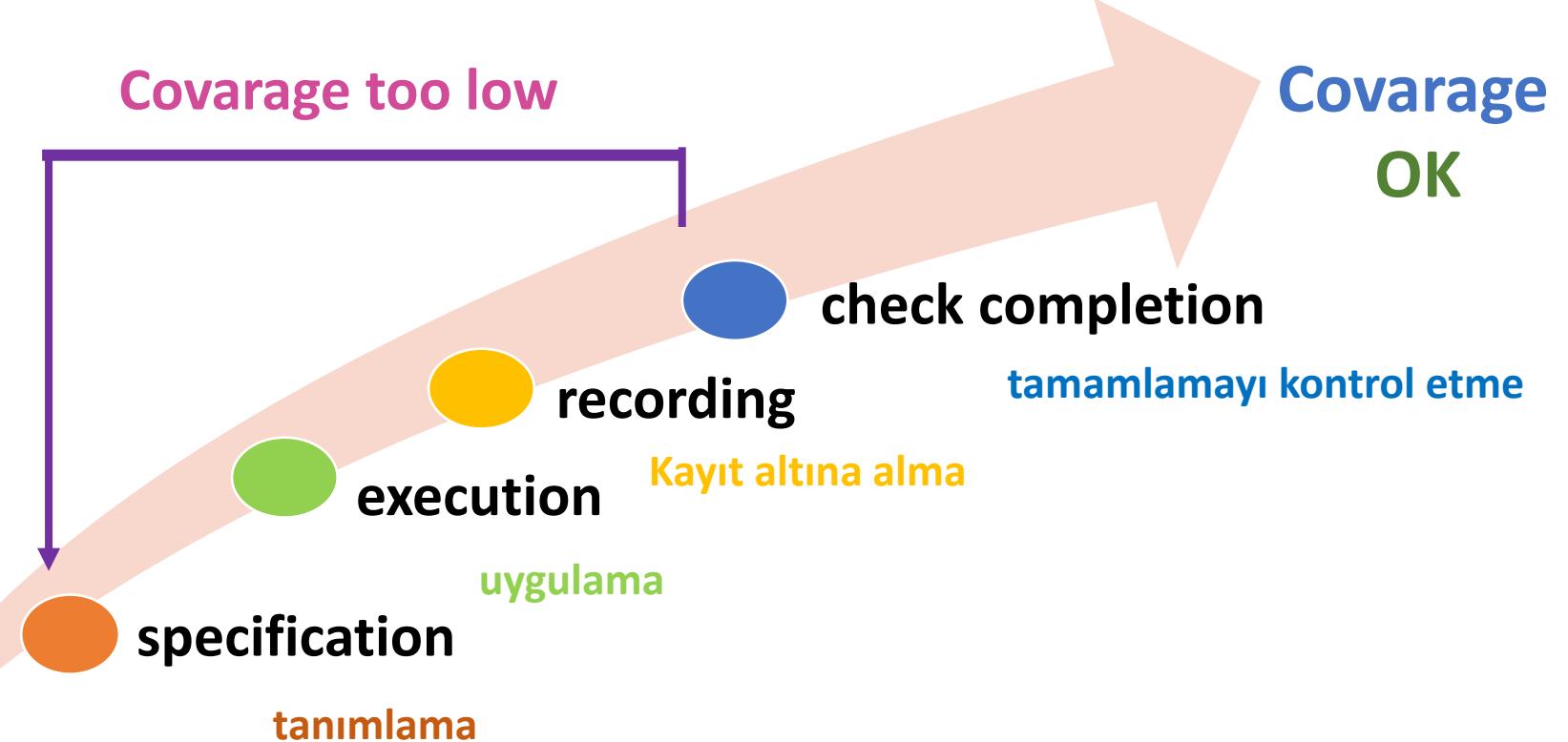
Testware, test ortamı ve test altyapısının daha sonraki test projeleri için arşivlenmesi

Projede tecrübe edilen paylaşılarak test sürecinin olgunluğunun artırılması



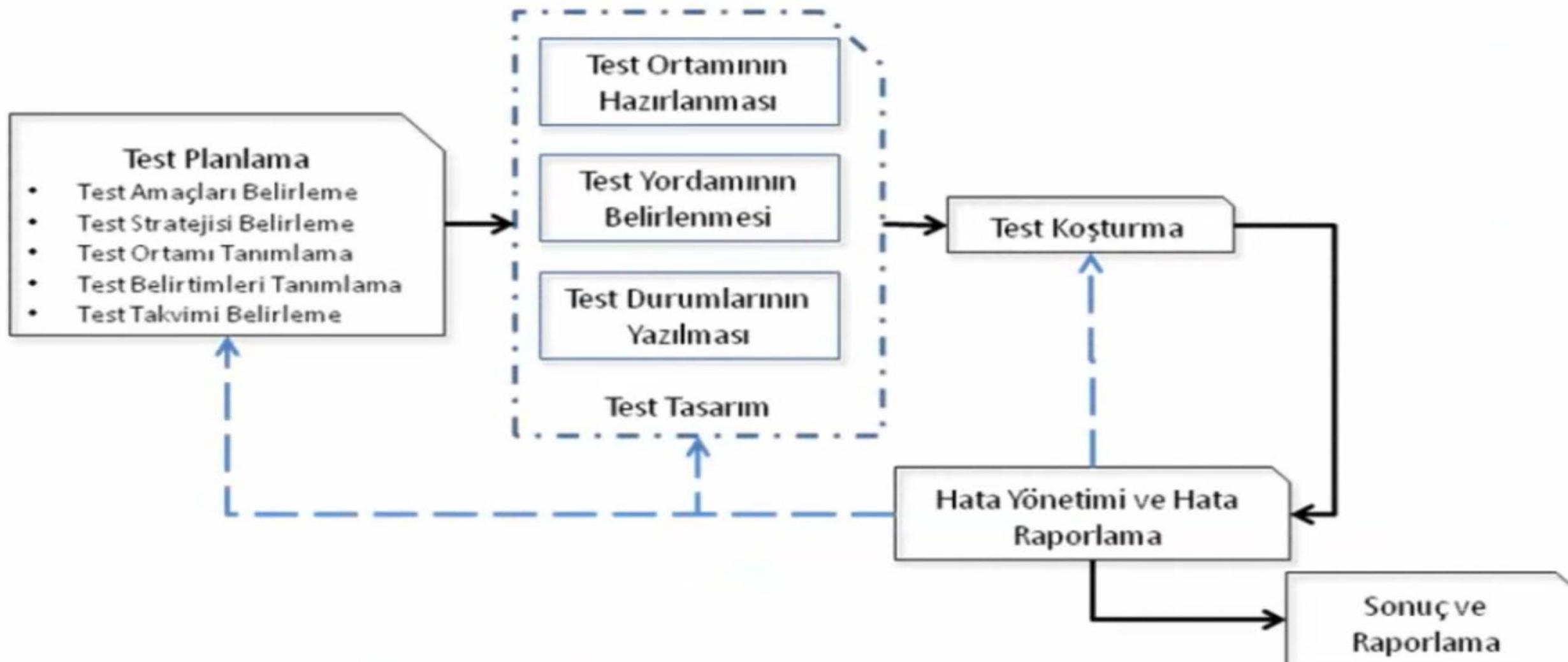
Test Sonlandırma

Covarage : kapsam alanı





Test Sürecine Farklı Bakış





Yazılım Test Süreci



Yeni Test Süreci ve Standartları



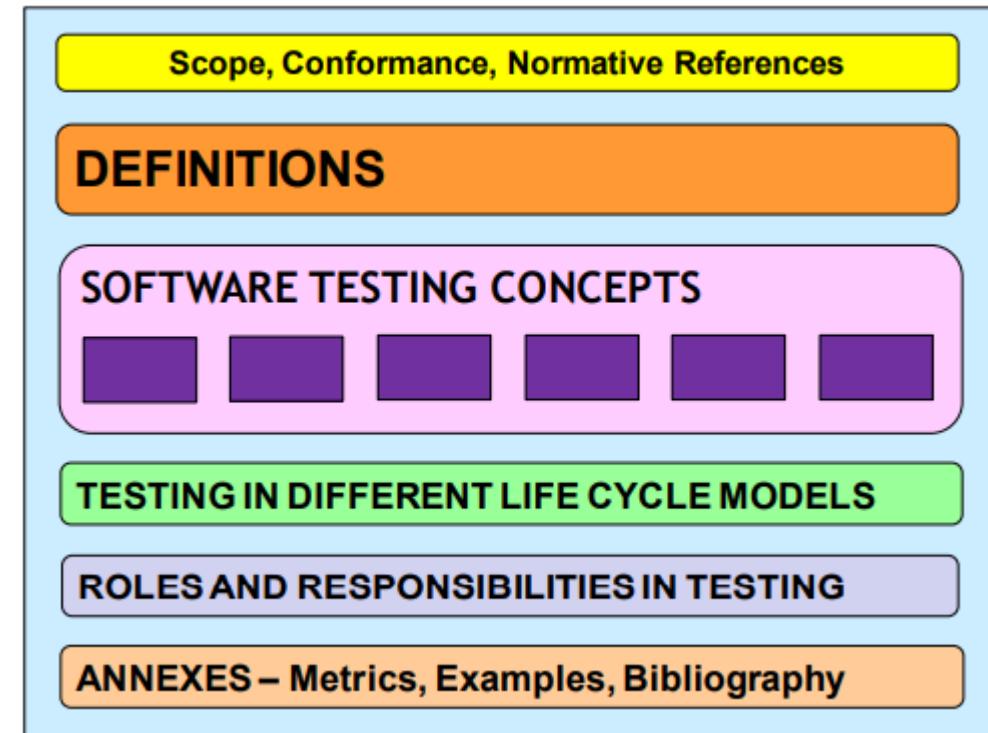


ISO/IEC/IEEE 29119 Software Testing Standards

Part 1:

- **ISO/IEC/IEEE 29119 -1:2013
Concepts and definitions**

- standartlar grubuna bir giriş sağlar ve (tutarlı) standartlar kümesinin tamamını içerir.
- setinin üzerinde bulunduğu temel test kavramlarını tanımlamanın yanı sıra diğer tüm bölümler için tanımlar standartlar oluşturulmuştur.



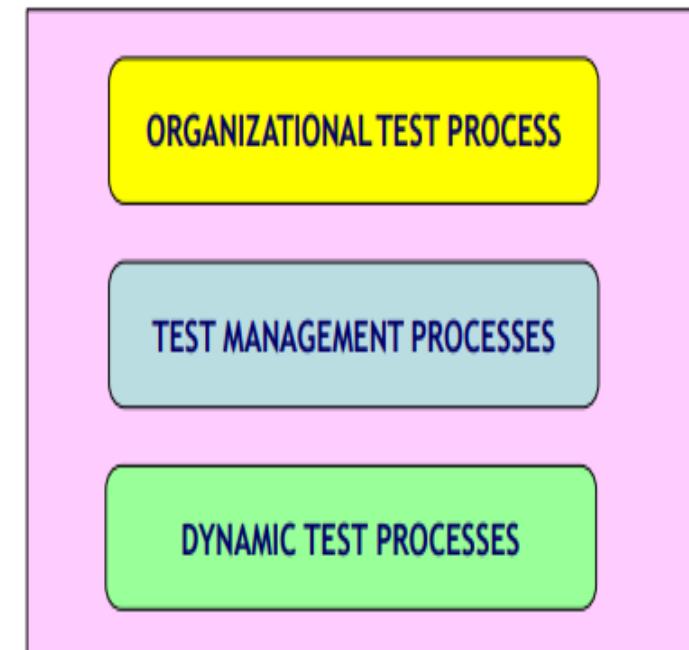


ISO/IEC/IEEE 29119 Software Testing Standards

Part 2:

- **ISO/IEC/IEEE 29119 -2:2013 Test Processes**

- herhangi bir organizasyon, proje veya test faaliyeti için yazılım testlerini yönetmek, yönetmek ve uygulamak için kullanılabilecek test süreçlerini belirtir.
- Yazılım test süreçlerini tanımlayan genel test süreci tanımlarını içerir. Süreçleri açıklayan destekleyici bilgilendirici diyagramlar da sağlanmaktadır.
- tüm yazılım geliştirme yaşam döngüsü modellerinde test için geçerlidir.
- özellikle yazılım testini yönetmek, yönetmek ve uygulamaktan sorumlu olanlar olmak üzere test uzmanları, test yöneticileri, geliştiriciler ve proje yöneticilerine yönelik ancak bunlarla sınırlı değildir.



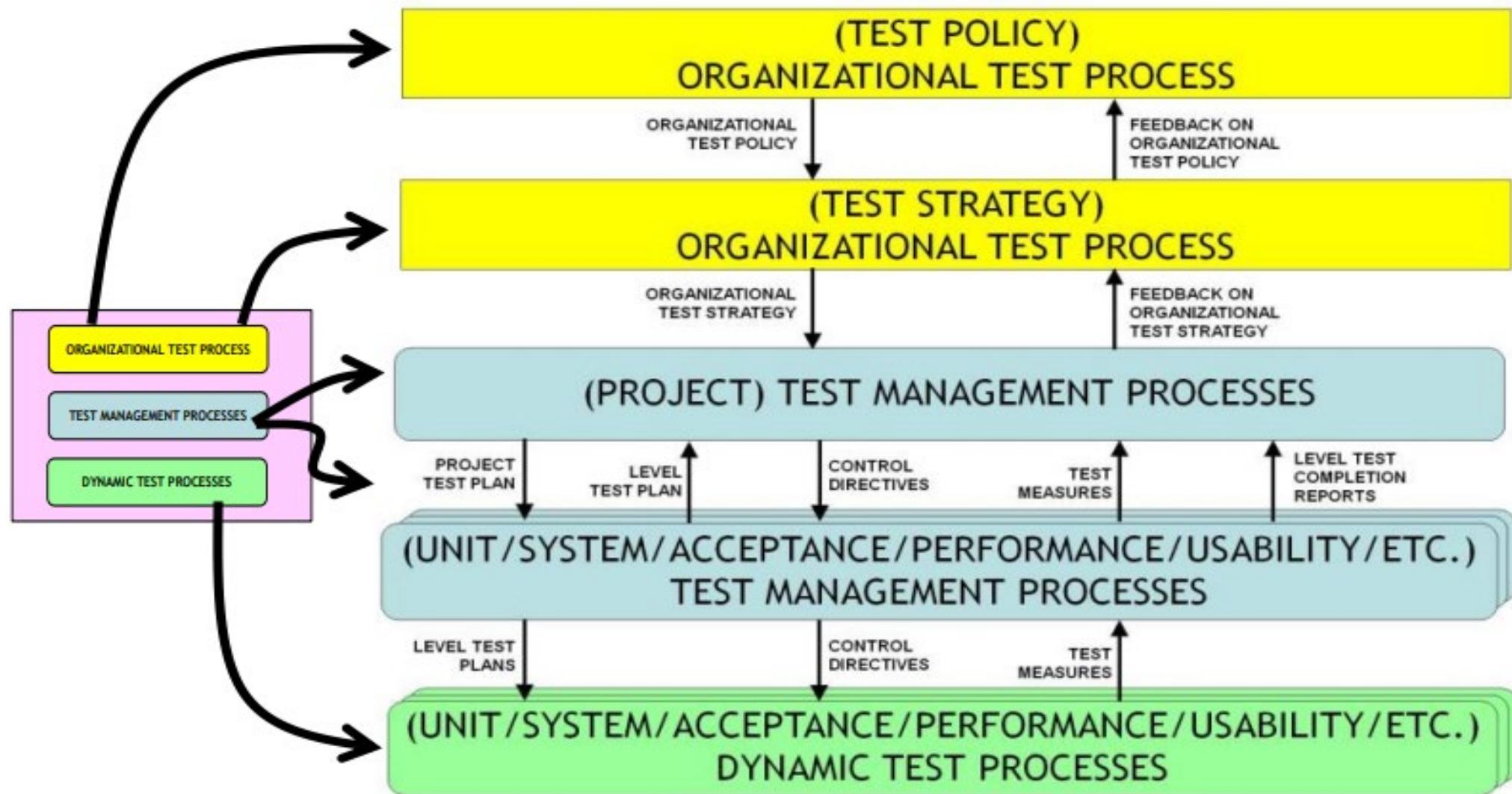


Test Sürecini Etkileyen Faktörler

- Yazılım geliştirme yaşam döngüsü modeli ve kullanılan proje metodolojileri
- Ürün ve proje riskleri
- İş alanları
- Aşağıdakileri içeren ancak bunlar ile sınırlı olmayan operasyonel kısıtlamalar
 - Bütçe- kaynak
 - Karmaşıklık
 - Sözleşme- yasal gereklilikler
 - Organizasyon politikaları ve uygulamaları
 - Gerekli iç ve dış standartlar

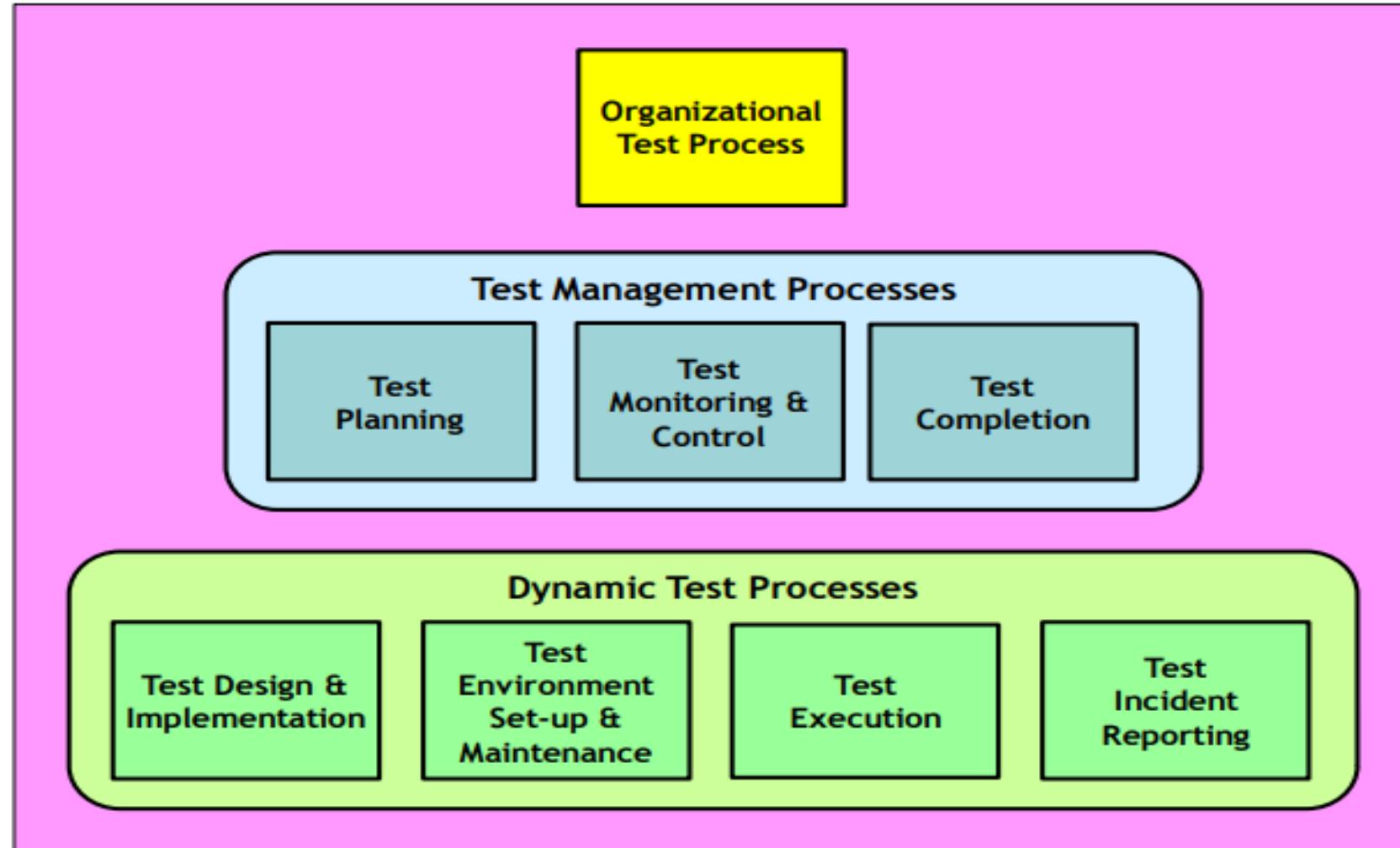


ISO/IEC/IEEE 29119 Software Testing Standards





ISO/IEC/IEEE 29119 Software Testing Standards





ISO/IEC/IEEE 29119 Software Testing Standards

Part 3:

- **ISO/IEC/IEEE 29119 -3:2013**

Test documentation

- Bu belge, herhangi bir organizasyon, proje veya test faaliyeti için kullanılabilecek yazılım testi belge şablonlarını belirtir. ISO/IEC/IEEE 29119-2'de belirtilen süreçlerin çıktısı olan test belgelerini açıklar.
- Bu belge, tüm yazılım geliştirme yaşam döngüsü modellerinde test için geçerlidir. Bu belge, özellikle yazılım testini yönetmek, yönetmek ve uygulamaktan sorumlu olanlar olmak üzere test uzmanları, test yöneticileri, geliştiriciler ve proje yöneticilerine yönelik ancak bunlarla sınırlı değildir.

**Scope, Conformance,
Normative References**

TEST DOCUMENTATION

ANNEXES - EXAMPLES



ISO/IEC/IEEE 29119 Software Testing Standards

Part 4:

• ISO/IEC/IEEE 29119 -4:2015

Test techniques

- ISO/IEC/IEEE 29119-2'de tanımlanan test tasarıımı ve uygulama sürecinde kullanılabilecek test tasarım tekniklerini tanımlar.

- Her teknik, ISO/IEC/IEEE 29119-2'de tanımlanan, test tasarımı ve uygulama sürecini takip eder. Bu belge, test uzmanları, test yöneticileri ve geliştiriciler, özellikle de aşağıdakilerden sorumlu olanlar için tasarlanmıştır, ancak yazılım testlerini yönetmek ve uygulamak bunlarla sınırlı değildir.

Scope, Conformance, Normative References

TEST DESIGN TECHNIQUES

BLACK BOX

WHITE BOX

TEST COVERAGE MEASUREMENT

ANNEXE – EXAMPLE APPLICATION OF TECHNIQUES

ANNEXE – TESTING OF QUALITY CHARACTERISTICS

ANNEXE – SELECTION OF TECHNIQUES

ANNEXE – TEST TECHNIQUE EFFECTIVENESS



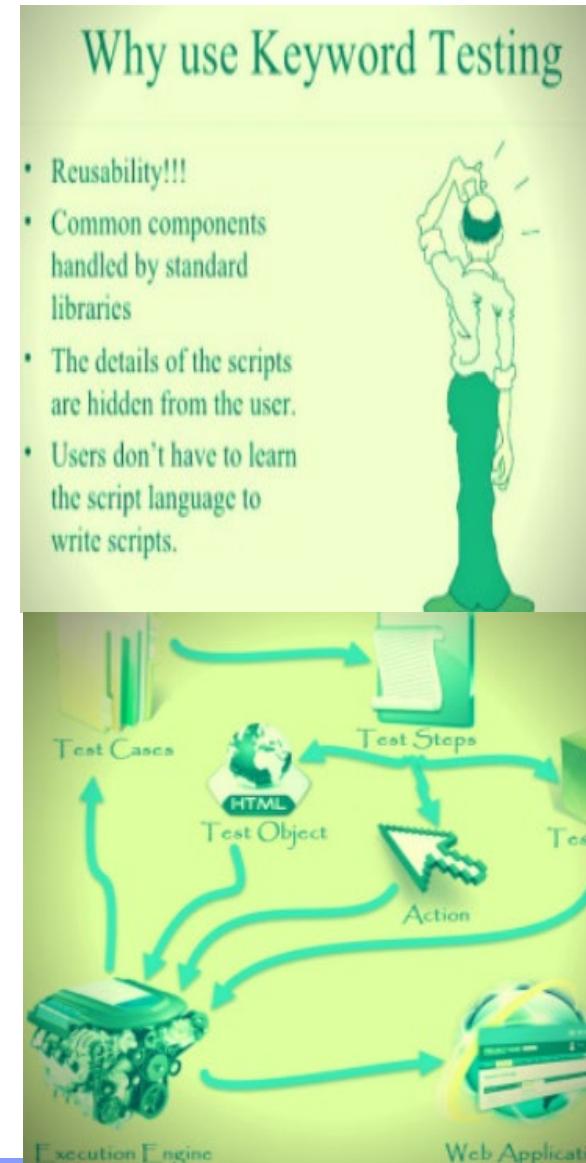
ISO/IEC/IEEE 29119 Software Testing Standards

Part 5:

- **ISO/IEC/IEEE 29119 -5:2016**

Keyword-Driven Testing

- Bu standart, anahtar kelimeye dayalı test spesifikasyonları oluşturmak, karşılık gelen çerçeveler oluşturmak veya anahtar kelimelere dayalı test otomasyonu oluşturmak isteyenler için geçerlidir.
- Otomatikleştirilmiş test senaryosu yürütme görevini otomasyon çerçevesinden izole eden yeni nesil test otomasyonu yaklaşımıdır .





Test Psikolojisi



Test Psikolojisi

Neden Test?

- Güven inşa etmek için
- Yazılımın doğru çalıştığını ispat etmek için
- Gereksinimlere uygunluğunu göstermek için
- Hata var ise bulmak için
- Masrafları azaltmak için
- Sistemin kullanıcı ihtiyaçlarını karşıladığılığını görmek için
- Yazılımın kalitesini sürdürmek için





Test Psikolojisi

Geleneksel Test Yaklaşımı ile

- ✓ Yazılımın yapması gereken özellikleri
- ✓ Yazılımın yapmadığı özellikleri saptanır

Sonuç: çalışıyor mu?

Başarısı: çalışanlar bulunur

Kolay test caseler hazırlanır

Sonuç: Hatalar bulunamaz





Test Psikolojisi

Daha iyi test yaklaşımları ile

- ✓ Yazılımın yapmaması gereken özelliklerini
- ✓ Yazılımın neleri yapmadığı saptanır

Sonuç: defect bulunur

Başarısı: sistem hataları saptanır

Zor test caseler hazırlanır

Sonuç: Çok az hata kalır





Test Psikolojisi

Test Paradoksu

Amaç : Hata bulmaktadır..

Ama .. Hatalar güveni yıkar

Zaten amaç : güveni zedelenmek olmalıdır...

Güven zedelenmeli ki;
kullanıcıya GÜVEN verilsin...





Test Psikolojisi

Testçinin özellikleri



Test mühendisi, lideri veya yönetici sürec hakkında bilgili ve iletişim kurmakta zorlanmayan, test sürecini uygulamak gibi hedefleri olan kişiler olmalıdır.



Geliştirme ortamı kötü bile olsa test ortamı daima ideal yapıya daha yakın olmalıdır.



Test ortam ile ideal test ortamı arasındaki farkı bilen ve bu farklılıktan dolayı ortaya çıkabilecek sorunları analiz edebilen özellikte olmalıdır.





Test Psikolojisi

Testçinin İhtiyaçları

- ✓ Süreç hakkında tam bilgi
- ✓ Yazılımın içeriği hakkında bilgi
- ✓ Teste gelen kodun tamamlanmış olması
- ✓ Uzmanlık alanı olarak kabul görülmesi
- ✓ Hataları bulma hakkı
- ✓ Test planı ve detaylandırılmasında söz hakkı
- ✓ Ciddi hataları raporlama yetkisi (tekrar gözlenemeyen)
- ✓ Gelecekteki hata seviyesi hakkında öngörüde bulunma
- ✓ Test süreçlerini iyileştirme yetkisi





Test Psikolojisi

Yazılımda Hata Bulmak Kötü Bir Şey Mi?

Ne zaman bulduğuna göre değişir?

Sürümeye girmeden keşfedilen hata muhteşem bir hatadır.
Müşteriye ulaşan hata, kayıp (para, mal, can, imaj vb.) demektir.



Test Psikolojisi

Testçinin Sorumlulukları

- ✓ Test planını takip etmek
- ✓ Hataları tarafsız ve gerçekçi raporlamak
- ✓ Hatayı raporlamadan önce kontrol etmek
- ✓ Yazılımcıyı değil yazılımı test ettiğinin bilincinde olmak
- ✓ Riskleri tarafsız değerlendirmek
- ✓ Hataları önceliklendirmek (prioritise)
- ✓ Gerçekleri paylaşmak





Test Faaliyetleri ve Görevleri



Yazılım Test Süreci Nedir?

- **Yazılım testleri ne zaman başlar ?**

Yazılım geliştirildikten sonra başlar..



Test eylemi, yazılım geliştirme yaşam döngüsü içerisinde, yazılım geliştirme basamaklarında, en erken safhada başlar..





Test Süreci ve Faaliyetleri

- Aslında evrensel bir test sürecinden bahsedemeyiz...
- Yazılım testini hedefine ulaşmak için gerçekleştirilen bir dizi faaliyetten bahsederiz..
- Bu faaliyetler dizisi bir TEST SURECİNİ oluşturur..





Test Sürecindeki Faaliyetler

Test Planlama

Test İzleme ve Kontrol

Test Analizi

Test Tasarımı

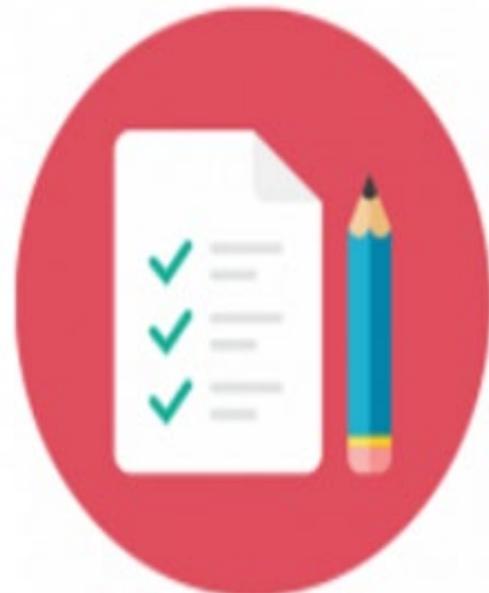
Testi Oluşturma

Test Koşturma

Testi Sonlandırma



Test Planlama



- Tüm planlar bu aşamada yapılır
- **Master test planı yazılır**
- Test Stratejisi belirlenir
- Test yaklaşımına karar verilir
- **Test başlama (entry criteria) ve Test sonlandırma (exit criteria) kriterleri belirlenir**



Test Planlama



Yazılım geliştirme stratejisi



Müşteri Gereksinimleri



Testin Amacı



Proje Alanı ve Konusu



Test İzleme ve Kontrol

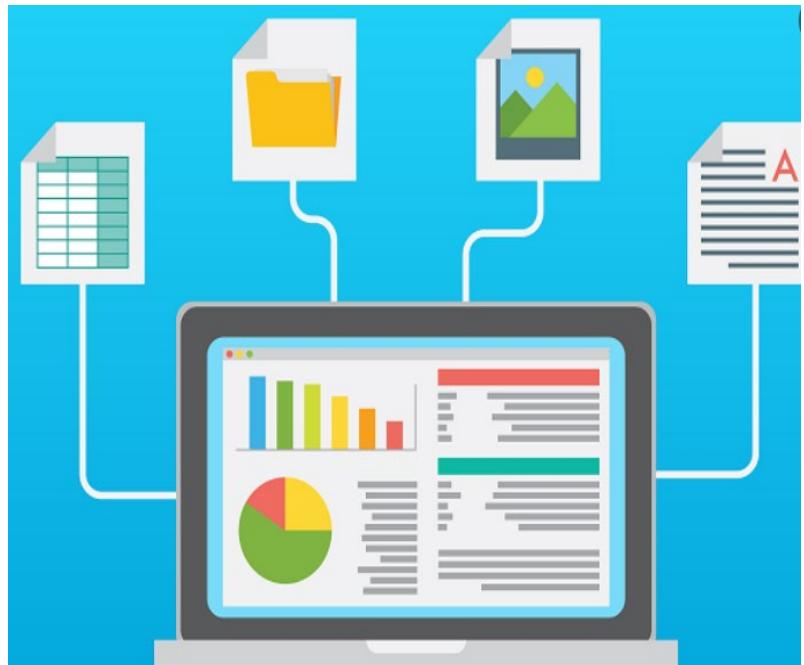


TESTING

- Test planında tanımlanan metriklerin toplanması
- Toplanan verilere göre test ilerlemesinin değerlendirilmesi
- **Agile projelerde kabul kriterlerine ulaşılıp ulaşılmadığının değerlendirilmesi**
- Daha fazla testte ihtiyaç var mı yok mu? 'nun kararına varılması aşamasıdır.



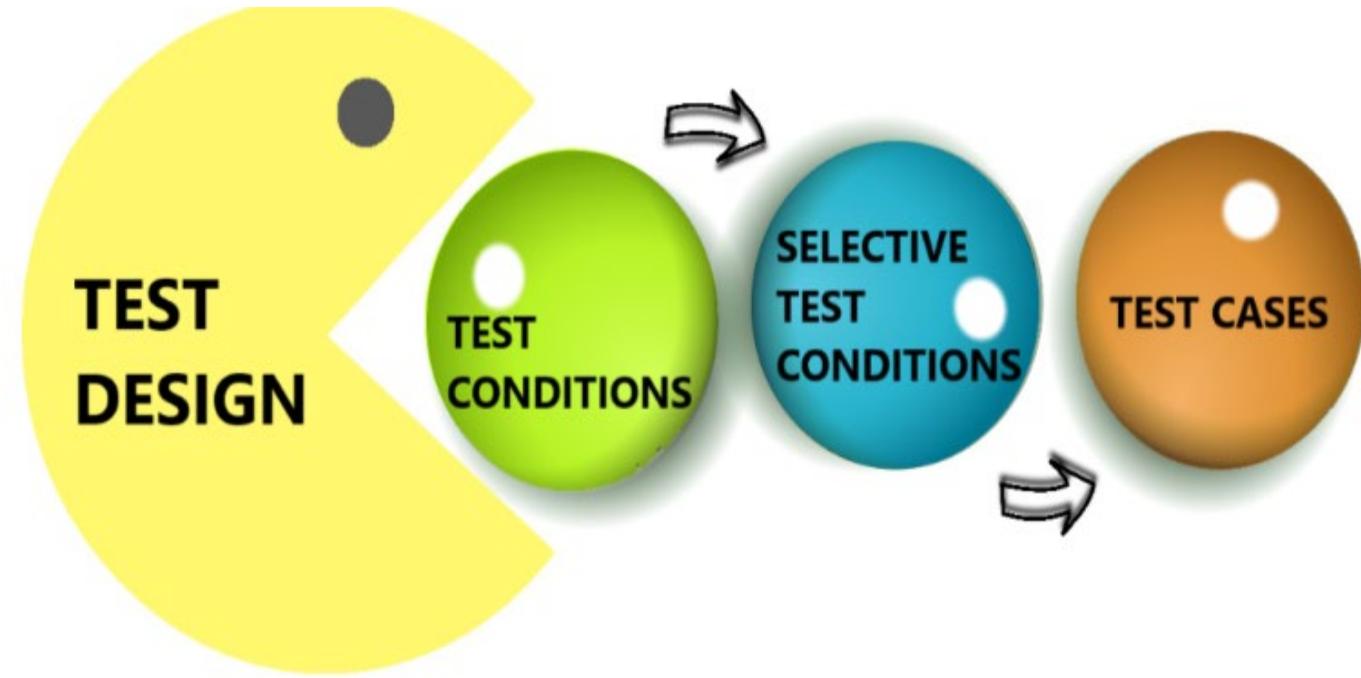
Test Analizi



- Testin altyapısının ve ortamının belirlendiği aşamadır.
- Test edilmesi planlanan sistemlerin ihtiyaçlar kapsamında test edilebilirliği belirlenir.
- **Gereksinimler belirlenir.**
- **Test edilemeyecek** olan modüller belirlenir.
- İş analisti ile beraber çalışarak yazılımın içinde bulunduğu alan analiz edilir ve müşteri gözünden bakılmaya çalışılır.



Test Tasarımı



- Bir özelliği test etmek için gereken test senaryolarının belirtilmesidir
- Test senaryoları oluşturulur ve önem sıralarına göre önceliklendirilir
- Test durumlarında kullanılmak üzere gerekli test verileri belirlenir
- Test ortamı tasarılanır ve gerekli araçlar belirlenir
- İzlenebilirlikler kurulmalıdır



Test Tasarımı



Test tasarımlı aşağıdaki ana faaliyetleri içerir;

- **Test senaryoları** ve test senaryo setlerini tasarlama ve önceliklendirme
- Test koşullarını ve test senaryolarını desteklemek için gerekli test verilerinin belirlenmesi
- Test ortamının tasarlanması ve gerekli altyapı ve araçların belirlenmesi
- Test temeli, test koşulları ve test senaryoları arasında çift yönlü izlenebilirlik yakalama ”



Test Oluşturma

- Testleri ve test prosedürleri geliştirmek
- Test prosedürlerinden belirli test takımları oluşturmak
- Otomatik test komut dosyaları oluşturmak
- Test verilerini hazırlamak ve test ortamına yerleştirmek
- Test ortamında çift yönlü izlenebilirliği sağlamak



Test Koşturma



Test execution

- Belirlenen sıra ile test senaryoları manuel yada otomasyon aracı kullanılarak çalıştırılır.
- Test sonucunda elde sonuçlar ile beklenen sonuçlar karşılaştırılır.
- Eğer karşılaştırmada bir hata bulunursa bildirilir.
- Çıkan hatalara neden olan temel sorunlar tespit edilmelidir..
- Hatalar düzeltildikçe yineleme ve onay testleri yapılır.



Testi Sonlandırma



- Planlanan çıktıların üretildiği, önemli hataların düzeltildiği ve test projesinin istenen şekilde dökümante edildiği kontrol edilir.
- Testware, test ortamı ve test altyapısı, daha sonraki test projeleri için arşivlenir.
- Projede tecrübe edilen durumların paylaşılarak kurumsal hafıza oluşturma adına geleceğe bilgi aktarımı yapılır.
- Test logları ve raporları incelenir.
- Raporlar ve logların planlama aşamasında belirlenen sonlandırma kriterlerini sağlayıp sağlamadığı kontrol edilir.
- Kontroller sonunda “**daha fazla test**” veya “**test sürecinin sonu**” kararının verildiği aşamadır.
- Test projesi özeti üst yönetime sunulur.



Yazılım Test İş Ürünleri



Test İş Ürünleri

ISO/IEC/IEEE 29119 -3:2013 Test documentation

- Test iş ürünleri rehberidir..
- Test sürecinin sonucu olarak ortaya çıkarırlar
- Test faaliyetlerine göre test iş ürünleri farklılık gösterir..

**ISO/IEC/IEEE 29119
SOFTWARE TESTING
STANDARDS**
A Practitioner's Guide*

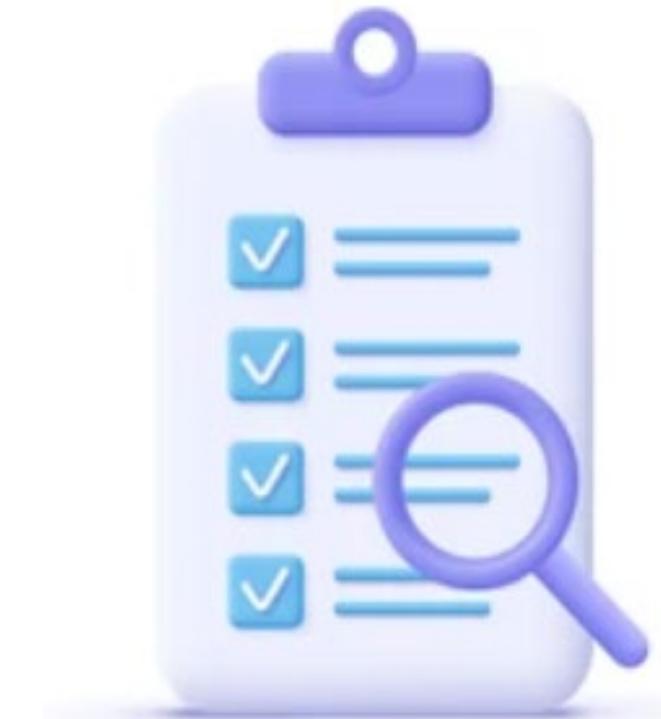
Dr Stuart Reid
STA Testing Consulting





Test Planlama İş Ürünleri

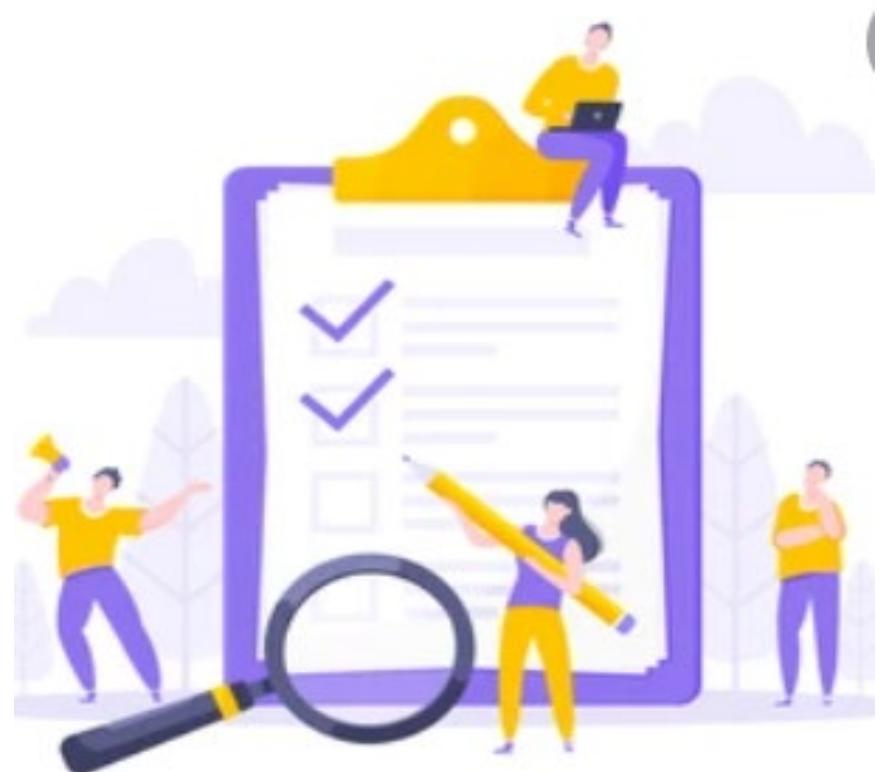
- Test Ana Planı (MTP)
- Diğer seviye test planları
 - ✓ Kabul Test Planı
 - ✓ Sistem Test Planı
 - ✓ Yazılım Test Planı
 - ✓ Sistem Entegrasyon Test Planı





Test İzleme ve Kontrol İş Ürünleri

- Test İşleyiş (progress) raporları
- Test raporları
- Test Summary Report (Test Özeti Raporu)





Test Analizi İş Ürünleri

- Test Gereksinimleri/belirtimleri
- Tanımlanmış ve önceliklendirilmiş test öğeleri





Test Tasarımı İş Ürünleri

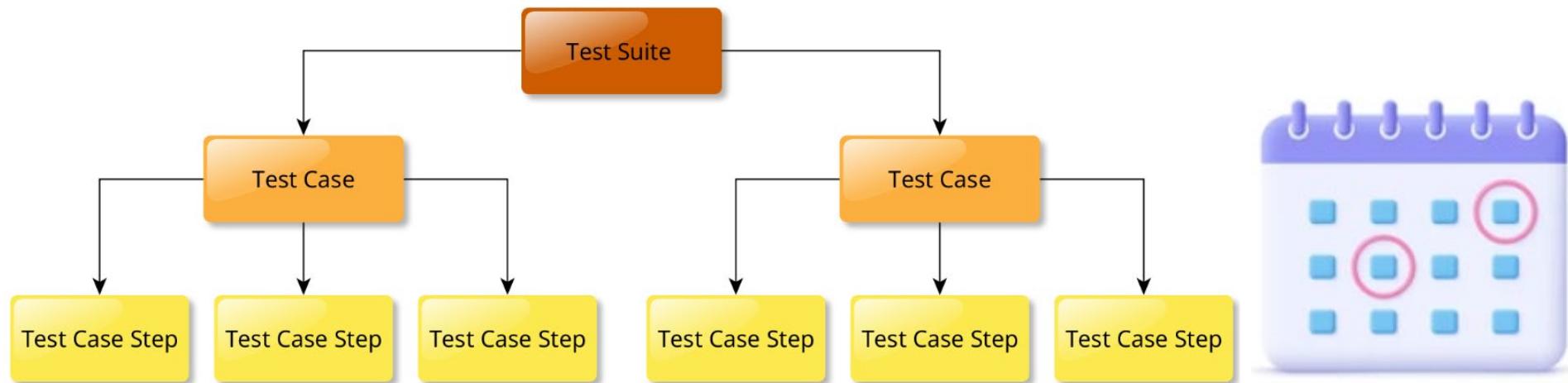
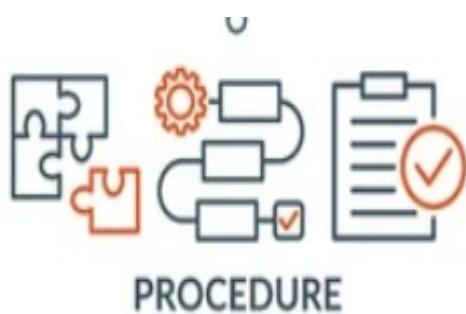
- Test durumları
- Test senaryoları
- Test verileri
- Test ortamı





Test Oluşturma İş Ürünleri

- Test prosedürleri ve bu prosedürlerin sırası
- Test suitleri (takımları)
- Test koşturma takvimi





Test Koşturma İş Ürünleri

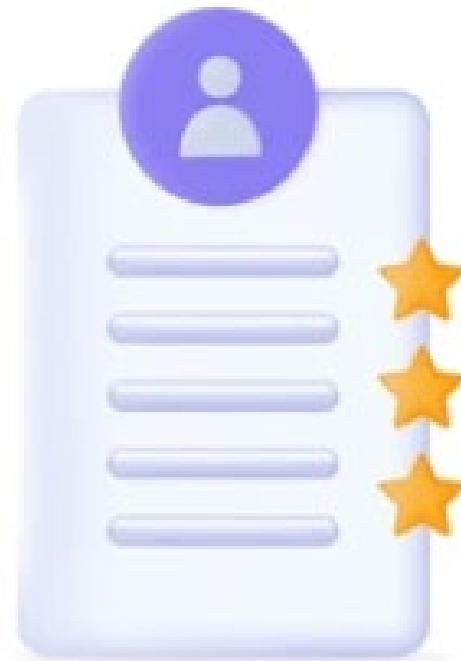
- Test durumlarının ve test prosedürlerinin statü durumu (örnek: çalışmaya hazır, başarısız, geçmiş, bloke edilmiş vb.)
- Hata raporları
- Test raporları





Test Sonladırma İş Ürünleri

- Test summary report- Test özet raporu
- İyileştirme yönünde oluşturulan raporlar





BATCH : **BATCH 59**

LESSON : **ISTQB-03**

DATE : **06.06.2022**

SUBJECT : **ISTQB**

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



Yazılım Yaşam Döngüsü Boyunca Test



Yazılım Yaşam Döngüsü Boyunca Test

Yazılım Geliştirme Modelleri

Test Seviyesi

Test Türleri

Bakım Testi

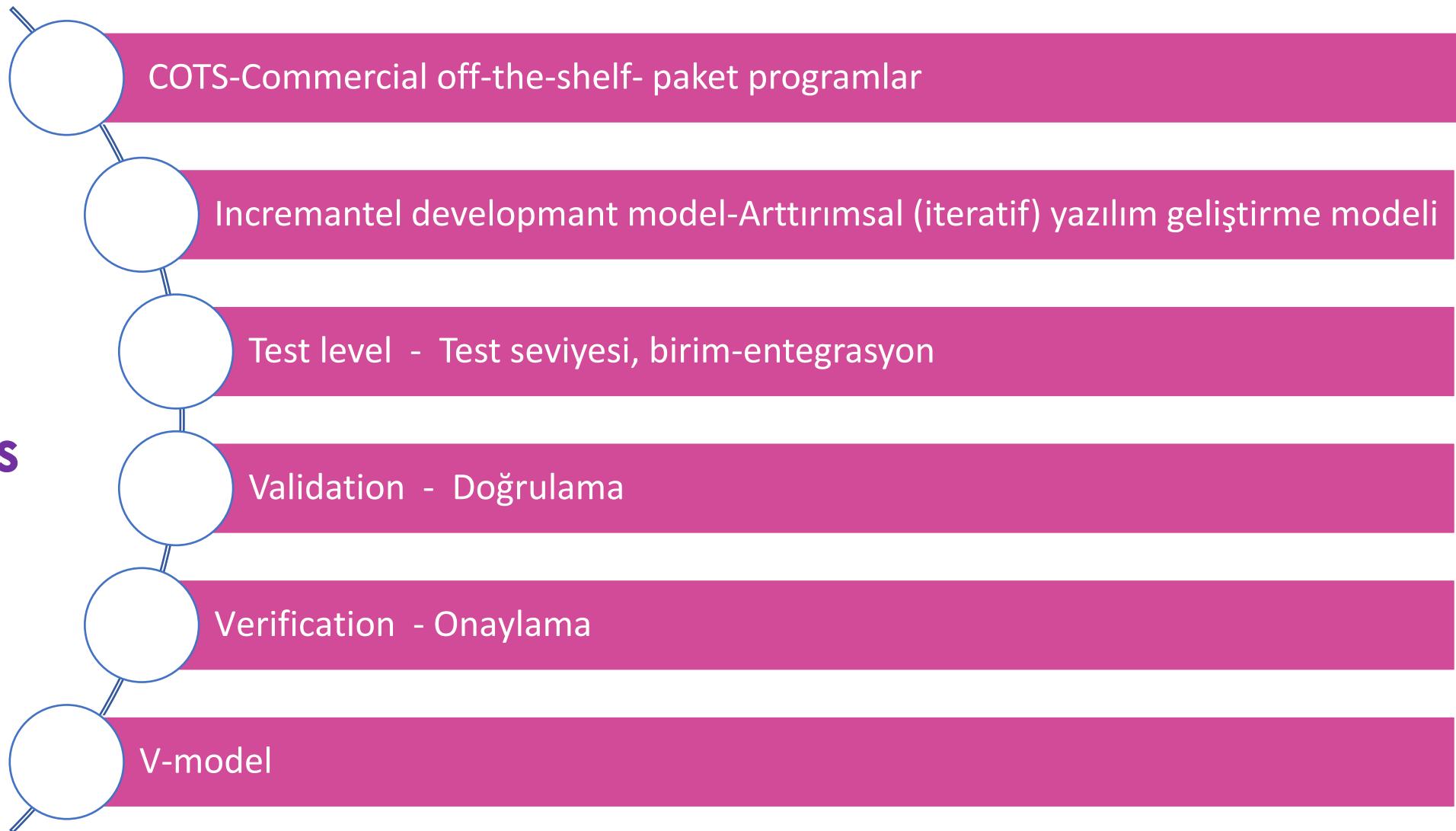


Yazılım Geliştirme Modelleri



Yazılım Geliştirme Modelleri

Keywords





Software Development Life Cycle(SDLC)

SDLC

Yazılım Geliştirme Yaşam Döngüsü

Bir yazılımın geliştirilme süreci boyunca geçirdiği tüm aşamalar yazılım yaşam döngüsü olarak tanımlanır.





SDLC Aşamaları

PLANNING and ANALYSIS

- ✓ Temel aşamasıdır.
- ✓ Kİdemli üyeleri ve müşteri tekliflerine göre yapılır.
- ✓ İhtiyaç analizi ortaya koyulur.
- ✓ Riskler belirlenir.
- ✓ Genel hatlarıyla proje planı ortaya koyulur.

PLANNING



SDLC Aşamaları

DEFINING-REQUIREMENT PHASE

- ✓ ihtiyaç analizi yapıldıktan sonra ürün gereksinimleri tanımlanır.
- ✓ DEFINING aşamasında BRD ve FRD hazırlanır.
(Business Analyst hazırlar)
- ✓ Detaylı Dokümantasyonun yapıldığı aşamadır.
- ✓ BUSINESS REQUIREMENT DOCUMENT
- ✓ FUNCTIONAL REQUIREMENT DOCUMENT





SDLC Aşamaları

DESIGNING

- ✓ Görseller ve tasarım oluşturulur
- ✓ DESIGNER (SOLUTION ARCHITECT)'ler yapar.
- ✓ Birden fazla öneri sunarlar.
- ✓ Sunulan öneriler DDS' de belgelenir.

DESIGN DOCUMENT SPESIFICATION

- ✓ Yapılan öneriler BRD' ye göre yapılır.





SDLC Aşamaları

DEVELOPMENT- BUILDING

- ✓ Developerlar yapar.
- ✓ Kodlama yönergesine uyularak yapılır.
- ✓ Gerçek geliştirme bu aşamada başlar.
- ✓ Ürün inşa edilir.
- ✓ DDS' ye göre programlama kodu üretilir.

DEVELOPMENT



SDLC Aşamaları

TESTING

- ✓ Test, kodlama tamamlandıktan ve modüller test için yayınlandıktan sonra başlar. Bu aşamada, geliştirilen yazılım kapsamlı bir şekilde test edilir ve bulunan tüm kusurlar, düzeltmeleri için geliştiricilere atanır.
- ✓ Yeniden test, regresyon testi, yazılımın müşterinin bekłentisine göre olduğu noktaya kadar yapılır. Test uzmanları, yazılımın müşterinin standardına uygun olduğundan emin olmak için SRS (Software Requirements Specification) belgesine başvurur.





SDLC Aşamaları

DEPLOYMENT

- ✓ Ürün test edildikten sonra, üretim ortamında veya ilk olarak devreye alınır.
- ✓ Yani dağıtım aşamasında ise test süreçlerinden sonra hiçbir hata ve bug barındırmayan sistem yayınlanır ve var ise dağıtım sorunları kontrol edilir.





SDLC Aşamaları

MAİNTEANCE

- ✓ Sahadan veya kullanıcılarından gelen geri bildirim ile yazılım üzerinde düzeltici, uyarıcı, iyileştirici ve önleyici bakımlar gerçekleştirilmektedir.
- ✓ Bütün bu adımlar, farklı metodolojiler ile farklı şekillerde uygulanarak birden fazla yazılım geliştirme modeli ortaya çıkarmıştır.





SDLC Modelleri

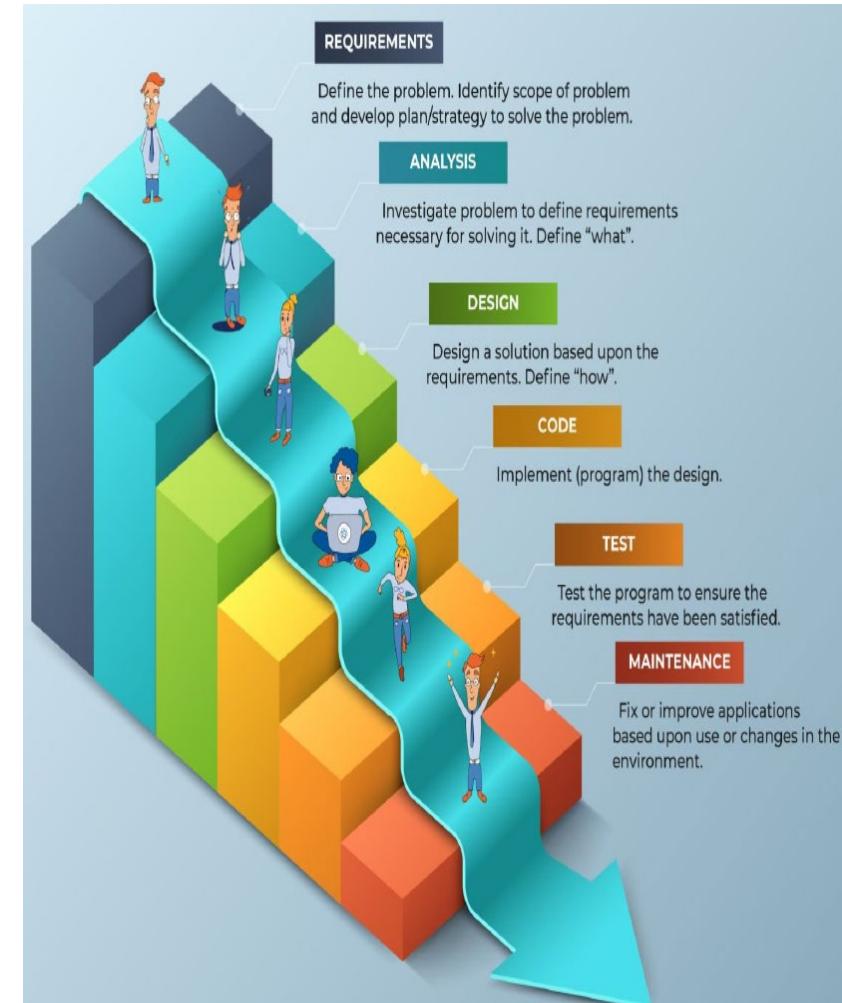
- Waterfall Model
- Incremental Model
- Iterative Model
- V model
- Spiral Model
- Agile Model
 - Key Agile concepts (User story ,Burn Down chart)
 - Agile Methodologies
 - SCRUM
 - KANBAN
 - EXTREME Programming



SDLC Modelleri - Waterfall Model

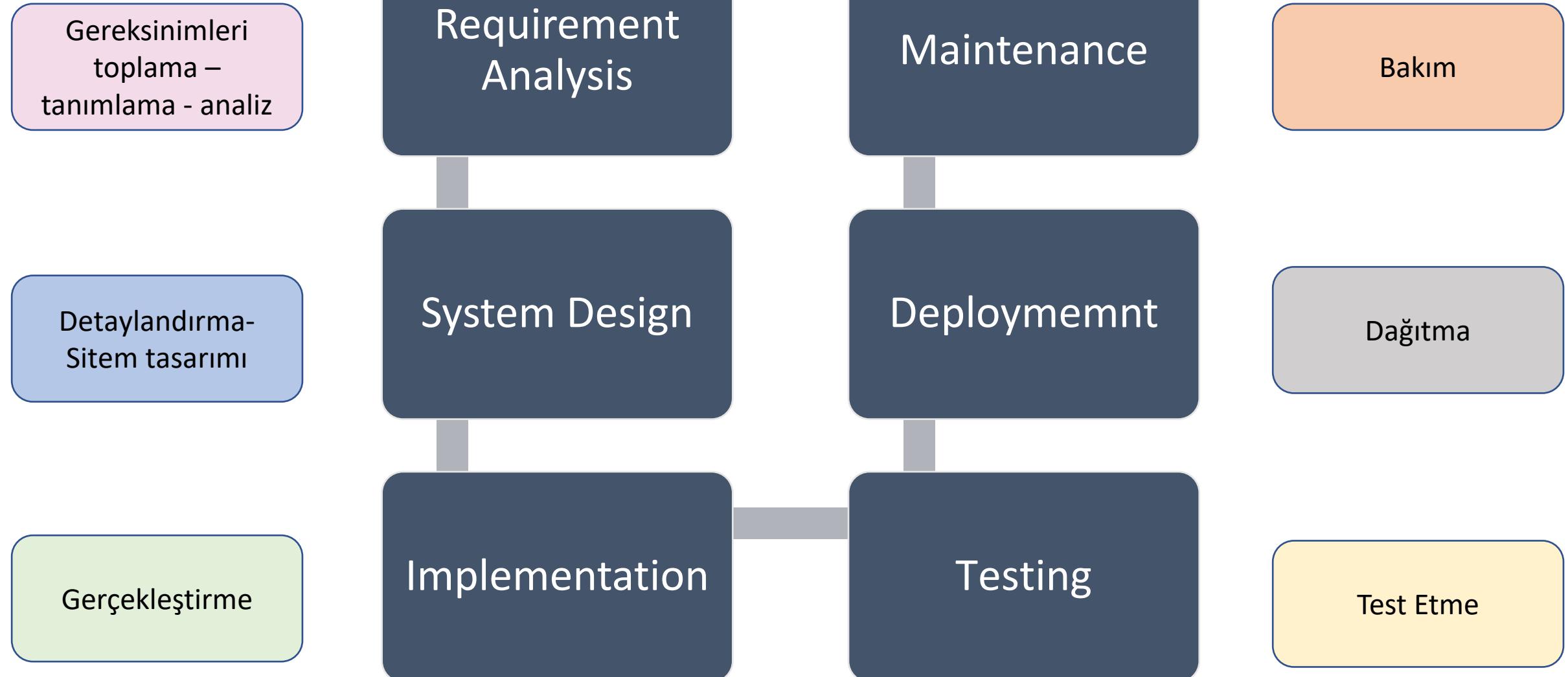
Waterfall Model -Şelale Modeli

- En bilinen ve 50'li yıllarda bu yana yaygın bir şekilde kullanılan bir modeldir.
- SDLC' de kullanılan **ilk modeldir. Doğrusal sıralı model** olarak da bilinir.
- Bu modelde, **bir aşamanın sonucu, bir sonraki aşamanın girdisidir**. Bir sonraki aşamanın geliştirilmesi ancak önceki aşama tamamlandığında başlar.





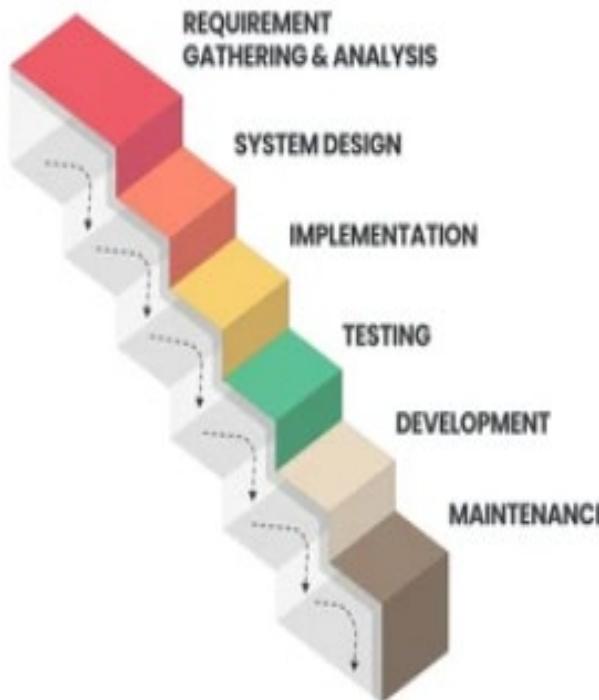
SDLC Modelleri - Waterfall Model





SDLC Modelleri - Waterfall Model

Avantajları:



- Waterfall modeli, **kolay anlaşılabilen basit modeldir** ve tüm aşamaların **adım adım yapıldığı** modeldir.
- Her aşamanın **teslim edilecekleri** iyi **tanımlanmıştır** ve bu karmaşıklığa yol açmaz ve projeyi **kolayca yönetilebilir** hale getirir.



SDLC Modelleri - Waterfall Model

Dezavantajları:

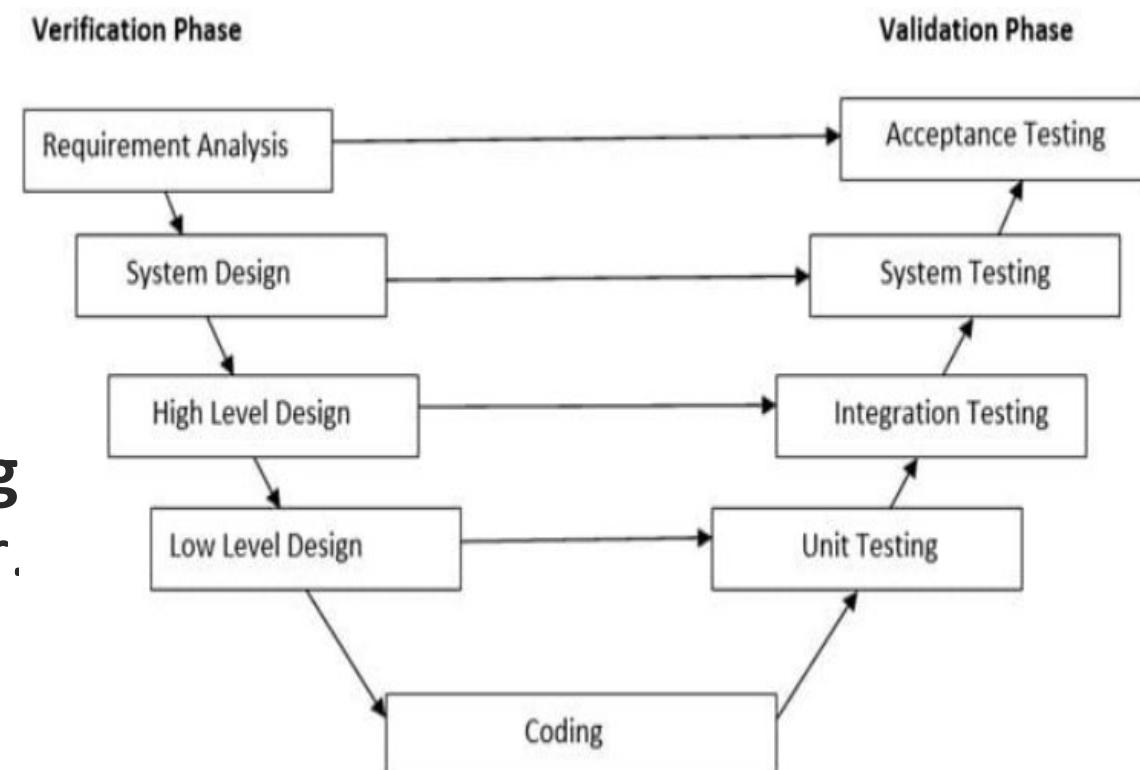
- Waterfall modeli **zaman alıcıdır** ve kısa süreli projelerde kullanılamaz çünkü bu modelde devam eden aşama tamamlanana kadar **yeni bir aşama başlatılamaz**.
- Waterfall modeli, **gereksinimi belirsiz olan veya gereksinimin değişmeye devam ettiği** projeler için kullanılamaz, çünkü **bu model gereksinim toplama ve analiz aşamasında ihtiyacın net olmasını beklemektedir** ve sonraki aşamalarda herhangi bir değişiklik, **maliyetin yükselmesine** neden olacaktır. Tüm aşamalarda değişiklikler gerekli olacaktır.



SDLC Modelleri - V Modeli

V Modeli-Validation ve Verification

Bu modelde bir tarafta **validation** aşamaları diğer tarafta ise **verification** aşamaları bulunmaktadır ve ikisini **Coding (Kodlama)** aşaması birleştirir durumdadır.

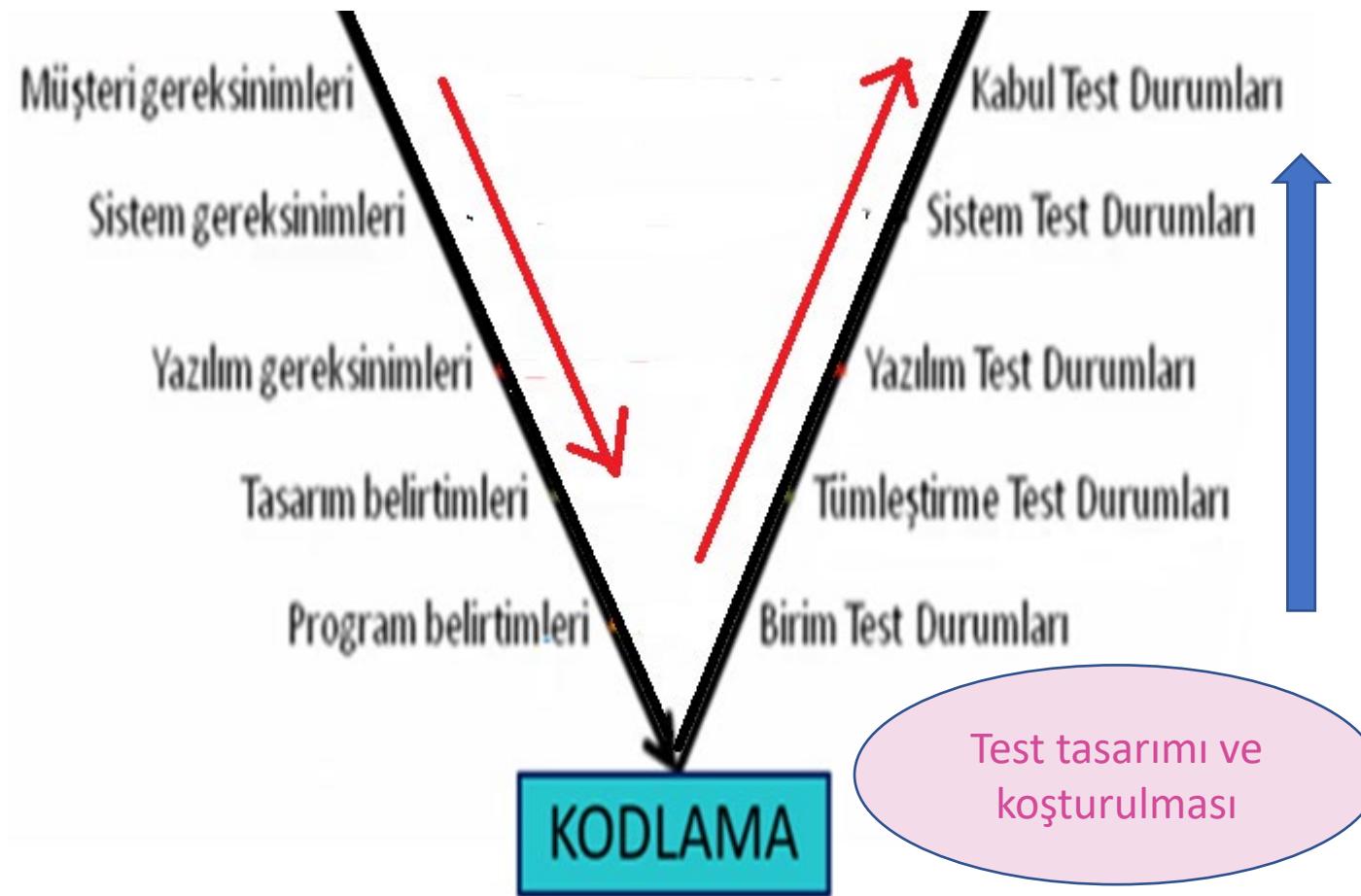




SDLC Modelleri - V Modeli

V Modeli - Geç Tasarım

Eğer gereksinim analizinden başlayıp, kodlama kısmına kadar inen daha sonra da **kabul testine kadar çıkan** bir yaklaşım kabul edilirse bu “**geç tasarım**” olarak adlandırılır.

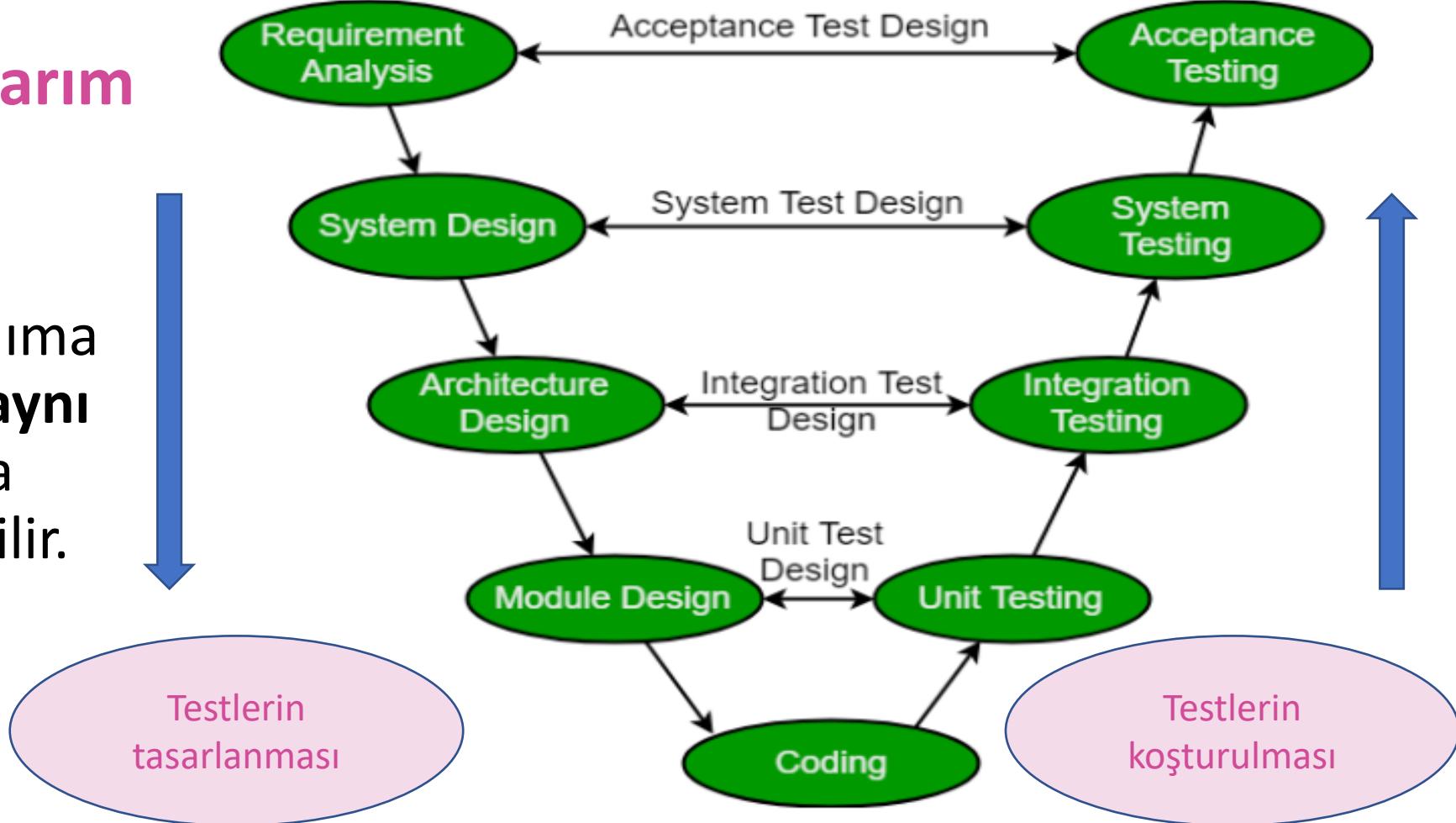




SDLC Modelleri - V Modeli

V Modeli - Erken Tasarım

Öte yandan geliştirme adımlarındaki her bir adıma **karşılık gelen testlerin aynı anda** yapıldığı yaklaşımı **“erken tasarım”** adı verilir.





SDLC Modelleri - V Modeli

Avantajları:

- ✓ **Basit ve kolay anlaşılır** bir modeldir.
- ✓ V modeli yaklaşımı, **ihtiyacın tanımlandığı** ve erken aşamada donduğu küçük projeler için iyidir.
- ✓ **Yüksek kaliteli bir ürünle sonuçlanan** sistematik ve **disiplinli** bir modeldir



SDLC Modelleri - V Modeli

Dezvantajları:

- ✓ V şeklindeki model, **devam eden projeler için** **iyi değildir**.
- ✓ Daha sonraki aşamadaki **gereksinim değişikliği çok yüksek maliyetli** olacaktır.



SDLC Modelleri - Agile Methodology

Agile(Çevik) Methodology

- Herhangi bir projenin SDLC süreci devam ederken, geliştirme ve testin sürekli olarak etkileşimde bulunmasını destekler.





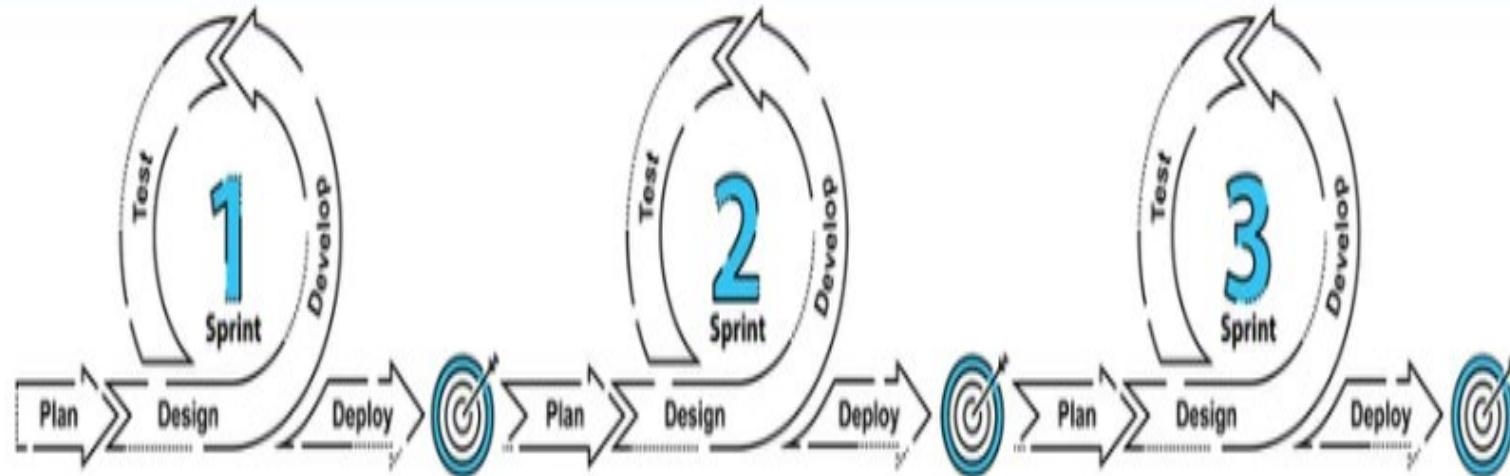
SDLC Modelleri - Agile Methodology

Uygulama şekillerine göre agile modeller

- ✓ Uç sınırsal programlama (Extreme Programming-XP)
- ✓ Agile Birleştirilmiş Süreç (Agile Unified Process)
- ✓ **Scrum*****
- ✓ Test Gündümlü Geliştirme (Test Driven Development)
- ✓ Kanban
- ✓ Özellik Gündümlü Geliştirme (Feature Driven Programming)



SDLC Modelleri - Agile Methodology



AGILE METHODOLOGY

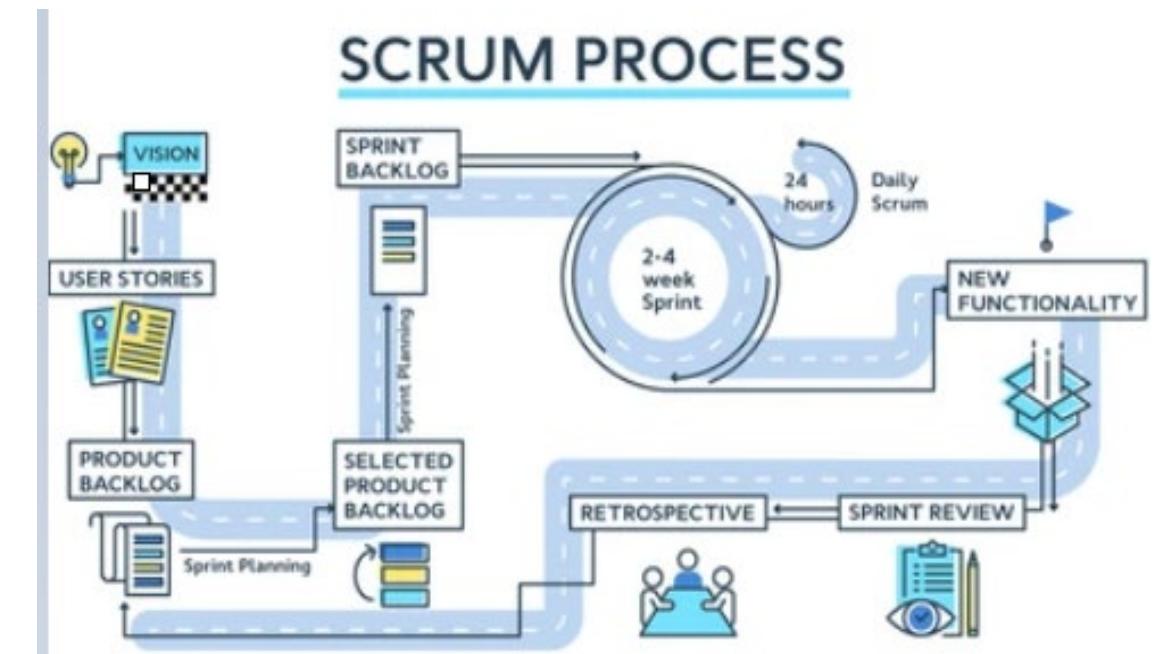
- Bu metotda tüm proje küçük artımlı yapılara bölünmüştür, bu yapıların hepsi yinelemelerde sağlanmıştır ve her bir yineleme 2–3 hafta sürer.



SDLC Modelleri - Agile Methodology

SCRUM

- User story
- Product Backlog
- Sprint Backlog
- Scrum Events





SDLC Modelleri - Agile Methodology

Dezvantajları:

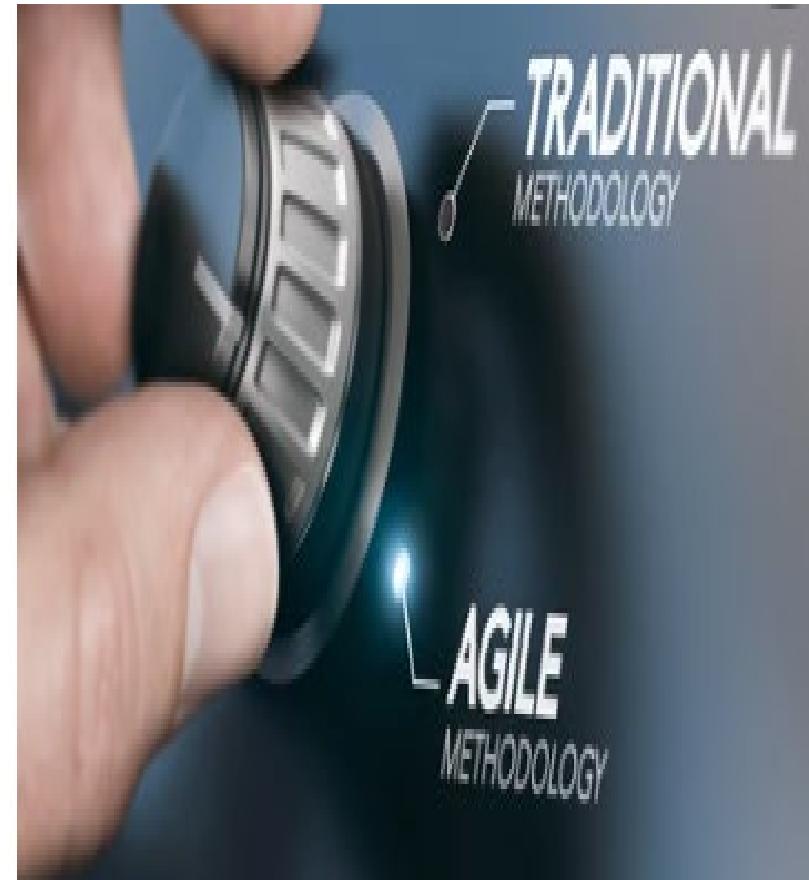
- ✓ Belge eksikliği
- ✓ Agile, deneyimli ve yüksek vasıflı kaynaklara ihtiyaç duyar.
- ✓ Bir müşteri ürünün tam olarak nasıl olmasını istediği konusunda net değilse, proje başarısız olur.



SDLC Modelleri - Agile Methodology

Avantajları:

- ✓ Değişikliklere uyum sağlamak için daha fazla esneklik sağlar.
- ✓ Yeni özellik kolaylıkla eklenebilir.
- ✓ Her aşamada geri bildirim ve öneriler alındığı için müşteri memnuniyeti vardır



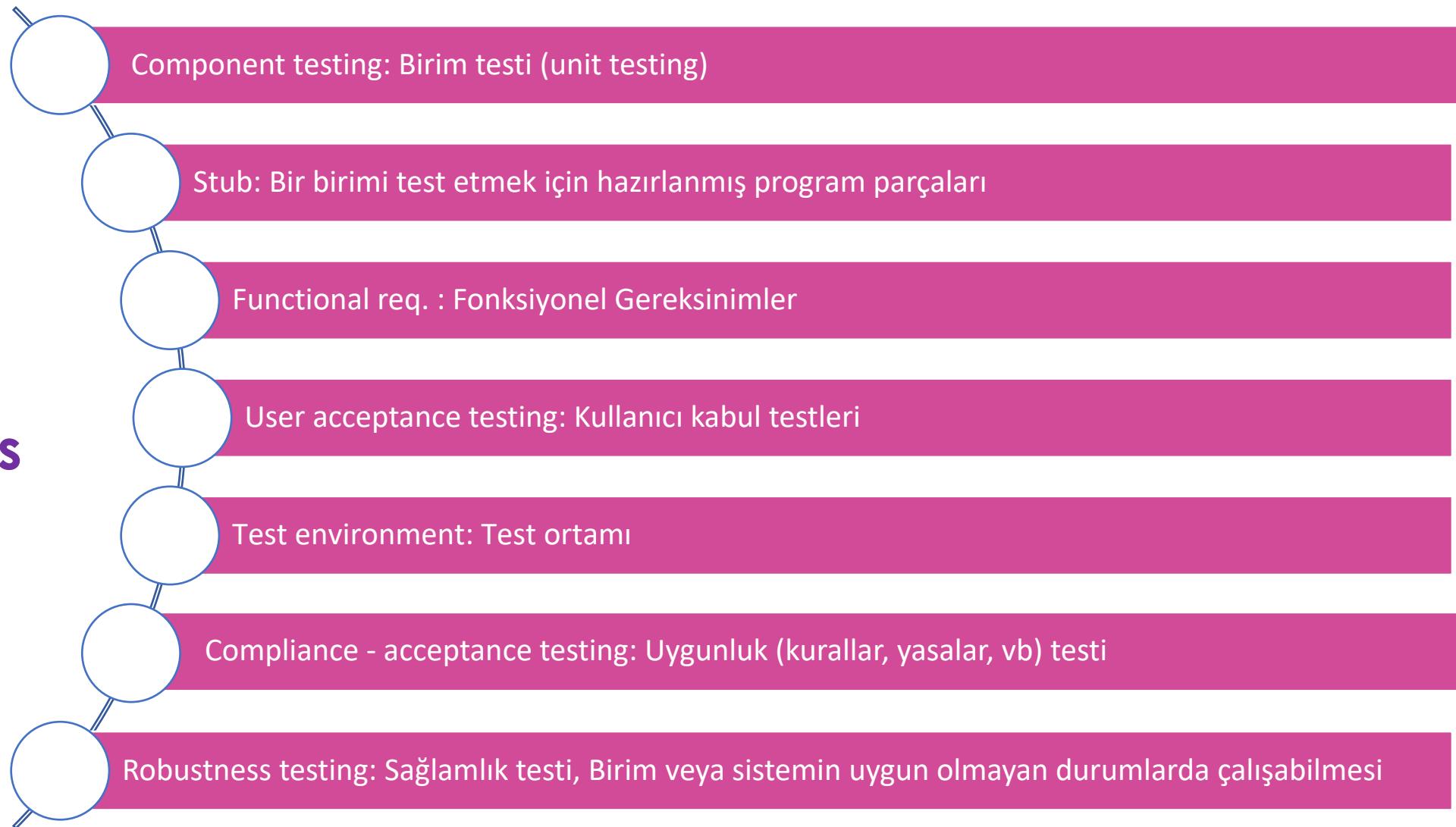


Yazılım Test Seviyeleri



Yazılım Test Seviyeleri

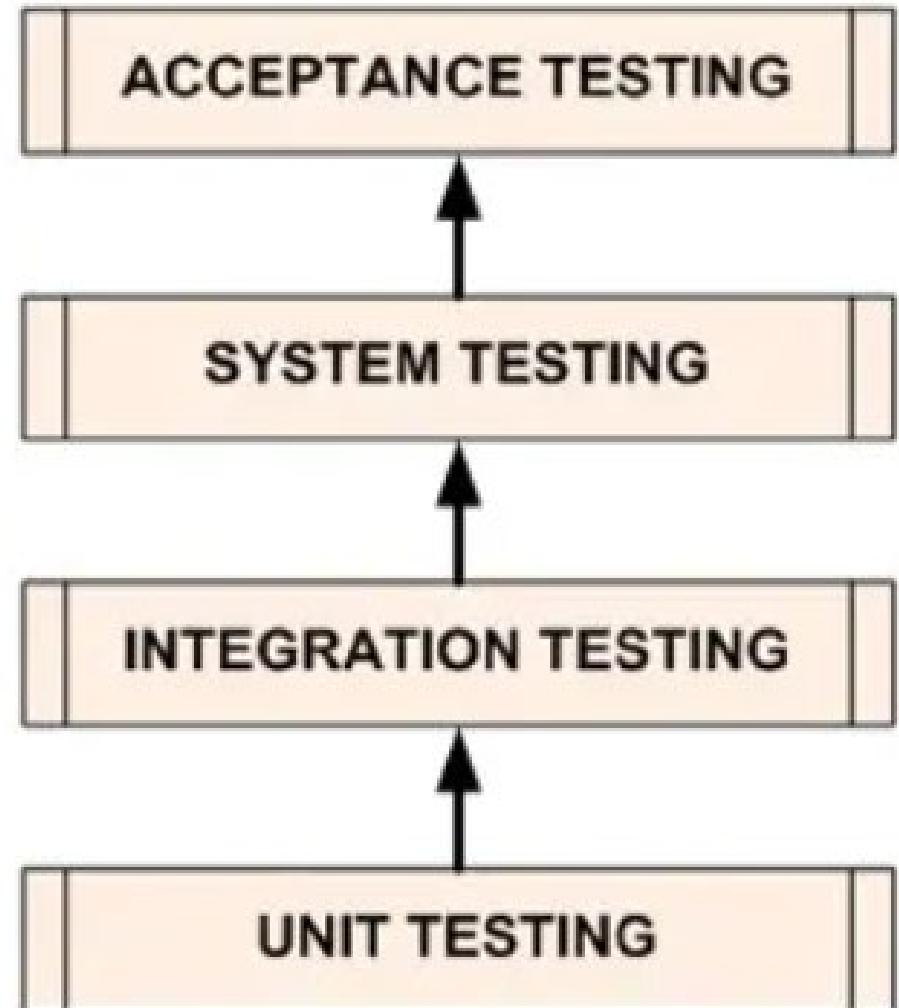
Keywords





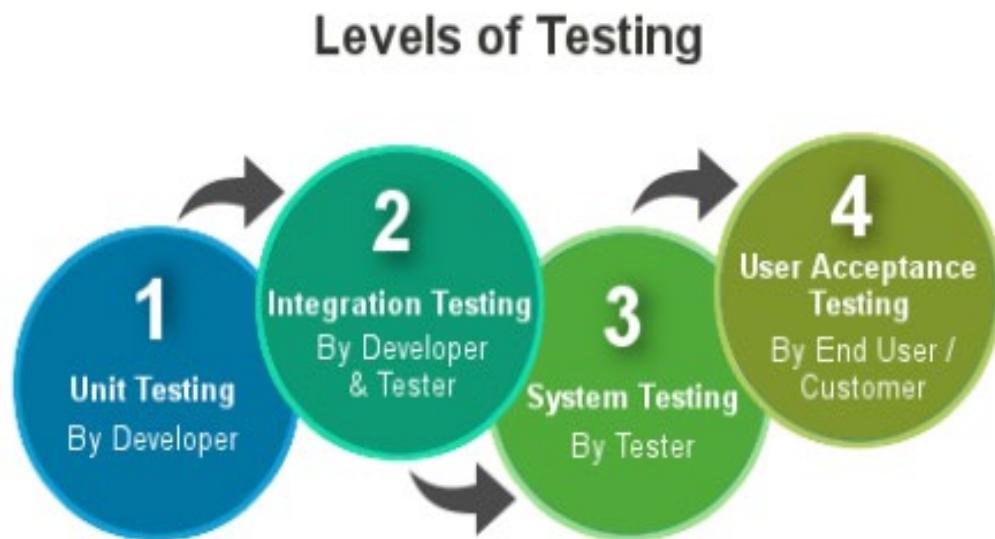
Yazılım Test Seviyeleri

- ✓ **Test seviyeleri** temel olarak eksik alanları tanımlamak ve SDLC aşamaları arasındaki tekrarı önlemek içindir.
- ✓ SDLC gereksinim analizi, tasarım, geliştirme, test ve bakım olmak üzere bir çok aşamadan oluşmaktadır. Her aşama da testten geçer. Bu nedenle çeşitli test seviyeleri vardır.





Yazılım Test Seviyeleri



Başlıca dört test seviyesi vardır.
Bunlar;

- ✓ **birim testi**
(unit testing)
- ✓ **entegrasyon testi**
(integration testing)
- ✓ **sistem testi**
(system testing)
- ✓ **kabul testidir**
(acceptance testing)



Yazılım Test Seviyeleri - Unit Testing

Birim Testleri-Unit Test

- ✓ Yazılım ekibi tarafından yapılan beyaz kutu testleridir.
- ✓ Test uzmanı dışında icra edilir.
- ✓ Metot, fonksiyon veya prosedür gibi **bir kod parçasının** kendisinden beklenen işlevselligi **doğru olarak yerine getirip getirmedigini ölçen ve yazılım geliştiriciler** tarafından gerçekleştirilen test tipidir.

Component

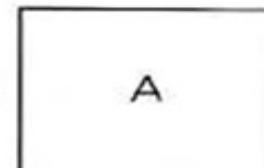
Module

Program

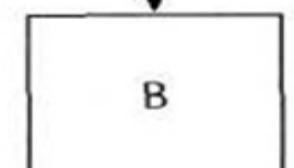
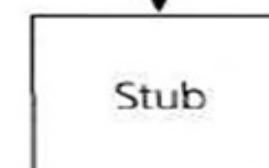
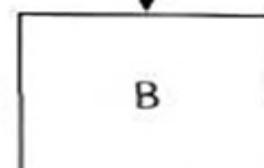
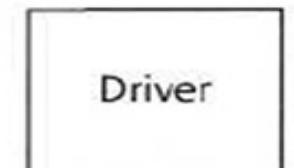
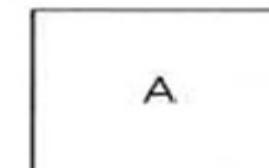


white box

Component:



Component:





Yazılım Test Seviyeleri – Integration Testing

Entegrasyon(Integration) Testi



- ✓ Yazılımın gereksinimlerini doğrulamak üzere test ekibi tarafından yapılan kara kutu testleridir.
- ✓ Entegrasyon testi bir yazılımın **bileşenlerinin birbirine entegre edilmesi** sırasında yapılabileceği gibi **iki farklı yazılımın birbirine entegre edilmesi** sırasında da yapılabilir.
- ✓ Bu yüzden entegrasyon testi, **birim entegrasyon testi** ve **sistem entegrasyon testi** olarak farklı test seviyelerinde yapılabilir.
- ✓ Entegrasyon test stratejileri
 - **Big bang**
 - **Incremental**- Artımlı (bottom-up, top-down, both)

Yazılım

Tümleştirme



EDITABLE STROKE



Yazılım Test Seviyeleri – Integration Testing

Integration Testing Strategy- Big Bang Testi

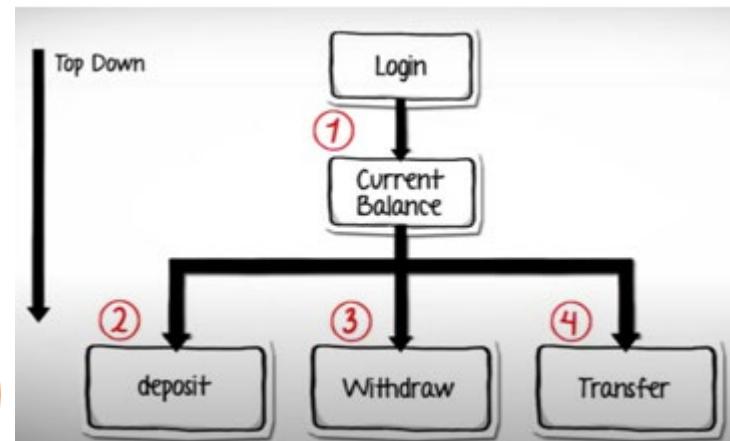
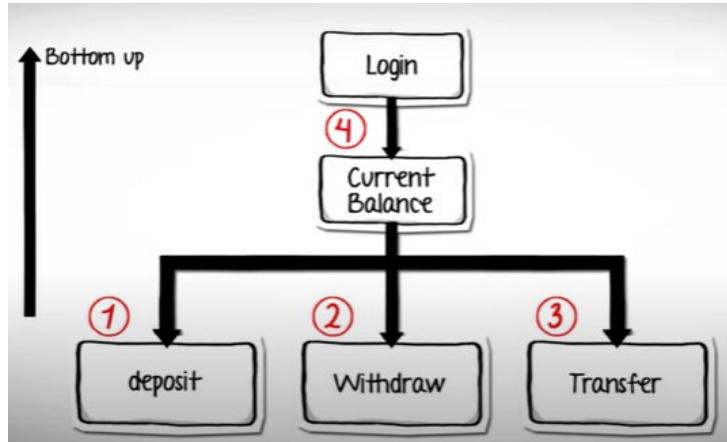
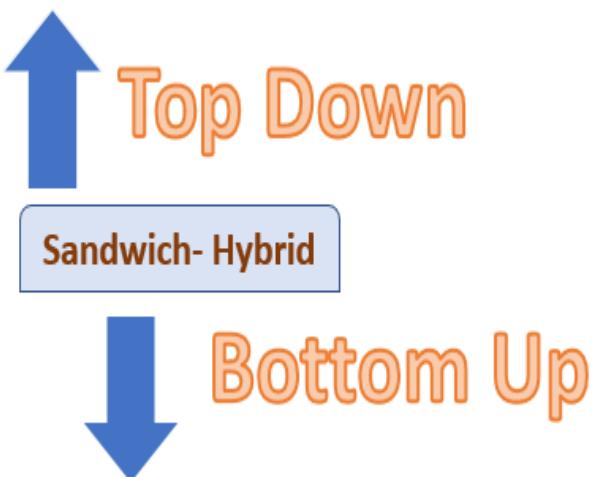
- ✓ Tüm bileşenlerin veya modüllerin aynı anda birbirine entegre edildiği ve ardından bir birim olarak test edildiği bir Entegrasyon testi yaklaşımıdır. Bu birleşik bileşen kümesi, test sırasında bir varlık olarak kabul edilir. Ünitedeki tüm bileşenler tamamlanmadıysa, entegrasyon süreci çalışmayacaktır.
- ✓ Küçük sistemler için uygundur.
- ✗ Hata Lokalizasyonu zordur.
- ✗ Bu yaklaşımda test edilmesi gereken çok sayıda ara yüz göz önüne alındığında, test edilecek bazı ara yüz bağlantıları kolayca gözden kaçabilir.
- ✗ Entegrasyon testi ancak "tüm" modüller tasarlandıktan sonra başlayabileceğinden, test ekibinin test aşamasında yürütme için daha az zamanı olacaktır.



Yazılım Test Seviyeleri – Integration Testing

Integration Testing Strategy- Incremantal (Artımlı) Test

- ✓ Bu yaklaşımada, test mantıksal olarak birbiriyle ilişkili ve sonra uygulamanın düzgün işlemesi için test edilir, iki veya daha fazla modüllerini entegre ederek yapılır.
- ✓ Daha sonra diğer ilgili modüller aşamalı olarak entegre edilir ve mantıksal olarak ilişkili tüm modüller entegre edilip başarıyla test edilene kadar süreç devam eder.
- ✓ Artımlı Yaklaşım ise iki farklı Yöntemle gerçekleştirilir
 - Altüst-Aşağıdan Yukarıya
 - Yukarıdan aşağıya

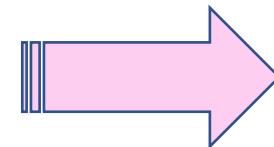




Yazılım Test Seviyeleri – System Testing

Sistem (System) Testi

- ✓ Sistem testi, genellikle **sistemin gerçekleştirebileceği uçtan uca görevleri** ve bu görevleri eksiksiz ve entegre bir sistemde gerçekleştirir. Sistem testi, bir bütün olarak sistemin uçtan uca davranışına odaklanmalıdır(**hem işlevsel hem de işlevsel olmayan**)
- ✓ **Gereksinimlere göre** sistemin uygunluğunun kontrol edilmesini sağlar.
- ✓ **Bileşenlerin genel etkileşimi**ni test eder.
- ✓ **Yük, stres, performans, güvenilirlik ve güvenlik** testlerini içerir..
- ✓ Sistem testi, sistemin şartnameye **uygun olduğunu doğrulamak için yapılan son testtir.**



white box & black box



system testing



Yazılım Test Seviyeleri – Acceptance Testing

Kabul (Acceptance) Testi



black box

- ✓ Müşteri gereksinimlerini doğrulamak üzere **test ekibi tarafından** yapılan kara kutu testleridir.
- ✓ Bu testinin amacı, **sisteme, sistemin parçalarına, kalitesine veya sistemin fonksiyonel olmayan gereksinimlerine** karşı **güven oluşturmak**, sistemin tamamlandığının ve **beklendiği gibi çalışacağı**nın doğrulanmasıdır.
- ✓ Kabul testinde **ana odak hataları bulmak** değildir, **sistemin canlıya hazır olduğunu göstermek**tir.

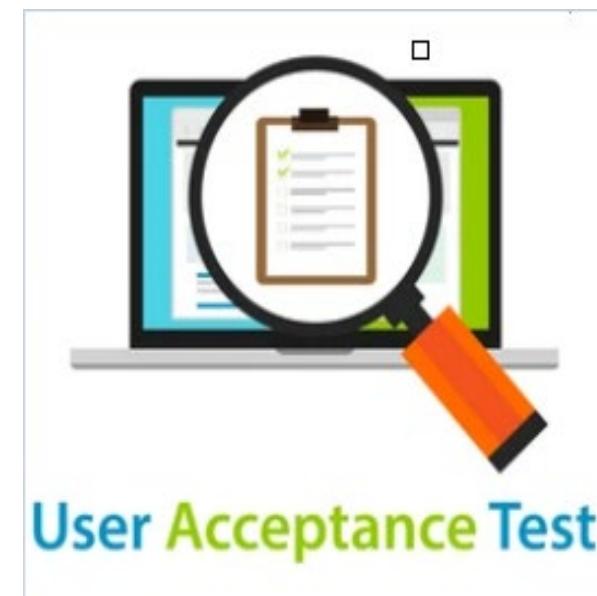




Yazılım Test Seviyeleri – Acceptance Testing

Kabul testleri bir çok aşamada olabilir;

- ✓ Kullanıcı Kabul Testi
- ✓ Operasyonel (Kabul) Test
- ✓ Sözleşmeye Göre Yapılan Kabul Testleri
- ✓ Yasal Mevzuata Göre Yapılan Kabul Testleri
- ✓ **Alfa Testi**
- ✓ **Beta (Saha) Testi**





Yazılım Test Seviyeleri – Acceptance Testing

Alfa Testi (Alpha Testing), ürün neredeyse kullanılabılır durumdayken, geliştirme sürecinin sonuna doğru yapılan bir test türüdür. Alfa Testi, bir tür kabul testidir. Nihai ürünü **son kullanıcılarla sunmadan önce olası tüm sorunları ve hataları belirlemek için** gerçekleştirilir.

Saha Testi (Beta Testing), bir sistemin ya da bileşenin kullanıcının ihtiyaçlarını karşılayıp karşılamadığını, iş süreçlerine uygun olup olmadığını belirlemek amacıyla **kullanıcılar/müşteriler tarafından kendi sahalarında** yapılan kabul testidir, genellikle yazılımın yanı sıra donanım testlerini de içerir.





Soru Çözümleri



Soru Çözümleri-1

Aşağıdakilerden hangileri Tester' in görevidir?

- i. Projede kullandığınız test tool'unu en iyi şekilde kullanabilmek için Test Tool satıcısı ile görüşmek
- ii. Test Data'larini almak ve hazırlamak
- iii. Tüm seviyelere uygun testleri hazırlamak, uygulamak ve sonuçları kaydetmek
- iv. Test spesifikasyonlarını hazırlamak

- A. i, ii, iii doğru ve iv yanlış
- B. ii, iii, iv doğru ve i yanlış
- C. i doğru ve ii, iii, iv yanlış
- D. iii ve iv doğru, i ve ii yanlış

Testçinin Sorumlulukları

- ✓ Test planını takip etmek
- ✓ Hataları tarafsız ve gerçekçi raporlamak
- ✓ Hatayı raporlamadan önce kontrol etmek
- ✓ Yazılımcıyı değil yazılımı test ettiğinin bilincinde olmak
- ✓ Riskleri tarafsız değerlendirmek
- ✓ Hataları önceliklendirmek (prioritise)
- ✓ Gerçekleri paylaşmak

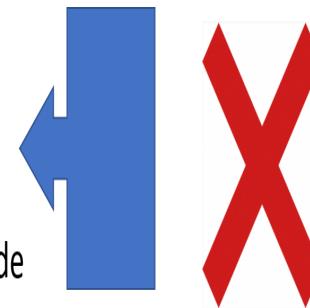


Soru Çözümleri-2

Testin ne kadar yapılması gerekiğinin planladığımız sırada, aşağıdaki etmenlerden;

- i. **Teknik ve Is ürün ve proje riskleri**
 - ii. **Zaman ve maliyet gibi proje kısıtlamaları**
 - iii. **Test takımının büyüklüğü**
 - iv. **Yazılım ekibinin büyüklüğü**
-
- A. i,ii,iii doğru, iv yanlış
 - B. i,iv doğru ii,iii yanlış
 - C. **i,ii doğru ve iii, iv yanlış**
 - D. ii,iii, iv doğru ve i yanlış

- Sürekli, her zaman
- Planlanan süre bittiği zaman
- Kullanıcı yeterli dediğinde
- Yazılımın doğru çalıştığı ispat edildiğinde



- Yazılımın doğru çalıştığından emin olunduğunda
- Yazılımın risk değerlendirmesine bağlı olarak

şartlara göre
doğru kabul edilir

Kesin doğru....



Soru Çözümleri-3

Yazılımın son kullanıcılarına
karşısında görünür olan, belirli veya
beklenen davranıştan sapmasına ne
denir?

- a) Yanlış
- b) işlevsel hata
- c) Arıza
- d) Bug
- e) Kullanım hatası



```
if (new  
    {newPa  
    -UserNa  
    -Passw  
    retur  
    lse {  
        f
```

... Bu yazılım için bir
kusur (fault) demektir....



... Bu kusurlar çalışma
sırاسında arızaya
(failure) neden olur.



Soru Çözümleri-4

Gereksininin(requirements) ve sistemin test edilebilirliğini değerlendirmek ,aşağıdaki aşamalardan hangisinde gerçekleştirilir?

- A. Test analizi ve Tasarımı
- B. Test planaması ve kontrol
- C. Testi hayata geçirme ve yürütmesi
- D. Çıkış kriterlerinin değerlendirilmesi ve raporlama

Verification

-Doğrulama

-Onaylama

Validation

Gözden geçirme

Test



Soru Çözümleri-5

Acceptance testinin odaklandığı temel amaç :

- A. sistemdeki yanlışları bulmak
- B. Sistemin tüm kullanıcılar tarafından kabul edilebilir olmasını sağlamak
- C. Sistemi diğer sistemlerle test etmek
- D. business(is veya ticari)perspektifle test etme

Kabul (Acceptance) Testi

- ✓ Müşteri gereksinimlerini doğrulamak üzere **test ekibi tarafından** yapılan kara kutu testleridir.
- ✓ Bu testinin amacı, **sisteme, sistemin parçalarına, kalitesine veya sistemin fonksiyonel olmayan gereksinimlerine karşı güven oluşturmak**, sistemin tamamlandığının ve **beklendiği gibi çalışacağının doğrulanmasıdır.**
- ✓ Kabul testinde **ana odak hataları bulmak** değildir, **sistemin canlıya hazır olduğunu göstermektedir.**



Soru Çözümleri-6

Onaylama (Validation) aşağıdakilerden hangilerini kapsar ?

- i. Yapılan projenin süreçlere uyulduğunu kontrol etmeye yardım eder
- ii. Projede doğru ürünün yapıldığını kontrol etmeye yardım eder
- iii. Projede yazılımın geliştirmesine yardım eder
- iv. Boşa harcamaları ve artık kullanılmayan parçaları belirlemeye yardım eder.

- A. i,ii,iii,iv doğru
- B. ii doğru ve i, iii,iv yanlış
- C. i,ii,iii doğru ve iv yanlış
- D. iii doğru ve i,ii, iv yanlış

Verification

Doğrulama

Onaylama

Validation

- **Doğrulama:** Yazılımın doğru şekilde üretilmesini sağlamaktır. Geliştirme sürecinde her aşamanın çıktısı, o aşamanın gereklere göre kontrol edilir. Doğrulama ile «Ürün doğru mu geliştirildi?» sorusuna cevap aranır.

- **Onaylama:** Geliştirilen yazılımın kullanım amacıyla uygunun gösterilmesidir. Onaylama ile «Doğu ürün mü geliştirildi?» sorusuna cevap aranır.



Soru Çözümleri-7

Asagidakilerden hangisi Testing için en iyi tarifdir?

- A. Testin hedefi / sebebi, programın çalıştığını göstermektir.
- B. Testin amacı, programın hatasız olduğunu göstermektir.
- C. Testin amacı, programın yapması gerekeni yaptığı göstermektir.
- D. Testing kusurları bulmak amacıyla yazılımı çalıştmaktır



Soru Çözümleri-8

Aşağıdakilerden hangisi Temel test süreci (Fundamental Test Process) in bir parçası değildir ?

- A. Test planlanması ve kontrol
- B. Testi uygulamaya koyma ve çalışma
- C. Gereksinim analizi
- D. Çıkış kriterlerinin belirlenmesi ve raporlama

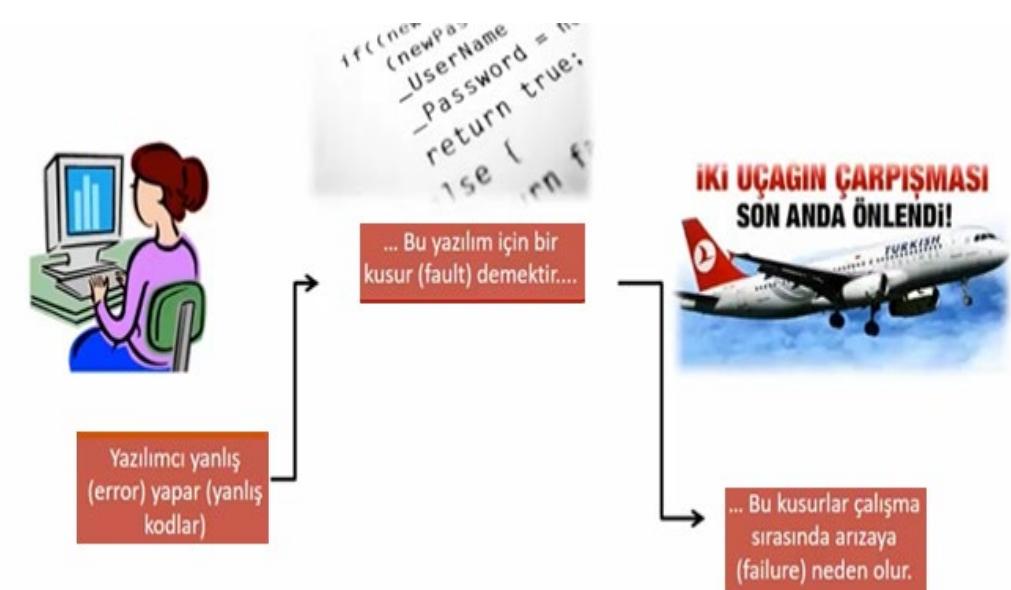




Soru Çözümleri-9

Arıza nedir ?

- A. Yazılımda bulunur; yapılan bir yanlışın sonucudur
- B. Sistemin canlıda belirtilen davranıştan sapılmasıdır
- C. Yazılımda yanlış gerçekleştirilen bir adım veya süreç
- D. Yanlış sonuç veren bir yazılımcı eylemi.





Soru Çözümleri-10

Hangi ifade testing'in rolünü EN İYİ SEKİLDE tanımlar?

- A. Testing, kaliteyi kendi içinde iyileştirir
- B. Testing, kodların doğru versiyonda olmasını sağlar
- C. Testing, kaliteyi değerlendirmek için kullanılabilir.
- D. Testing, yazılımın hatasız olduğunu gösterir.



Soru Çözümleri-11

Bir test yaklaşımının seçiminde aşağıdakilerden hangisi EN önemlidir?

- A. Önerilen teknikleri destekleyecek araçların mevcudiyeti
- B. Önerilen tekniklerde eğitim için izin verilen bütçe
- C. Önerilen tekniklerde mevcut beceri ve deneyim
- D. Test ekibinin yeni teknikleri öğrenmeye istekliliği

- **Test Yaklaşımı**

Projeyi oluşturulan bütün bileşenleri içeresine katarak daha başarılı test sonuçları yakalamak için kullanılan test tiplerinin gruplandırıldığı test anlayışıdır.

Seçilen test yaklaşımı **bağlama(context)** bağlıdır ve riskler, güvenlik, mevcut kaynaklar ve beceriler, teknoloji, sistemin doğası, test hedefleri ve düzenlemeler gibi faktörleri dikkate alabilir.



Soru Çözümleri-12

Yazılım geliştirme sürecinde test süreci hangi noktada başlayabilir?

- A. Kod yazımı tamamlandığında**
- B. Tasarım tamamlandığında**
- C. Yazılım gereksinimleri(software requirements) onaylandığında**
- D. İlk kod modülü birim testi(unit testing) için hazır olduğunda**



3. Testlerin erken başlaması zaman ve para kazandırır

- ✓ Hataları erken bulmak için, yazılım geliştirme yaşam döngüsünde statik ve dinamik test faaliyetleri mümkün olduğunca erken başlatılmalıdır.
- ✓ Testing' e erken başlamaya bazen **sola kaydırma(shift left)** denir.
- ✓ Yazılım geliştirme yaşam döngüsünün başlarında test yapmak, geç fark edildiği için maliyeti artacak değişiklikleri azaltmaya veya ortadan kaldırmaya yardımcı olur.





Soru Çözümleri-13

Aşağıdakilerden hangisi test tasarımlının EN İYİ tanımıdır?

- A. Bir test takımının oluşturulması**
- B. Test senaryolarının çalıştırılması gereken sıranın belirtilmesi**
- C. Bir özelliği test etmek için gereken test senaryolarının belirtilmesi**
- D. Test edilmesi gereken özelliklerin belirlenmesi**

Test tasarımlı aşağıdaki ana faaliyetleri içerir;

- Test senaryoları ve test senaryo setlerini tasarlama ve önceliklendirme
- Test koşullarını ve test senaryolarını desteklemek için gerekli test verilerinin belirlenmesi
- Test ortamının tasarlanması ve gerekli altyapı ve araçların belirlenmesi
- Test temeli, test koşulları ve test senaryoları arasında çift yönlü izlenebilirlik yakalama ”



Soru Çözümleri-14

Beta testi;

- A. Müşteriler tarafından kendi sitelerinde gerçekleştirilir**
- B. Müşteriler tarafından yazılım geliştiricisinin sitesinde gerçekleştirilir**
- C. Bağımsız bir test ekibi tarafından yapılır**
- D. İstek üzere yapılan yazılımı test etmek için kullanışlıdır**
- E. Yaşam döngüsünde mümkün olduğu kadar erken gerçekleştirilir**

Saha Testi (Beta Testing), bir sistemin ya da bileşenin kullanıcının ihtiyaçlarını karşılayıp karşılamadığını, iş süreçlerine uygun olup olmadığını belirlemek amacıyla kullanıcılar/müşteriler tarafından kendi sahalarında yapılan kabul testidir, genellikle yazılımin yanı sıra donanım testlerini de içerir.



Soru Çözümleri-15

Aşağıdakilerden hangisi makul bir test hedefi

DEĞİLDİR;

- A. Yazılımdaki hataları bulmak**
- B. Yazılımın hatasız olduğunu kanıtlamak**
- C. Yazılıma güven duyulmasını sağlamak**
- D. Performans sorunlarını bulmak**

Neden Test?

- Güven inşa etmek için
- Yazılımın doğru çalıştığını ispat etmek için
- Gereksinimlere uygunluğunu göstermek için
- Hata var ise bulmak için
- Masrafları azaltmak için
- Sistemin kullanıcı ihtiyaçlarını karşıladığılığını görmek için
- Yazılımın kalitesini sürdürmek için

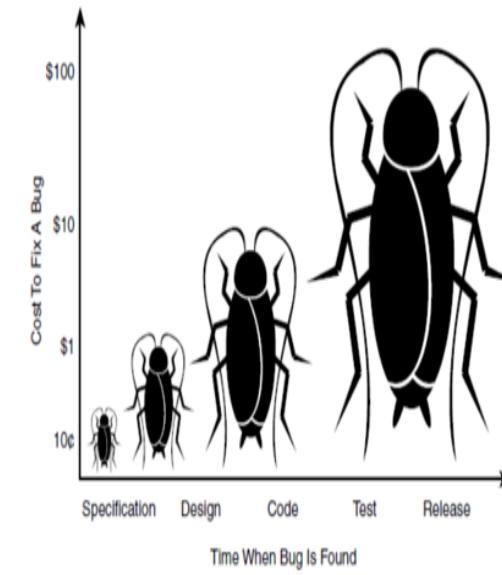


Soru Çözümleri-16

Geliştirme yaşam döngüsünde bir hata ne kadar geç keşfedilirse, düzeltmesi o kadar pahalı olur.

Neden?

- A. belgeler yetersiz olduğundan yazılımın ne yaptığını bulmak daha uzun sürer.
- B. ücretler arttığı için
- C. hata daha fazla belgeye, koda, teste vb. entegre edilmiş olacağından
- D. yukarıdakilerin hiçbirini



Hatalar projenin erken safhalarında bulunursa, o hatayı düzeltmek daha kolay ve hatanın düzeltme maliyeti o kadar az olur

Hatanın geç bulunması...daha fazla belgeye, koda, teste neden olur, bu yüzden maliyet artar



Test Türleri



Test Türleri

Keywords

Code coverage

Maintainability Testing

Security Testing

Load Testing

Re-testing

Regression Testing

Specification-based Testing

Reliability Testing

Functional Testing

Performance Testing

Stress Testing

Test suite

Interoperability Testing

Portability Testing

Structural Testing

Usability Testing



Test Türleri

Test Türleri

Portability

Usability

Maintainability

Black box

Fonksiyonel
Testler

Performance

Reliability

Fonksiyonel Olmayan
Testler

White box

Yapısal Testler
(Structural Testing)

Regresyon ve
Yeniden Onaylama
Testleri



Test Türleri

Fonksiyonel Testler

- Fonksiyonlar test edilir
- "(if...what) ne yapar?" sorusuna cevap verir
- Genelde Black box testlerdir
- Fonksiyonlara ve özelliklere ve bunların belirli sistemlerle birlikte çalışabilirliğine dayanır
- Tüm test seviyelerinde gerçekleştirilebilir
- İki farklı boyutta incelenebilir
 - ✓ Gereksinim temelli testler
 - ✓ İş-süreç temelli testler

Bir sistemin, alt sistemin veya bileşenin gerçeklestirmesi gereken fonksiyonlar, gereksinimler, kullanım senaryoları veya bir fonksiyonel spesifikasyon gibi kriterler için tanımlanabilir. Fonksiyonlar, sistemin "yaptıklarıdır".

Security Testing



Iteroperability Testing



Test Türleri

Fonksiyonel Olmayan Testler

- “(if...how) nasıl yapar?” sorusuna cevap verir
- Genelde White box testlerdir
- Tüm test seviyelerinde gerçekleştirilebilir
- En önemli türleri;
 - ✓ Performance, stress, load
 - ✓ Usability
 - ✓ Maintainability
 - ✓ Reliability
 - ✓ Portability
 - ✓ Security
 - ✓ Compatibility
 - ✓ Volume

Yazılımın fonksiyonel olmayan diğer kalite unsurları test edilir. Fonksiyonel olmayan test terimi, yazılımın performans testindeki yanıt süreleri gibi değişen bir skalada ölçülebilen karakteristiklerini ölçmek için gereken testleri tanımlar.



Test Türleri

Yapısal Testler (Structural Testing)

- Yazılımın yapısı, mimarisi test edilir
- Tüm test seviyelerinde gerçekleştirilebilir
- Yapısal test yaklaşımları ayrıca sistem, sistem entegrasyonu veya kabul testi seviyelerinde uygulanabilir
- White-box ağırlıklı testlerdir
 - ✓ Code coverage
 - ✓ Statement coverage
 - ✓ Decision coverage
 - ✓ Condition coverage

Yapısal tekniklerin test kapsamını tamamlamaya yardımcı olması amacıyla gereksinim bazlı tekniklerden sonra kullanılması önerilir.



Test Türleri

Regresyon ve Yeniden Onaylama Testleri(Regression-Confirmation)

- Değişikliğe bağlı testlerdir
- Onaylama testi (re-test, confirmation test)
 - ✓ Saptanmış hatanın kontrolü
 - ✓ Yerel testtir
- Regresyon testleri
 - ✓ İlgili bölümler test edilir
 - ✓ Genel testlerdir
 - ✓ Sistemde oluşabilecek yan etkiler saptanır
 - ✓ Otomasyona uygundur
 - ✓ Yazılımda her güncelleme sonrası yapılmalıdır

Re-test

Regresyon, yapılan değişiklikler sonucunda oluşan yeni hataları keşfetmek amacıyla zaten test edilmiş olan bir programı yeniden test etme işlemidir.



Bakım Testi



Bakım Testi

Bakım testleri neden gereklidir?

- ✓ Kalitenin devamlılığı
- ✓ Kullanıcı istekleri
- ✓ Kullanıcı ortamındaki değişiklikler
- ✓ Değişen yasalar
- ✓ Performans, güvenlik, uyumluluk, ... eksikleri



Maintanence Testing

Bakım testleri, yazılım devreye alındıktan sonra bakım gerektirdiği durumlarda güncelleme sonrasında yapılan testlerdir..



Bakım Testi

Bakım Testi Nasıl Yapılır?

- Planı yapılmalıdır
 - * Herkes bilmeli
 - * Hazırlık yapılmalı
- Etki analizi yapılmalıdır
 - * Değişikliğin etkisi bilinmeli
 - * Nereler ne kadar test edilecek
- Gereksinimler belirlenmeli
 - Test sonuçları karşılaştırılmalı
 - Hatalı durumlar raporlanmalı
 - Gereksinim listesi güncellenmeli

Without a specification,
you cannot really test,
only explore.

You can validate, but
not verify





BATCH : **BATCH 59**

LESSON : **ISTQB-04**

DATE : **10.06.2022**

SUBJECT : **ISTQB**

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



Statik Testler



Statik Testler

Keywords

Statik Testing (Statik Testler)

Dynamic Testing (Dinamik Testler)

Reviews (Gözden Geçirme)



Statik Testler

- Yazılım geliştirme boyunca kod seviyesine inen ve geliştirme yapan mühendislere **destek vererek hataları önceden kestirmelerine yardımcı olabilen** yazılım destek teknikleridir.
- Statik testlerde **yazılımın doğruluğu kontrol** edilir.
- Bu yüzden **herhangi bir kod çalıştırması olmaz** ve yazılı olan her şey **gözden geçirme sürecine dahil** edilerek çıktılar üzerindeki hatalar önceden kestirilir, yani kodun çalıştırılması yerine yazılan **kod hata bulmak amaçlı okunur**.
- Statik testlerin önemi, **hataların** yazılım yaşam döngüsü içerisinde **erken safhalarda yakalanmasını sağlamasıdır**.
- Statik testler 2 temel tipte oluşur

Gözden Geçirmeler

Statik Kod Analizleri



Statik Testler - Gözden Geçirmeler

- ✓ Proje dökümanlarının **manuel olarak incelenmesine *gözden geçirme (review)*** denir.
- ✓ **Yazılımı yürütmeden kontrol edebildiğimiz** bir yazılım test tekniğidir. Diğer bir tanımla kod çalıştırılarak uygulamayı kontrol eden **dinamik testin tamamlayıcı yanıdır**.
- ✓ Gözden geçirme, yazılımın tasarımındaki **olası hataları bulmak için yapılan işlemlerdir**. Yazılımı test etmenin bir yoludur ve **dinamik testler yapılmadan önce** gerçekleştirilebilir.
- ✓ **Gereksinimler, tasarımlar, kod, test planları, test gereksinimleri, test senaryoları, test komut dosyaları, kullanım kılavuzları veya web sayfaları** gibi tüm yazılım ürünlerinin gözden geçirilmesidir.



Statik Testler - Gözden Geçirmeler

**Bireysel olarak yapılan
gözden geçirme yöntemleri**

yazılımcının kendisinin yaptığı test yöntemleridir. Bu yöntemlerle grup gözden geçirmeleri öncesinde kendi hatalarını azaltarak daha olgun bir kod ortaya çıkartır.

**Grup olarak yapılan gözden
geçirmeler**

konuda uzmanlaşmış kişiler tarafından yapılır. Burada amaç sistematik bir şekilde ön yargılardan uzaklaşarak başka kişilerin yakalayabileceği hata veya hataya sebep verebilecek zayıf yapıları ortadan kaldırmak amaçlanır.



Gözden Geçirmeler

Bireysel olarak yapılan gözden geçirme yöntemleri

- Desk Checking (masada kontrol)
- Proof Reading (uzman deneyimi)

Grup olarak yapılan gözden geçirmeler

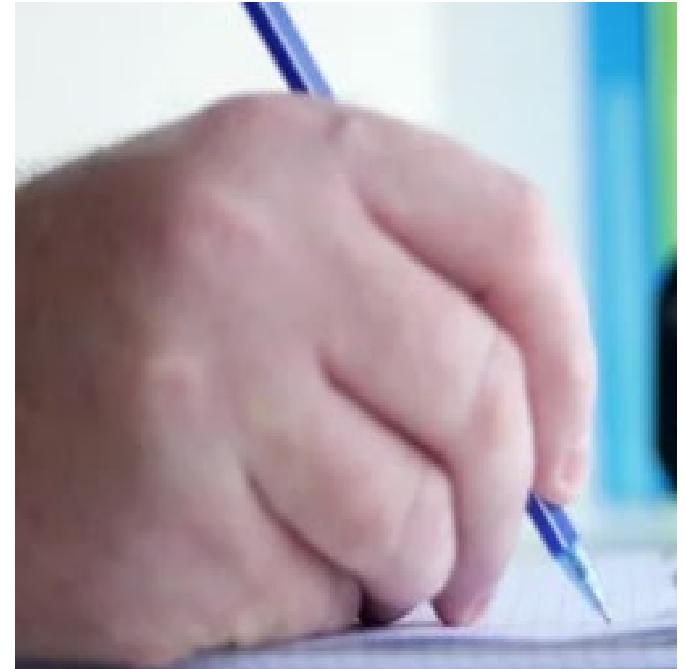
- Reviews (Gözden geçirme)
- Walkthrough (üzerinden geçme)
- Inspection (inceleme-teftiş)



Gözden Geçirmeler - Bireysel

Desk Checking (masada kontrol):

- Yazılan kodun çıktısı alınır ve yazılımcı masa başında bilgisayardan yardım almadan satır satır kod üzerinden geçerek olayı simule etmiş olur.
- Yazılımcı mantık sorguları kontrol eder ve hataları kendi bulur.





Gözden Geçirmeler - Bireysel

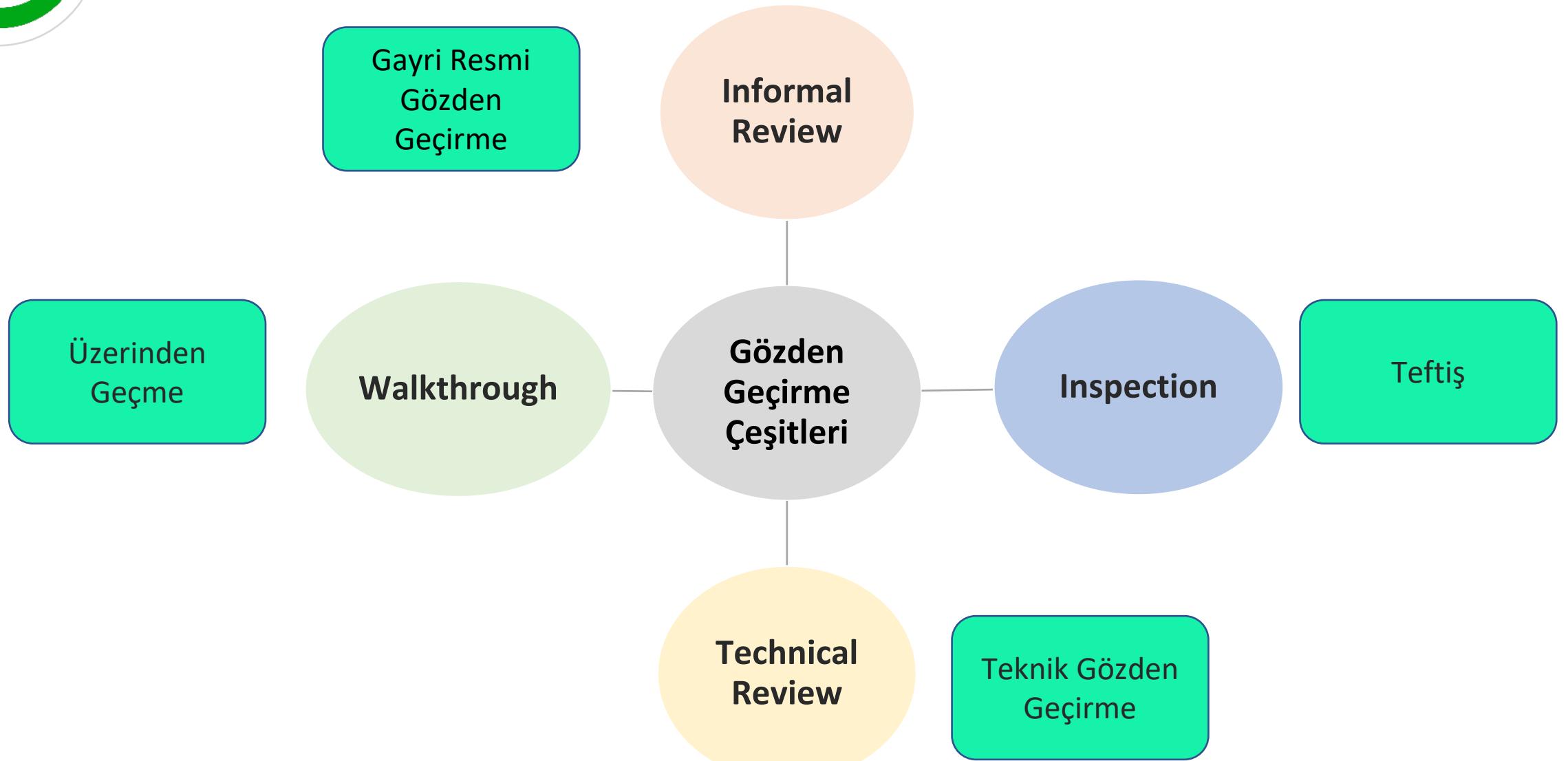
Proof Reading (uzman deneyimi):

Daha önceden yapılmış ve sorunsuz çalışan uzmanlar tarafından hazırlanmış kod, algoritma vb., dökümanları okuyarak kendi çıktılarını test edilir.





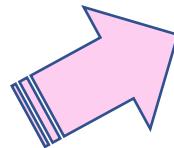
Gözden Geçirmeler – ISTQB dökümanına göre





Gözden Geçirme Çeşitleri

1. Informal Review



- Resmi süreç yoktur.
- Proje gelişim süreci-planlama-proje takvimi izlenir
- Tasarımları ve kodu gözden geçiren tecrübeli teknik biri tarafından gerçekleştirilebilir.
- Sonuçların kayıt altına alınması isteğe bağlıdır.
- Alt seviye teknik detaylara girmez
- Projenin hedeflerine ulaşması için uygun yaklaşımlar değerlendirilir
- **Amaç: Kolay ve hızlı bir şekilde defect bulmak**

Yönetimsel Gözden Geçirme



Gözden Geçirme Çeşitleri

2. Walkthrough

- **Yazar tarafından yönetilir.**
- Scribe kullanımı isteğe bağlıdır. Scribe, gözden geçirilecek iş ürünün sahibi olmamalıdır.
- Senaryoların üzerinden geçme şeklinde yapılır.
- Üzerinden geçme süreci gayri resmiden çok resmiye kadar değişebilir.
- **Amaç: Öğrenme, anlama, defect bulma**



Gözden Geçirme Çeşitleri

3. Technical Review

- Doküman te edilmiş, tanımlanmış defect bulma süreci olan
- Teknik kişileri ve çalışma arkadaşlarını sürece dahil eden
- Yönetimin isteği bağlı olarak katıldığı
- Bulgular listesini, yazılımın gereksinimleri karşılayıp karşılamadığı ile ilgili kararı ve uygun olan durumlarda bulgularla ilgili önerileri içeren gözden geçirme raporunun hazırlanması
- İdeal olarak eğitimli moderatör tarafından yönetilir.
- İsteğe bağlı kontrol listesi kullanılır.
- Amaç: Tartışma, karar verme, alternatifleri değerlendirme, defect bulma, teknik problemleri çözme ve gereksinimlere, planlara, düzenlemelere ve standartlara uyumu kontrol etme



Gözden Geçirme Çeşitleri

4. Inspection

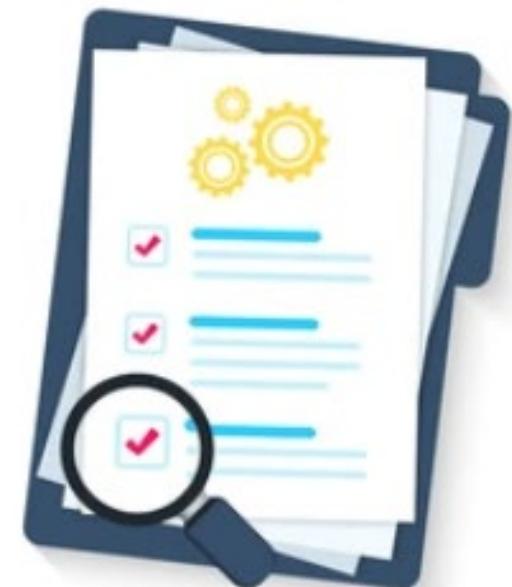
- Kurallara ve kontrol listelerine dayanan resmi süreç
- Yazılım ürününün kabulü için belirli giriş ve çıkış kriteri
- Toplantı öncesi hazırlık
- Bulgular listesini içeren teftiş raporu
- Eğitimli moderatör tarafından yönetilir.
- Scribe kullanımı isteğe bağlıdır.
- **Amaç: Defect bulmak**



Statik Testler - Gözden Geçirmeler

Gözden Geçirmeler Sayesinde

- ✓ Daha az defect ve yazılım maliyeti
- ✓ Yazılım geliştirme faaliyetlerinde verim artar
- ✓ Yazılım geliştirme süresini kısaltır
- ✓ Hata seviyesini düşürür, daha az ciddi hatalar
- ✓ Müşteri ilişkilerini geliştirir
- ✓ Deneyim paylaşımı artar
- ✓ Standartlara uyum artar

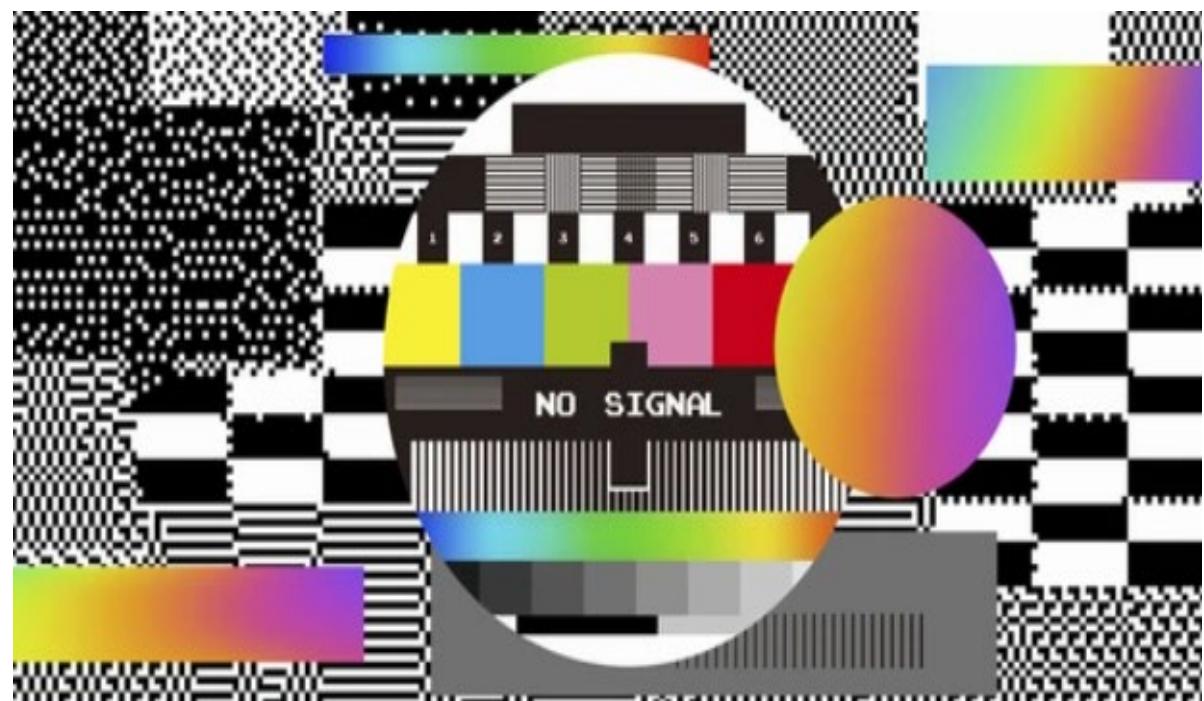




Statik Testler - Gözden Geçirmeler

Gözden Geçirme Sürecindeki Hata Türleri

- ✓ Gereksinim hataları
- ✓ Tasarım hataları
- ✓ Yanlış ara yüz özellikleri





Resmi gözden geçirme işlemleri

Gözden Geçirme Aşamaları

Planning

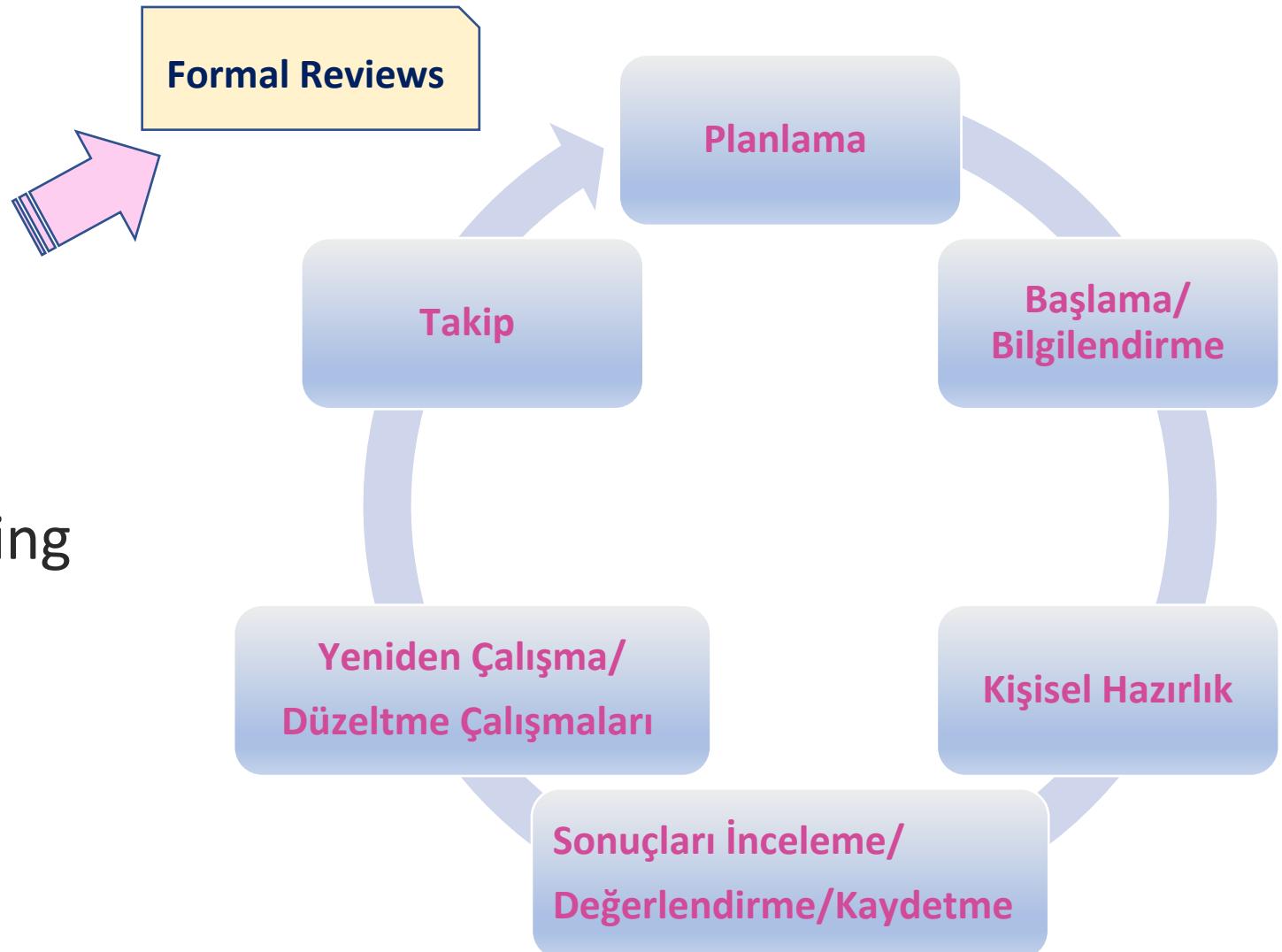
Kick-off Meeting/Overview

Preparation/Individual checking

Review Meeting

Rework

Follow up





Resmi gözden geçirme işlemleri

1. Planlama

- ✓ Gözden geçirme kriterini belirleme
- ✓ Personeli seçme
- ✓ Roller分配ma
- ✓ Daha resmi gözden geçirme çeşidi için giriş ve çıkış kriterini belirleme
(örnek: teftişler)
- ✓ Belgenin hangi bölümlerinin gözden geçirileceğini seçme
- ✓ Giriş kriterini kontrol etme



Resmi gözden geçirme işlemleri

2. Başlama

- ✓ Belgeleri dağıtma
- ✓ Katılımcılara hedefleri, süreci ve belgeleri açıklama

3. Kişisel hazırlık

- ✓ Belgeleri gözden geçirerek gözden geçirme toplantısına hazırlanma
- ✓ Potansiyel hataları, soruları ve yorumları not etme



Resmi gözden geçirme işlemleri

4. Sonuçları inceleme/değerlendirme/kaydetme (gözden geçirme toplantısı)

- ✓ Belgelenen sonuçlar veya tutanaklar üzerinden tartışma veya kayıt tutma
- ✓ Hataları not etme, hataların giderilmesiyle ilgili öneriler sunma, hatalarla ilgili kararlar verme
- ✓ Toplantılar sırasında veya grubun elektronik iletişimlerini izleyerek sorunları inceleme/değerlendirme ve kaydetme



Resmi gözden geçirme işlemleri

5. Yeniden çalışma

- ✓ Bulunan hataları düzeltme (genellikle gözden geçirilen ürünün sahibi/yazar tarafından yapılır)
- ✓ Hataların güncel durumunu kaydetme (resmi gözden geçirmelerde)

6. Takip

- ✓ Hataların ele alınıp alınmadığını kontrol etme
- ✓ Metrikleri toplama
- ✓ Çıkış kriterini kontrol etme



Roller ve Sorumluluklar

- Leader/ Moderator(Lider/Moderatör) şeytini büküş uşanı arasımda
aracılık, gözden geçirme
başarısının bağlı olduğu kişidir
 - Author (Yazar) Kod, dokuman sahibi
 - Reviewers/Inspectors(Gözden geçiriciler) Kod, döküman, vb. inceleyen
uzmanlaşmış (işinin ehli)kişiler
 - Managers(Yöneticiler) Planlamayı yapar, yönetir, iletişim kurar
 - Scriber(Yazıcı-Katip) Toplantı sırasındaki tüm bulguları, gereklilikleri, vb. kayıt eder.



Gözden Geçirme için Başarı Faktörleri

- Her bir gözden geçirmenin önceden **belirlenmiş net hedefleri** vardır
- Gözden geçirme hedeflerine **uygun kişiler dahil edilir**
- Test uzmanları, gözden geçirmeye katkı sağlayan aynı zamanda testleri daha erken hazırlamalarını sağlayacak şekilde **ürün hakkında bilgi edinen gözden geçiricilerdir**
- Bulunan **hatalar normal karşılanır** ve objektif şekilde ifade edilir
- İnsan psikolojisine dikkat** edilir (örnek: süreç yazar için olumlu bir deneyim haline getirilir)
- Gözden geçirme **güven ortamı içinde gerçekleştirilir** ve çıktılar katılımcıları değerlendirmek için kullanılmaz
- Gözden geçirme teknikleri, hedeflere ulaşmanın uygun olacağı şekilde ve yazılımın çeşidine ve seviyesine** ve ayrıca gözden geçiricilere uygun olacak şekilde uygulanır
- Hata tanımlama konusunda** etkinliği artırmak için uygun olan durumlarda **kontrol listeleri ve roller** kullanılır
- Gözden geçirme tekniklerinde, özellikle de **teftiş** gibi daha **resmi tekniklerde eğitim verilir**
- Yönetim** iyi bir **gözden geçirme sürecini destekler** (örnek: proje zamanlamalarında gözden geçirme aktiviteleri için yeterli zamanı vererek)
- Öğrenme ve **süreç iyileştirmesine** vurgu yapılır



Statik Kod Analizi



Statik Kod Analizi

Keywords



Complier (Derleyici)

Control Flow (Akış şeması)

Data Flow (Veri Akışı)

Static Analysis (Statik analiz)



Statik Kod Analizi

- “Statik analizin hedefi, yazılımın kaynak kodundaki ve yazılım modelindeki hataları bulmaktadır.”
- **Statik analiz bir program çalıştırılmadan kaynak kodun otomatik olarak incelenmesidir. Hata ayıklama yöntemi olarak da tanımlanır.**
- Statik analiz, bir statik test çeşididir. Statik test, **gözden geçirme(review)** ve **statik analiz** olmak üzere ikiye ayrılır.



Amaç : Kodunuzun belli bir kaliteyi tutturmasını sağlamak için yapılır.



Statik Kod Analizi

Örnek : kodun okunabilir olması.. Önemli bir özelliktir..



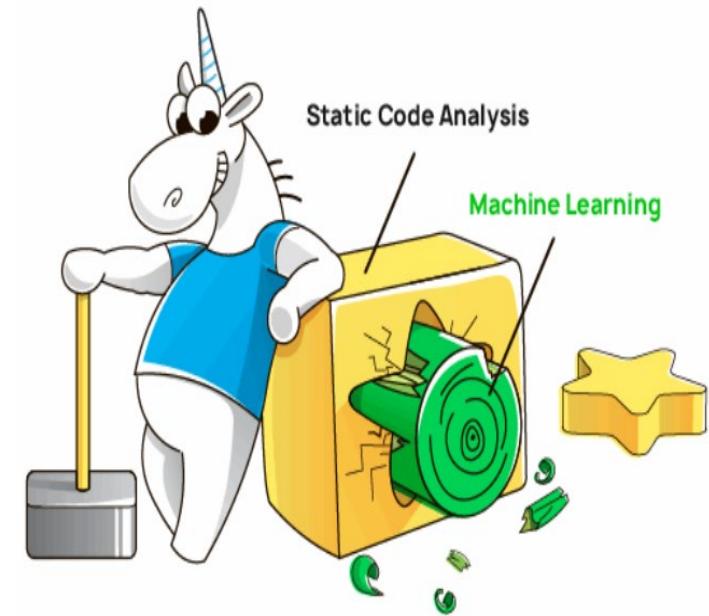
Static code



Statik Kod Analizi

Statik Analizin Önemi-Avantajları

- ✓ Çalıştırmadan önce hataların bulunması
- ✓ Karmaşık kodların bulunması
- ✓ Dinamik testte bulunması zor hatalar
- ✓ Tutarlılıkların bulunması
- ✓ İyileştirilmiş kod ve tasarım sürdürülebilirliği
- ✓ Tasarımın bakım açısından geliştirilmesi
- ✓ Hatalardan arınma(az maliyet ile)





Statik Kod Analizi

Statik Analiz Türleri

Kozmetik: Kod takım/dil tarzı standartlarına uygun mu?

Kodun Tasarım Özellikleri: Kod asgari derecede karmaşık ve korunabilir mi?

Sezgisel / Hata Kontrolü: Kodda aradığımız ihlaller var mı?

Öngörülü: Bu kod çalışma zamanında nasıl davranışacak?

Resmi Kanıt: Bu kod doğru mu?





Statik Kod Analizi

Statik Kod Analizinin Sağladığı Kazanımlar

- ✓ Proje içerisindeki hatalar ne kadar geç bulunursa, düzeltme maliyeti o kadar yüksek olacaktır.
- ✓ İtibar, para kaybı gibi yan etkileri en aza indirger.
- ✓ Proje bakımını oldukça kolaylaştırır.





Statik Kod Analizi

**Statik Kod
Analizinde
İncelenen
Konular**

- Hata ve Güvenlik Açıkları
- Kaynak Kod Metrikleri
- Mimari Analiz
- Kodlama Standartları
- Tersine Mühendislik



Statik Kod Analizi

Statik Kod Analizinin Çalışma Aşamaları

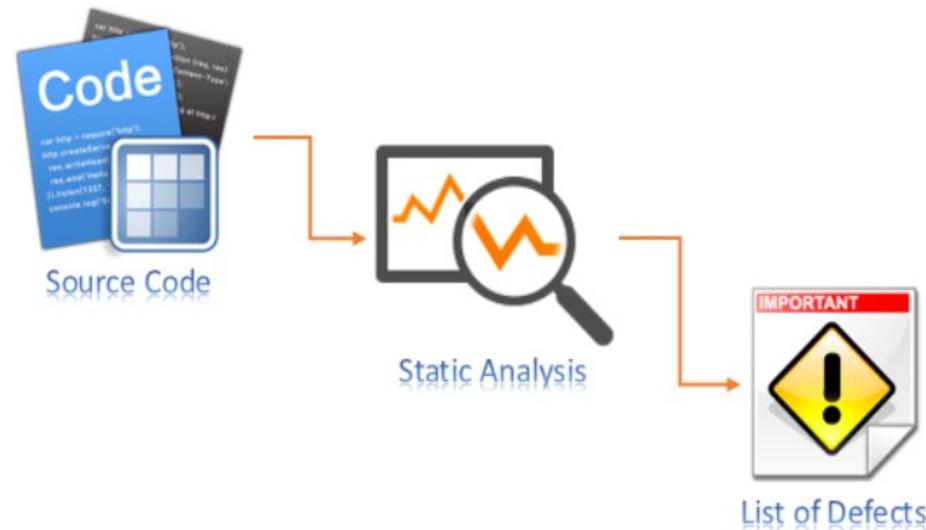




Statik Kod Analizi

Statik Analizle(Statik analiz araçları tarafından) Saptanabilecek Hatalar

- ✓ Değer atanmamış değişkenin yanlışlıkla referans gösterildiği durumlar
- ✓ Modul ve birimler arasında tutarsız arayüzler
- ✓ Kullanılmayan değişkenler
- ✓ Tipi uygun olmayan değişkenler
- ✓ Erişilmeyen ölü kod parçaları
- ✓ Sonsuz döngü gibi hatalı yapılar
- ✓ Standartların göz ardı edilmesi
- ✓ Güvenlik açıkları
- ✓ Syntax hataları





Test Tasarım Teknikleri



Test Tasarım Teknikleri Test Geliştirme Süreci

Keywords





Test Tasarım Teknikleri - Test Geliştirme Süreci

Test Geliştirme Süreci

- Testçilerin test yaparken kullandıkları döküman ve testleri sonucunda ortaya koydukları dökümantasyon formal ve formal olmayan firmalar arasında farklılıklar gösterebilir.
- Test yapabilmek için bazı bilgilerin hazır olması gereklidir. Gerekli tüm gereksinimlerin araştırılması işine test analizi denir. Gerekli bilgiler kısaca şunlardan oluşur;
 - **Neyin test edileceği**
 - **Nasıl test edileceği**
 - **Test etmek için gerekli veriler**
 - **Testin başarılı olması için gerekli koşullar**
 - **Testin sonuçlarını değerlendirmek için gerekli bilgiler**



Test Tasarım Teknikleri - Test Geliştirme Süreci

➤ Test Tasarımı

- ✓ Hatalı durumun belirlenmesi
- ✓ Koşullar
- ✓ Test oracle
- ✓ istenilen (expected result)- gerçeklesen sonuç (actual result)-> bu bağlamda **senaryolar** belirlenmelidir

Hatalı durumdan bağımsız olarak doğru durumun bulunması için gerekli bilgiler "testoracle" denir.

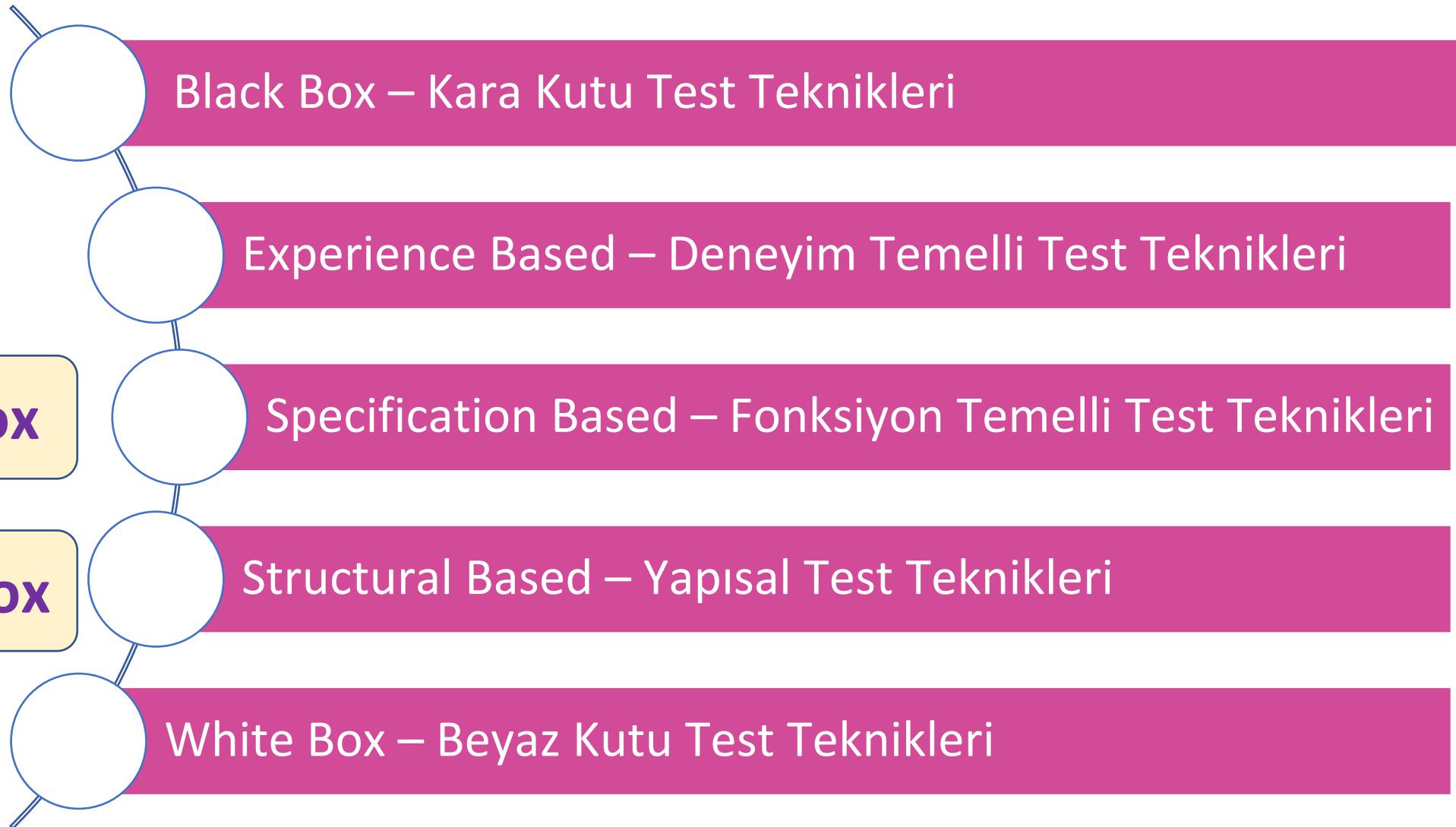
➤ Testin Tatbiki (implementation)

- ✓ Gruplama
- ✓ Sıralama

Test set



Test Tasarım Teknikleri





Test Tasarım Teknikleri

Test tasarım teknikleri

Black Box –
Kara Kutu Test Teknikleri

White Box –
Beyaz Kutu Test Teknikleri

Experience Based –
Deneyim Temelli Test Teknikleri

gereksinim bazlı
test tasarım
teknikleri

yapı bazlı
test tasarım
teknikleri

deneyim bazlı
test tasarım
teknikleri

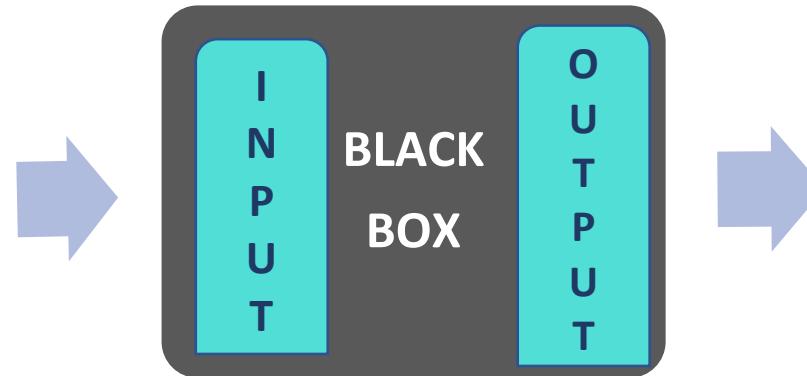


Kara Kutu Test Tasarım Teknikleri



Test Tasarım Teknikleri- Black Box

Yazılımın iç yapısı (kod, database, aktarım, vb.) görülmemiği için kullanıcı gibi test edilir, bu yüzden kara-kutu testler denir.



Girişlere karşılık gelen çıkışlar kontrol edilir, hatalı çıkışlar saptanır, test edilir

ISTQB bu konuyu iki kavram ile ifade etmiştir:

- 1.Kara kutu testi:** Bileşenin veya sistemin iç yapısı bilinmeden gerçekleştirilen işlevsel veya işlevsel olmayan testler.
- 2.Kara kutu test tasarımlı tekniği:** Bileşenin veya sistemin iç yapısı bilinmeden, bileşenin veya sistemin işlevsel veya işlevsel olmayan gereksinimi analiz edilerek test durumları hazırlamak.

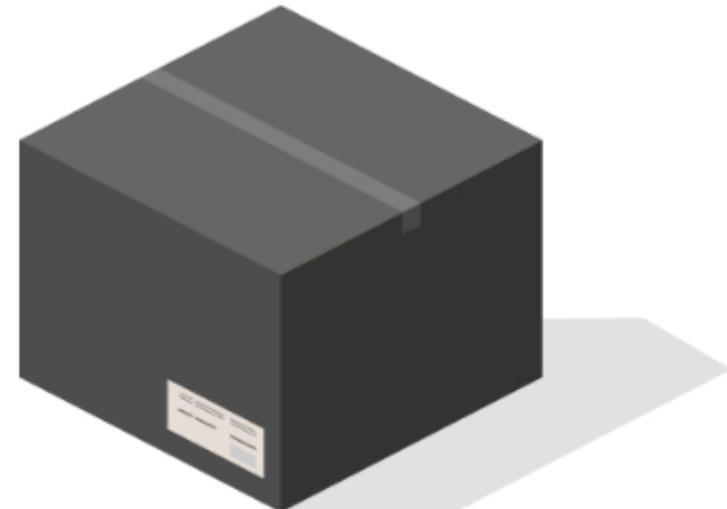


Test Tasarım Teknikleri- Black Box

QA testers

Black Box neyi amaçlar?

En temel anlamda kullanmamızın amacı, yazılımın **kullanıcının isteklerini** tatmin edip etmediğini anlamaktır.



Black box - we do not know anything



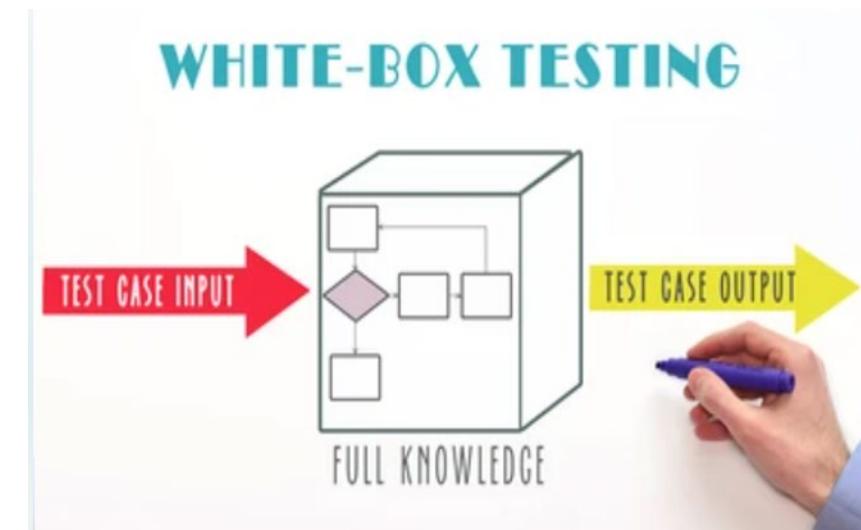
Test Tasarım Teknikleri - White Box

ISTQB bu teknigi aşağıdaki şekilde tanımlamıştır:

1. Beyaz kutu testleri: Bileşen veya sistemin iç yapısının analizine dayanan testlerdir.

2. Beyaz kutu test tasarım tekniği: Bir bileşen veya sistemin iç yapısının analizi ile test durumlarının hazırlanmasıdır.

Yazılımın iç dinamikleri (algoritmalar, mantık yapıları, database sorguları, mimari yapısı, vb.) göz önüne alınarak analiz çıktıları test edilir.





Test Tasarım Teknikleri- White Box

Developers

White Box neyi amaçlar?

En temel anlamda kullanmamızın amacı, **yazılımın kodlarının doğruluğunu ve uyumluluğunu** anlamaktır.



White box - we know
everything



Test Tasarım Teknikleri - Experience Based

Test yapanların deneyimlerinden yola çıkarak test edilir.

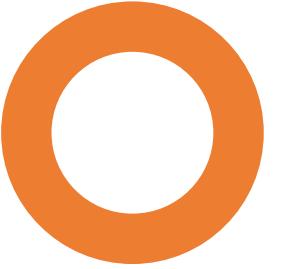
Farklı alanlar için farklı test metodları kullanılır.

Hataların kümeleşmesi (defect clustering) bu test teknliğinde önemli yer alır.

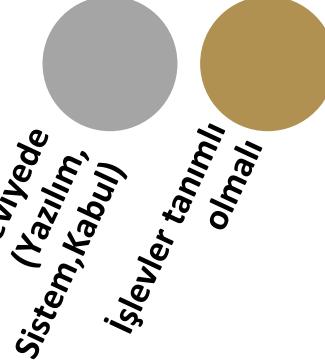


Test Tasarım Teknikleri - Kullanımı

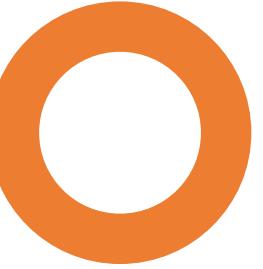
Kara Kutu Testleri



Her seviyede
(Yazılım,
İşlevler tanımlı
olmalı)



Beyaz Kutu Testleri



Birim

Birim

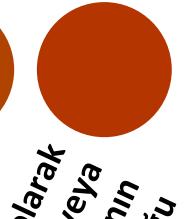
Tümleştirme

Kod kapsama

Deneysel Temelli Testler



Tanımlı
fonksiyon ve
gerekçinimlerin
olmadığı
durumlarda
Genel olarak
düşük risk veya
zaman baskısının
fazla olduğu
durumlarda





Black Box Test Teknikleri

Gereksinime dayalı test teknikleri

Equivalence Partitioning
(Eşdeğer Aralık Testi)

Boundary Value Analysis
(Sınır Değer Analizi)

Decision Table Testing
(Karar Tablosu Testi)

State Transition Testing
(Durum Geçiş Testi)

Use Case Testing (Fayda Analizi Testi)



Black Box Test Teknikleri- Equivalence Partitioning

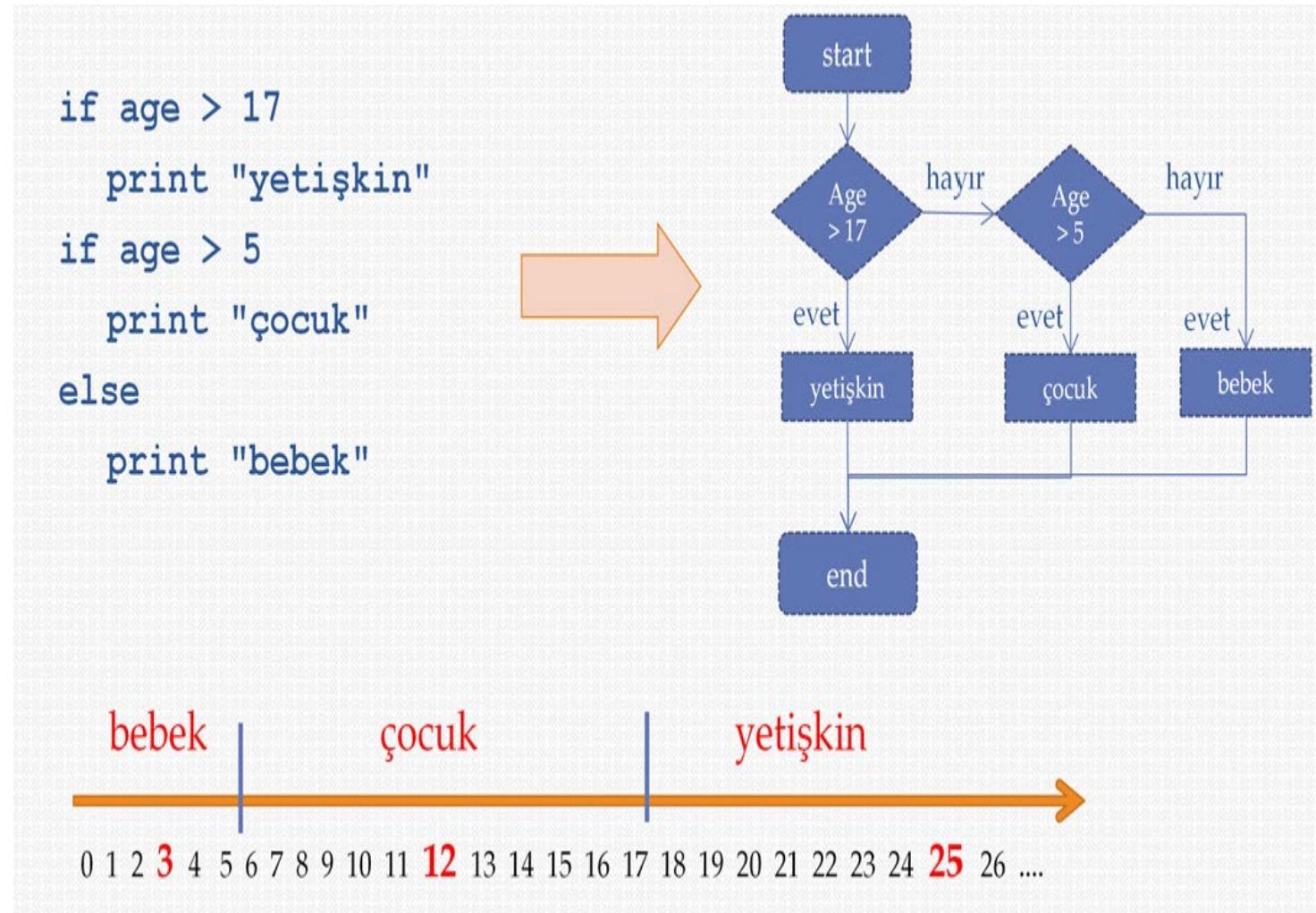
- **Eşdeğer aralık yöntemi**, aynı bölgedeki değerler girdi olarak kullanıldığında aynı sonucu verir ön koşulundan çalışmaktadır.

- ✓ Yazılımların tamamı için **test girdi seti çok geniş**tir.
- ✓ Bu nedenle **girdiler sonlu sayıda alt kümelere bölünür**.
- ✓ Her bir **alt alan bir eşdeğerlik sınıfı olarak bilinen ve en az bir test girişini kaynağı** olarak hizmet eder.
- ✓ Seçilen girdiler ile **uygun ve yeterli test kapsamı** sağlanmalıdır.
- ✓ Denklik paylarına ayırma tüm test seviyelerinde **uygulanabilir**.



Black Box Test Teknikleri- Equivalence Partitioning

- ✓ Yazılım için **aynı sonucu verecek girdi değerleri** belirlenir.
- ✓ Genelde iki sınıf tanımlanabilir:
 - **Geçerli Giriş Sınıfı :**
Yazılımın kabul edeceği geçerli giriş aralığıdır
 - **Geçersiz Sınıf :**
Formal olmayan veya geçerli aralık dışı veri kümeleridir.
- ✓ Tanımlanan her küme için 1 test durumu yazılır.





Black Box Test Teknikleri- Equivalence Partitioning

NASIL ÇALIŞIR?

- Eğer giriş koşulu değerleri belirli bir dizi belirtiyorsa, bu durumda 1 geçerli, 1 geçersiz sınıf vardır.
Kredi sadece **1+1, 3+1 ve 4+1** için verilir.
Geçerli : {1+1, 3+1 ve 4+1 }
Geçersiz : {herhangi bir değer}

- Eğer giriş koşulu değeri kesin bir koşulsa, bu durumda da 1 geçerli, 1 geçersiz sınıf vardır.
Kredi sadece **gerçek kişilere** verilir.
Geçerli : {Gerçek kişi}
Geçersiz : {tüzel, herhangi bir değer}



Black Box Test Teknikleri- Equivalence Partitioning

Bir başarı değerlendirme sisteminde,

0-10 puan aralığı "**başarısız**", (10 dahil)

11-20 puan aralığı "**yeterli**", (20 dahil)

21 puan ve üstü ise "**başarılı**" olarak belirleniyor. (21 dahil)

Böyle bir durumda "Denklik Paylarını" sayı doğrusu üzerinde gösterelim.

Bu yöntemle test yapıldığında kullanılabilcek giriş değerleri (input) sırasıyla TC1 (2, 13, 22) olsun.





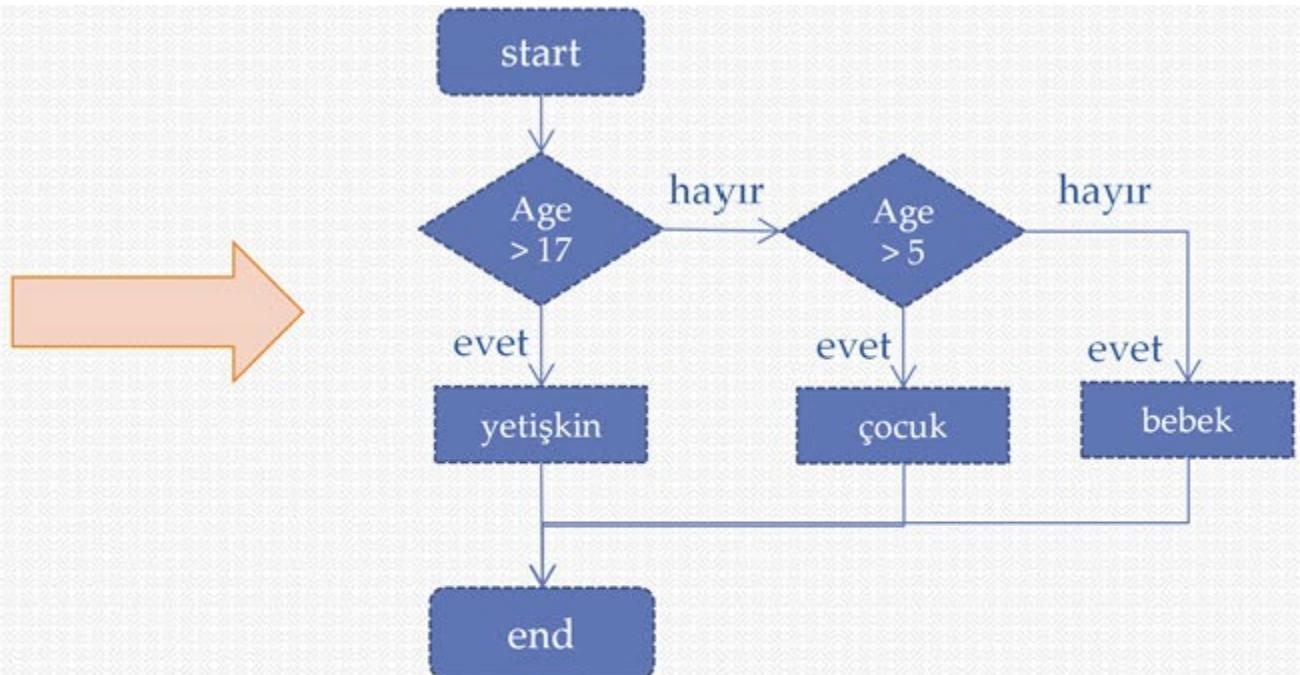
Black Box Test Teknikleri - Boundary Value Analysis

- ✓ Sınır değer testleri değişim noktalarında yazılımın kararlılığını test etmek üzere yapılır.
- ✓ Sınır değerler kontrol edilir(**maksimum ve minimum değerleri**)
- ✓ Bu yüzden eşdeğer aralıkların çıkartılması ve en büyük ve en küçük değişim noktalarının saptanması gereklidir.
- ✓ **2-değerli ve 3-değerli sınır değer testi yaklaşımı** vardır.
 - 2-değerli yaklaşımında değişim noktası ve bir sonraki değer alınırken
 - 3-değerli yaklaşımında değişim noktası, bir önceki ve bir sonraki değerleri de içerisinde alan üç farklı değer test edilir.



Black Box Test Teknikleri - Boundary Value Analysis

```
if age > 17  
    print "yetişkin"  
if age > 5  
    print "çocuk"  
else  
    print "bebek"
```





Black Box Test Teknikleri - Boundary Value Analysis

Bir başarı değerlendirme sisteminde,

0-10 puan aralığı "**başarısız**", (10 dahil)

11-20 puan aralığı "**yeterli**", (20 dahil)

21 puan ve üstü ise "**başarılı**" olarak belirleniyor. (21 dahil)

Sınır değer testleri için (10 ve 20) sınır değerlerini doğrusu üzerinde gösterelim.

Bu yöntemle test yapıldığında kullanılabilecek giriş değerleri (input) sırasıyla

TC1 (9, 10, 11: 19, 20, 21) olsun.





Black Box Test Teknikleri- Decision Table Testing

- Eğer kontrol edilecek çok sayıda durum var ise her bir durumu tablo içerisinde alınır ve **her bir duruma karşı yazılımın vereceği cevaplar** yine tabloya yerleştirilerek karar tablosu oluşturulur.

- ✓ Karar tabloları oluştururken gereksinimler analiz edilir ve yazılımın koşulları ve eylemleri belirlenir. **Girdi koşulları ve eylemler sıkılıkla doğru veya yanlış (Boolean) olacakları şekilde ifade edilir.**
- ✓ Karar tablosu testi, tetikleme koşullarını, genellikle tüm girdi koşulları için doğru ve yanlış kombinasyonlarını ve her bir koşul kombinasyonu için ortaya çıkan eylemleri, sonuçları içerir.
- ✓ Tablonun **her bir sütunu, farklı bir koşul kombinasyonunu tanımlayan bir iş kuralına karşılık gelir** ve bu kuralla bağlantılı eylemlerin yürütülmesi ile sonuçlanır.
- ✓ Karar tablosu testinde yaygın şekilde kullanılan test kapsamında amaç **tabloda yer alan sütunlardan her biri için en az bir testin yürütülmesi** şeklidindedir.
- ✓ Karar tablosu testinin **güçlü yanı, test sırasında gözden kaçabilecek koşul kombinasyonlarının net bir şekilde listelenmesidir.**
- ✓ Yazılım davranışının birden fazla mantıksal karara bağlı olduğu **tüm durumlarda uygulanabilir.**



Black Box Test Teknikleri- Decision Table Testing

input

output



Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Bekar	True	False	False	False	False
Evli	False	True	True	True	True
Eşi Çalışan	False	False	False	True	True
Çocuk Sahibi (1 tane)	False	False	True	False	True
Actions					
AGİ (TL)	268	321	362	268	308

3 şartın olası kombinasyonu için

Test edilmesi gereklili durum sayısı: $2^3 = 8$



Black Box Test Teknikleri- Decision Table Testing

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Minimum 8 Karakter	True	True	True	True	True	True
Rakam içermesi	False	True	True	True	False	False
Özel karakter içermesi *	False	False	True	True	True	True
Harf içermesi	False	False	False	True	True	False
Actions						
Geçerli Şifre	HAYIR	HAYIR	EVET	EVET	EVET	EVET



1

2



Black Box Test Teknikleri- State Transition Testing

- ✓ Yazılımın davranışı mevcut veya geçmişteki durumuna göre değişiklik gösterebilir. Bu tür davranışlar sergileyen yazılımlar bir **durum geçiş diyagramı** ile gösterilebilir.
- ✓ Durum geçiş diyagramları test uzmanının **yazılımın alabileceği durumları, durumlar arasındaki geçişleri, durum değişikliklerini (geçişleri) tetikleyen girdileri veya olayları ve bu geçişler sonucunda oluşabilecek eylemleri görüntülemesini** sağlar.
- ✓ Test edilen **sistemin veya nesnenin durumları ayrı ayrıdır, tanımlanabilir ve ölçülebilir sayıdadır.**
- ✓ Durum tablosu, **durumlar ve girdiler arasındaki ilişkiyi gösterir ve geçersiz olan olası geçişleri ortaya çıkarabilir.**
- ✓ Durum geçisi testi genel olarak **en fazla gömülü yazılımlarda ve teknik otomasyonda** kullanılır. Ayrıca bu teknikle nesnelerin modellenmesi veya ekran-dialog akışının test edilmesi (örnek: Internet uygulamaları veya iş senaryoları için) mümkündür.



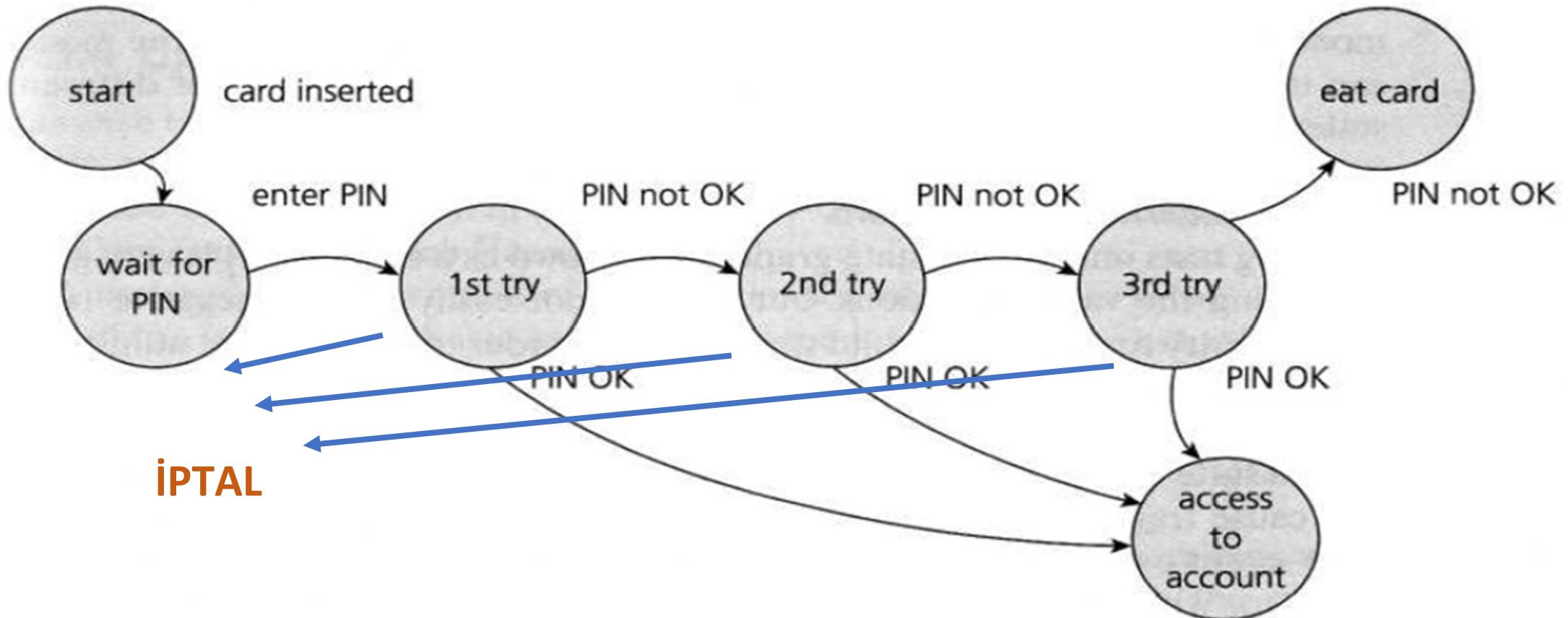
Black Box Test Teknikleri- State Transition Testing

Örnek : Kullanıcının kartını atm ye yerleştirmesiyle başlayan süreç sınırlı durumlar arasında geçiş yaparak kullanıcının isteklerini yerine getirmesini sağlamaktadır.

Bu senaryo da durum geçiş testleri neler olabilir ?



Black Box Test Teknikleri- State Transition Testing



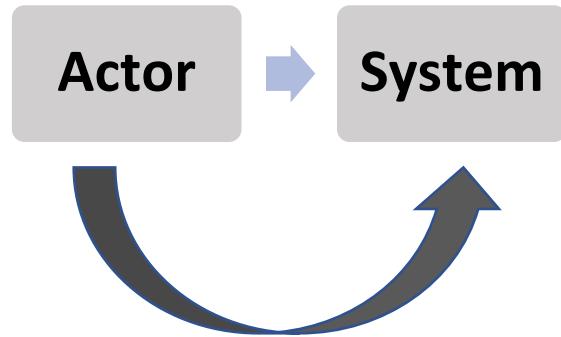


Black Box Test Teknikleri- Use Case Testing

- ✓ **Fayda Analizi Test-Kullanım Senaryosu Testi** olarak da bilinir..
- ✓ Kullanım senaryosu, **aktörler ve sistem arasındaki etkileşimleri tanımlayarak**; bu etkileşimler sonucunda **üretilen değeri gösterir**.
- ✓ Her bir kullanım senaryosunda, kullanım senaryosunun **başarılı bir şekilde çalışması için karşılanması gereken önkoşullar** bulunur.
- ✓ Kullanım senaryoları, **gerçek kullanımlara dayanarak** sistem boyunca "süreç" akışını tanımlar, bu nedenle kullanım senaryolarından türetilen test senaryoları, sistemin **gerçek dünyada kullanımı sırasında süreç akışlarında hataları ortaya çıkarmanın en kolay yoludur**.



Black Box Test Teknikleri- Use Case Testing



HAPPY PATH

	Step	Description
Main Success Scenario A: Actor S: System	1	A: Inserts card
	2	S: Validates card and asks for PIN
	3	A: Enters PIN
	4	S: Validates PIN
	5	S: Allows access to account
Extensions	2a	Card not valid S: Display message and reject card
	4a	PIN not valid S: Display message and ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat card and exit



Black Box Test Teknikleri- Use Case Testing

Use case, kullanıcı ile sistem arasındaki etkileşimi göstermek üzere kullanılan senaryolar topluluğudur.

Test uzmanları, Aktör ve sistem arasındaki etkileşimi adım adım tarifler ve aktörün bakış açısıyla hazırlar.

Kullanım senaryoları, soyut seviyede (Teknolojiden bağımsız, iş süreç seviyesi) ve Sistem seviyesinde (sistem kullanım senaryosu) tanımlanabilir.

Sistemin başından sonuna, her bir işlemin tek tek, ancak bütün sistemi kapsayacak şekilde test edilmesi amaçlanır.

Durumların genel sıralamasını ve her durumu kapsayacak, her geçisi deneyecek, belirlenmiş geçiş sıralamalarını deneyecek veya geçersiz geçişleri test edecek testler tasarlabilir.

Her bir kullanım senaryosunda, kullanım senaryosunun başarılı bir şekilde çalışması için karşılanması gereken " önkoşullar" ve "son koşullar" bulunur.

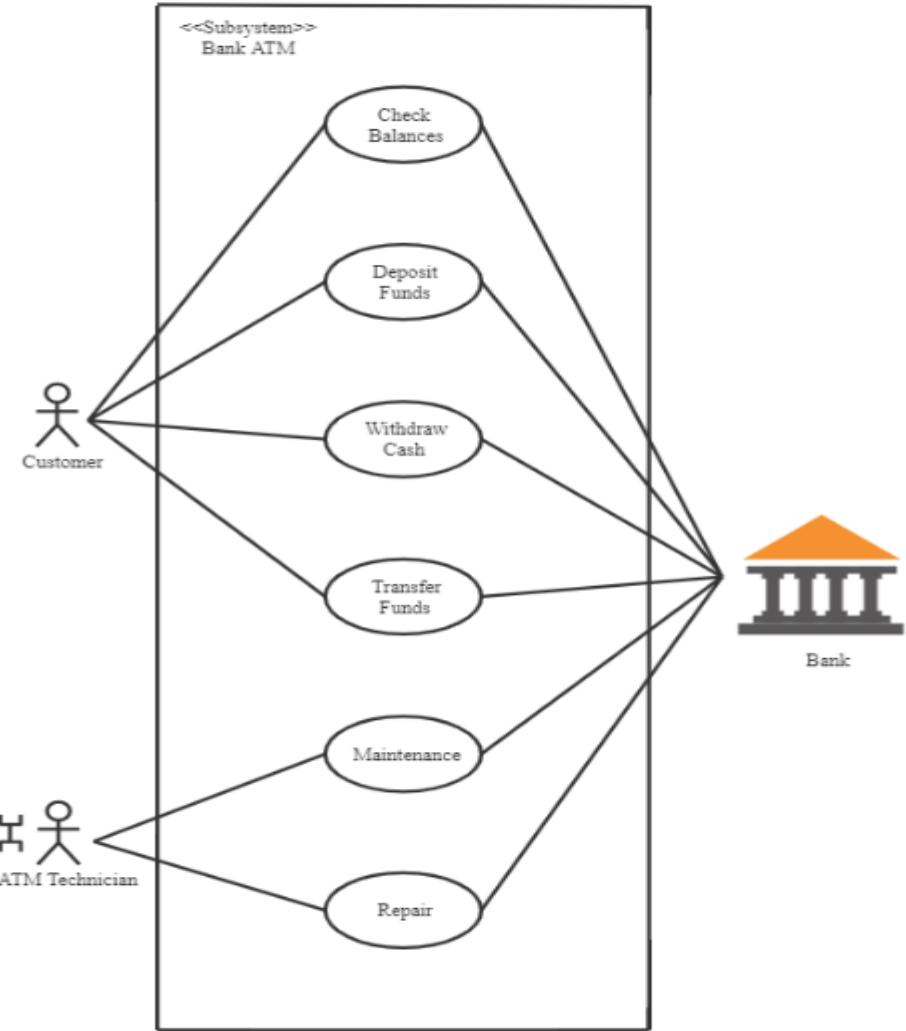


Black Box Test Teknikleri- Use Case Testing

- ✓ Kullanım senaryosunda genellikle **bir ana (en olası) senaryo ve alternatif senaryolar** bulunur.
- ✓ Kullanım senaryoları **«sistem» ve «kabul»** testlerinin tasarılanmasında test senaryoları için temel oluşturur
- ✓ <https://www.edrawsoft.com/article/use-case-diagram-examples.html>
bu web adresinde güzel örnekler bakabilirsiniz..



Use Case Testing – ATM için Kullanım Senaryosu Örneği



Use Case 1.1 *Withdraw money from an ATM*

Withdraw money from an ATM

Primary actors: Customer

ATM Technician

Bank

Preconditions: Network connection is active

ATM has available cash

Basic flow of events:

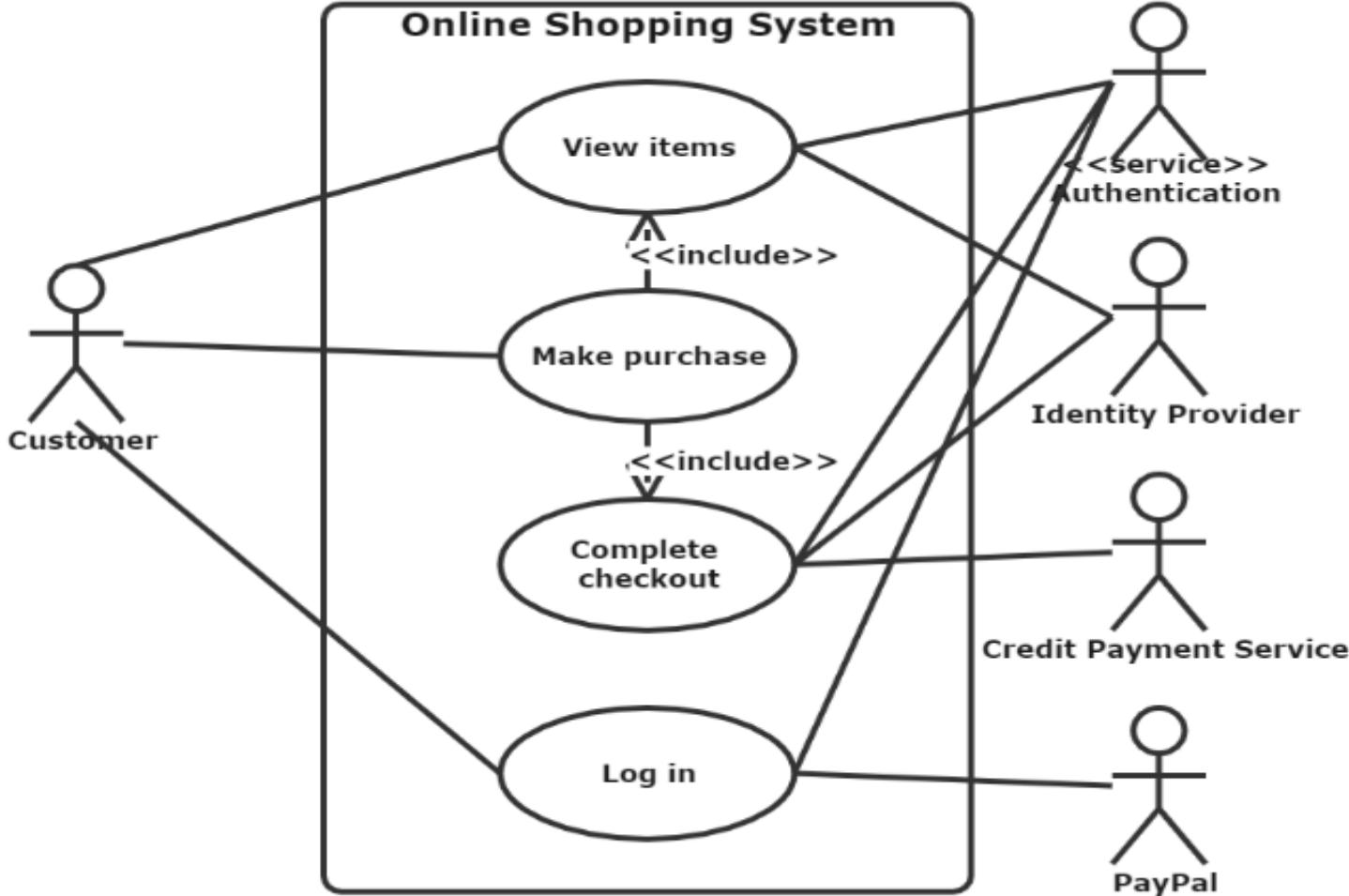
1. Bank customer inserts debit card and enters PIN.
2. Customer is validated.
3. ATM displays actions available on ATM unit. Customer selects Withdraw Cash.
4. ATM prompts account.
5. Customer selects account.
6. ATM prompts amount.
7. Customer enters desired amount.
8. Information sent to Bank, inquiring if sufficient funds/allowable withdrawal limit.
9. Money is dispensed and receipt prints.

Alternative flows

- 2a. Customer is not validated.
 - 2a1. ATM displays error message.
- 7a. Customer selects invalid amount.
 - 7a1. ATM prompts user to re-enter valid amount.
- 8a. Customer has insufficient funds.
 - 8a1. ATM displays error message.
 - 8a2. ATM shows available withdrawal limit, redirects to step 6.
- 9a. ATM has insufficient cash.
 - 9a1. ATM Technician is alerted.
 - 9a2. ATM displays error message and phone number to call.
- 9b. Cash gets stuck in dispensing.
 - 9b1. ATM displays error message.



Use Case Testing - Scenario Example for Online Shopping





Beyaz Kutu Test Tasarım Teknikleri



Beyaz Kutu Test Tasarım Teknikleri

Yapı bazlı test tasarım teknikleri

Bilinen özellikleri şöyledir:

- Geliştirilen yazılım ürününe ait algoritmalar, yazılımın iç yapısı, mimarisi, veri tabanı sorguları, mantık yapıları dikkate alınarak yapılan testlerdir.
- Yazılımcılar tarafından bu çıktılar dikkate alınarak gerçekleştirilir.
- **Yazılımın iç çalışma mantığı** ile ilgili bilgiler test senaryoları türetmek için kullanılır (kod ve detaylı tasarım bilgileri)
- Mevcut test senaryolarıyla yakalanan kapsam derecesi ölçülerek kapsamı artırmak için daha fazla test senaryosu yazılabilir

AMAÇ :

Test Kapsamının ölçülebilir olması

Test tasarımlının kaliteli yapılması



Beyaz Kutu Test Tasarım Teknikleri

Yapı bazlı test tasarım teknikleri

Beyaz kutu testi, yazılımın iç çalışma mantığına dayanır:

- ✓ **Bileşen seviyesi:** bir yazılım bileşeninin yapısı, örnek : komutlar, kararlar, dallar ve yollar
- ✓ **Entegrasyon seviyesi:** Test için ele alınan yapı bileşenlerin birbirlerini nasıl çağrırdığını gösteren bir çağrı ağaçları olabilir
(örnek modüllerin diğer modüllerini çağrırdığı bir diyagram)
- ✓ **Sistem seviyesi:** Yapı; menü yapısı, iş süreci veya web sayfası yapısı olabilir



Beyaz Kutu Test Tasarım Teknikleri

İfade Kapsama Testi

(Statement Coverage Testing)

Karar Kapsama Testi

(Decision CoverageTesting)

Diger Yapısal Testler



Beyaz Kutu Test Tasarım Teknikleri

İfade Kapsama Testi

(Statement Coverage Testing):

- ✓ Bu test teknigi ile yazılım kod parçasındaki kod bloğunda bulunan döngüler kontrol edilir.
- ✓ Bu döngüler için bir veya birden fazla olacak şekilde oluşturulan test senaryoları ile test yapılır.
- ✓ Yazılım geliştiriciler tarafından kullanılabilir.
- ✓ Birim test, entegrasyon test seviyesinde kullanılabilir.
- ✓ **Statement coverage**, yürütülen tüm durumların sayısı ile toplam durum sayılarının bölümü 100 ile çarpılarak bulunur.

Satır Kapsamı

Segment Kapsamı

$$S.C. = (\text{Çalıştırılan Test Durum Sayısı} / \text{Toplam Test Durum Sayısı}) \times 100$$



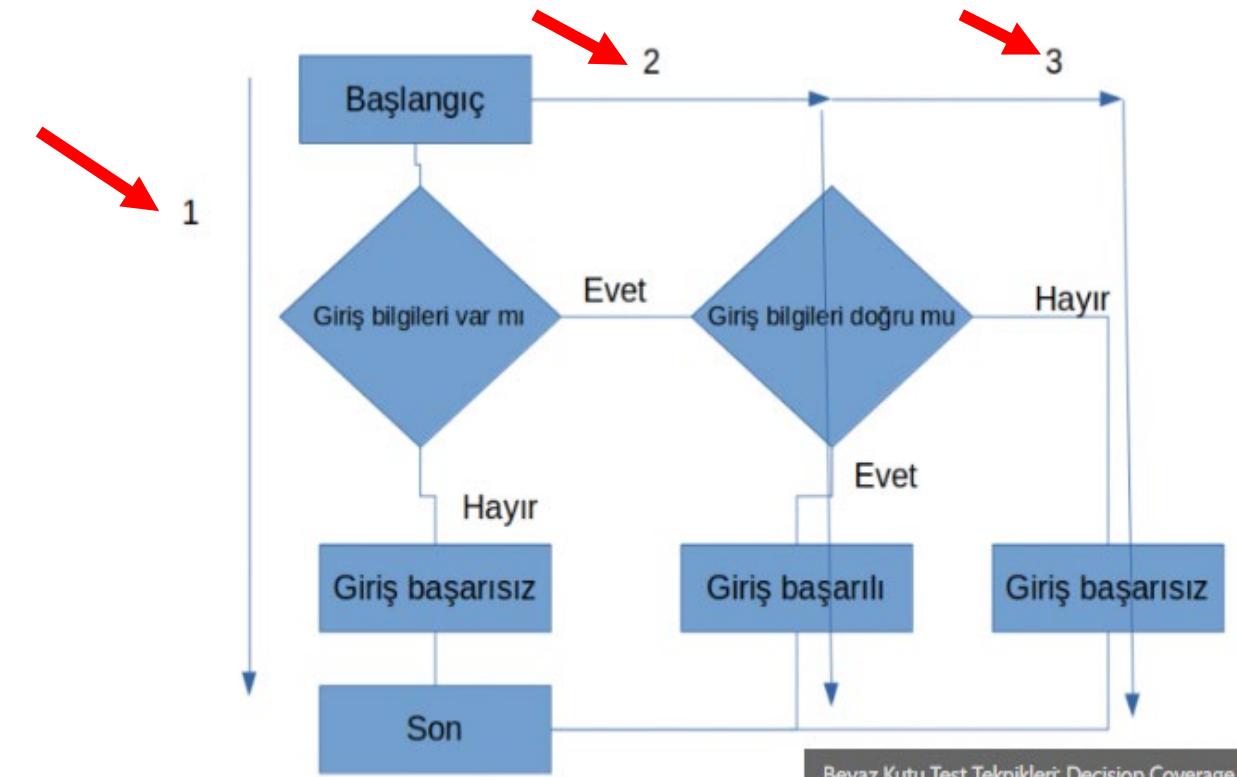
Beyaz Kutu Test Tasarım Teknikleri

**Karar noktalarının kapsamı
(Decision Coverage):**

- ✓ Kod bloğu içerisinde çalıştırılan döngülerin oranını veren birimdir.
- ✓ Tüm (branch) dalların ya da algoritmada ki döngülerin işletilmesi için gerekli olan en az test sayısı bulunur.

Branch coverage

All-edges coverage



Beyaz Kutu Test Teknikleri: Decision Coverage



Beyaz Kutu Test Tasarım Teknikleri

Diğer Yapısal Testler

Kod kapsamı (Code Coverage): Kod parçalarından hangilerinin çalıştırılıp çalıştırılmadığını belirleyebilmek için kullanılan birimdir.

Kod parçası bloğunda test için kapsam aşağıdaki oran ile belirlenir.

Kapsam= $(\text{Çalıştırılan Test Durum Sayısı} / \text{Toplam Test Durum Sayısı})$

Linear Code Sequence And Jump (LCSAJ) (Olabilecek Tüm Durumların Kapsanması)

- ✓ Test senaryolarının LCSAJ' leri严格执行mak için tasarlandığı bir beyaz kutu test tasarım tekniğidir. (**kod kapsamı ile + dal kapsamı**)
- ✓ Kapsam kavramı diğer test seviyelerinde de uygulanabilir.
- ✓ Bir test senaryo grubu koşturulduğunda elde edilen LCSAJ yüzdesinin anlamı:

100% LCSAJ 100% ifade kapsamı

100% karar kapsamı
anlamına gelir.



Deneyimsel Temelli Test Tasarım Teknikleri



Deneyimsel Temelli Test Tasarım Teknikleri

- Tecrübeye dayalı teste, testler benzer uygulamalar ve teknolojilerle daha önce çalışmış test uzmanının becerisine, sezgilerine ve tecrübesine dayanılarak türetilir.
- Sistematik teknikleri artırmak için kullanıldığında bu teknikler, resmi teknikler tarafından kolayca yakalanamayan özel testleri belirlemek için kullanılabilir.
- Fakat bu tekniğin etkisi test uzmanının tecrübesine bağlı olarak çeşitlilik gösterecektir.



Deneyimsel Temelli Test Tasarım Teknikleri

Hata Tahminleme - Error Guessing

- Deneme yanılma
- İlk test olarak kullanılmamalı
- ✓ Tamamlayıcıdır
- ✓ En uç noktalar bulunur
- Kişisel deneyimler ön planda
- ✓ Önceden benzerini test etmiş olmak
- ✓ Hataları koklamak
- Method yoktur
- Eski defect haritası kullanılabilir
- Test script yoktur

- **Hata listeleri**
- **Saldırı**



Deneyimsel Temelli Test Tasarım Teknikleri

Keşif Temelli Testler

- Tester test yapar
- Eş zamanlı tasarım ve çalışma
- Döküman yok
- Script yok
- Amaç öğrenmek
 - ✓ Zayıf yanları
 - ✓ Güçlü yanları
 - ✓ * Ne yapar
 - ✓ * Ne yapmaz
- Diğer test teknikleri de kullanılabilir

Analizin olmadığı ve
surenin kısıtlı olduğu
durumlarda
kullanılabilir.

Sınır değer testi

Eşdeğer aralık testi



Test Tasarım Teknikleri

Uygun Test Tekniğinin Seçimi-Choosing Test Techniques

En iyi teknik nedir?

- Her teknik iyidir
- Doğrusunu seçmek önemli

Önemli faktörler

- Sistemin türü
- Standartlar
- Gereksinimler
- Risk seviyesi
- Risk türleri
- Dökümantasyon, zaman, testçi deneyimleri...





BATCH : **BATCH 59**

LESSON : **ISTQB-05**

DATE : **17.06.2022**

SUBJECT : **ISTQB**

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



Test Yönetimi



Test Yönetimi

Hedefler, her bir modülü tamamladıktan sonra neler yapabileceğinizi tanımlar.



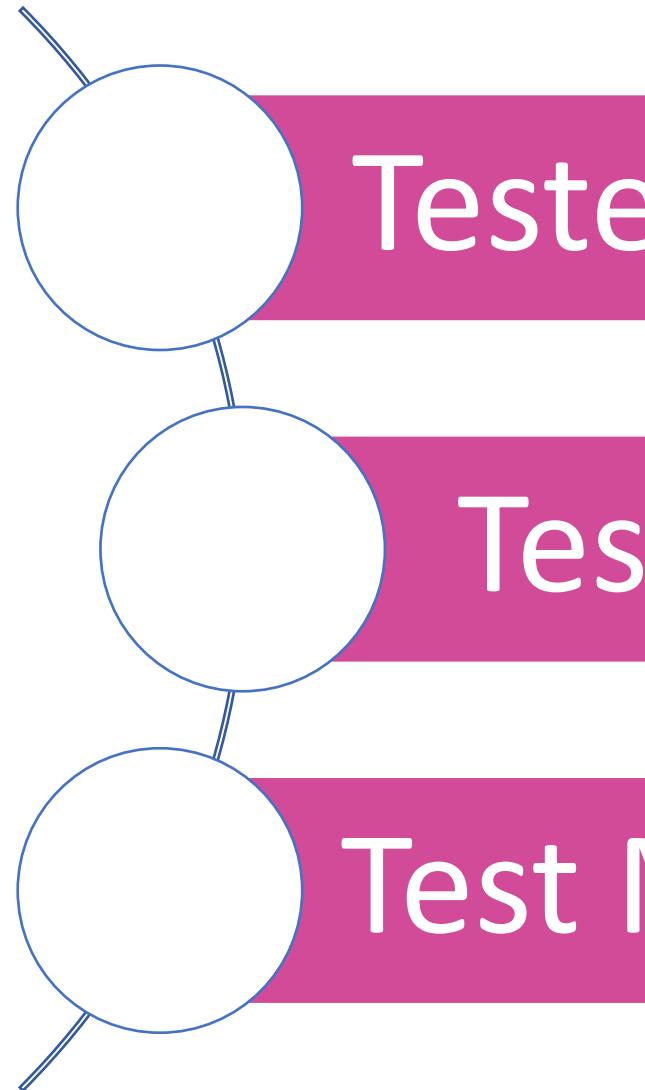


Test Organizasyonu



Test Yönetimi - Test Organizasyonu

Keywords





Test Yönetimi - Test Organizasyonu

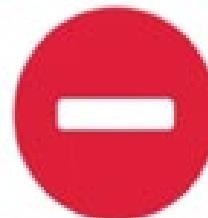
Bağımsızlık Avantaj ve Dezavantajları



Bağımsız test uzmanları önyargısızdır.



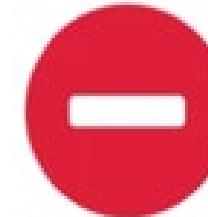
Bağımsız test uzmanları varsayımları doğrulayabilir.



Yazılım geliştirme ekibinden soyutlanma



Yazılımcıların kalite sorumluluk hislerinin kaybolması



Bağımsız test uzmanlarına karşı önyargılı yaklaşım



Test Yönetimi - Test Organizasyonu

Bağımsızlık seviyeleri

- ✓ Bağımsız test uzmanı yoktur; **yazılımcılar kendi kodlarını test eder**
- ✓ Yazılım geliştirme ekiplerinin içindeki **bağımsız test uzmanları**
- ✓ Proje yönetimine veya yönetici kadroya rapor veren, **organizasyon içindeki bağımsız test ekibi**
- ✓ **Şirket dışından bağımsız test uzmanları**
- ✓ Kullanılabilirlik testi uzmanları, güvenlik testi uzmanları veya sertifikasyon testi uzmanları (**bir yazılım ürününü standartlara ve düzenlemelere göre sertifikalandıran**) gibi spesifik **test çeşitleri için bağımsız test uzmanları**
- ✓ **Dış kaynak kullanımı hizmeti veren şirketlerden gelen bağımsız test uzmanları**



Test Yönetimi - Test Organizasyonu

Geliştiricilerin Testi



Artıları

- Kodu İyi Bilir
- Testçinin Kaçırdığı Hataları Görür
- Hatanın Çözümü Kolaydır

Eksileri

- Kendi İşini Bozmak İstemez
- Oluşandan Çok Olması Gerekeni Görür
- Öznel Değerlendirmeler Yapar

Test
uzmanı
yok



Test Yönetimi - Test Organizasyonu

Ayrık Test Grubu



Artıları

- Test Bilgi ve Deneyimi
- Grup Olarak Kalite Odaklı
- Yazılımı Tarafsız Değerlendirme
- Test Yapmak Esas Görevi

Eksileri

- Başka Grupların Baskısı
- Kısıtlı Yardımlaşma
- Kara Kutu Teste Odaklanması
- Yalnızlaştırılma Tehlikesi

Ekip Dışı
Test Ekibi



Test Yönetimi - Test Organizasyonu

Test Danışmanları



Artıları

- Test Bilgisi ve Deneyimi
- Yazılımı Tarafsız Değerlendirme
- Test Yapmak Esas Görevi

Eksileri

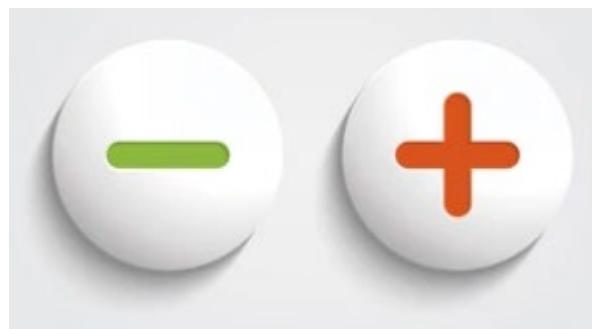
- Başka Grupların Baskısı
- Kısıtlı Yardımlaşma
- Kara Kutu Teste Odaklanması
- Yalnızlaştırılma Tehlikesi

Bağımsız
Test
Uzmanları



Test Yönetimi - Test Organizasyonu

**Outsource Test Hizmeti
(Testing As A Service)
(3rd Party)**



Artıları

- Yüksek Test Bilgisi ve Deneyimi
- Yazılımı Tarafsız Değerlendirme
- Farklı Firmalarda Test Deneyimleri

Eksileri

- Maliyeti Yüksek
- Firmanın Ürünleri Hakkında Kısıtlı Bilgi
- Edinilen Deneyimi Dışarı Paylaşma

**Dış
Kaynak**



Test Yönetimi - Test Organizasyonu

Test Organizasyonu ve Bağımsızlık

- ❖ Büyük, karmaşık ve riskli olan projelerde, birkaç seviyenin veya tüm seviyelerin **bağımsız test uzmanları tarafından gerçekleştirilmesi** genellikle en iyi uygulamadır.
- ❖ Yazılım geliştirme ekibi de özellikle **daha alt seviyelerde teste katılabılır**, ancak yeterince objektif olmadıklarından etkinlikleri sınırlıdır.
- ❖ Bağımsız test uzmanları, sadece testleri yürütmenin yanında **test süreçlerinin ve kurallarının gereksinimlerini belirleme yetkisine ve sorumluluğuna da sahip olabilir**.





Test Yönetimi - Test Organizasyon

Test Organizasyonunda Görevler

- Test aktivitelerini ve görevlerini planlar, izler ve koordine eder.
- Bu görev proje yöneticisi, yazılım geliştirme yöneticisi veya kalite yöneticisi tarafından gerçekleştirilebilir.

**Test Lideri/
Test Yöneticisi/
Test koordinatörü**

- Test analizi, tasarımları, çeşitleri ve otomasyonu konusunda uzman kişilerdir.
- Proje riskine, test seviyesine bağlı olarak farklı kişiler bu rolü üstlenebilir.

**Test Uzmanı/
Test Mühendisi**



Test Yönetimi - Test Organizasyonu

Test liderinin görevleri

- Test stratejisini koordine etme ve planlamak
- Projenin test stratejisini ve kuruluşun test politikasını oluşturmak
- Test yaklaşımının diğer proje adımlarını etkilemesini sağlamak
- Test testleri planlamak
- Test faaliyetlerini başlatmak, test sonuçlarını gözlemlemek ve çıkış kriterlerini kontrol etmek
- Test sonuçlarına ve ilerlemeye bağlı olarak test planlarını güncellemek





Test Yönetimi - Test Organizasyonu

Test liderinin görevleri

- Test yazılımlarının izlenebilirliği için ile yapılandırma yönetimini hazırlamak
- Test ilerleyişini ölçmek, testin ve yazılımın kalitesini değerlendirmek için uygun metrikleri belirlemek
- Test otomasyon kullanımına karar vermek
- Test araçlarını seçmek ve test uzmanlarını eğitmek
- Test ortamının uyarlanması ile ilgili karar verme
- Test özet raporlarını hazırlamak.





Test Yönetimi - Test Organizasyonu

Test uzmanının görevleri

- Test planlarını gözden geçirmek ve katkı sağlamak
- Kullanıcı gereksinimlerini ve test için oluşturulmuş modelleri analiz etmek, gözden geçirmek ve değerlendirmek
- Test gereksinimlerini oluşturmak
- Test ortamını hazırlamak
- Test verisini hazırlamak

TESTER



Test Yönetimi - Test Organizasyonu

Test uzmanının görevleri

- Tüm test seviyelerinde testleri uyarlamak, yürütmek ve kayıt altına almak, sonuçları değerlendirmek ve beklenen sonuçlardan sapmaları belgelemek
- Test yönetim ve izleme araçlarını kullanmak
- Testleri otomasyona geçirmek
- Bileşenlerin ve sistemlerin performansını ölçmek
- Başkaları tarafından geliştirilen testleri gözden geçirme





Test Yönetimi - Test Organizasyonu

Test uzmanında bulunması gereken özellikler

- ❖ Uygulama veya İş Alanı Bilgisi
- ❖ Teknolojiye Hakimiyet
- ❖ Test Bilgisi



Soru Çözümleri



Soru Çözümleri-1

Sınır değer testi(Boundary value testing)



- A. Eşdeğer aralık testleri equivalence partitioning ile aynıdır**
- B. Giriş ve çıkış uyumluluğunu, sınır değerlerinde , sınırın altında ve üstünde test eder**
- C. Giriş koşullarının kombinasyonlarını test eder**
- D. Beyaz kutu test stratejisinde kullanılır**

- ✓ Sınır değer testleri değişim noktalarında yazılımın kararlılığını test etmek üzere yapılır.
- ✓ Sınır değerler kontrol edilir(**maksimum ve minimum değerleri**)
- ✓ Bu yüzden eşdeğer aralıkların çıkartılması ve en büyük ve en küçük değişim noktalarının saptanması gereklidir
- ✓ **2-değerli ve 3-değerli sınır değer testi** yaklaşımı vardır.
 - 2-değerli yaklaşımında değişim noktası ve bir sonraki değer alınırken
 - 3-değerli yaklaşımında değişim noktası, bir önceki ve bir sonraki değerleri de içerisinde alan üç farklı değer test edilir.



Soru Çözümleri-2

Neyin test edileceğini önceliklendirirken en önemli madde hangisidir?

- A. Mümkün olduğunca çok arıza bulma
- B. Yüksek riskli alanları test etme
- C. iyi bir test kapsamı elde etme
- D. En kolay olanı test etme

- Yazılımın doğru çalıştığından emin olunduğunda

şartlara göre
doğru kabul edilir

- Yazılımın risk değerlendirmesine bağlı olarak

Kesin doğru....



Soru Çözümleri-3

Bir toptancı yazıcı kartuşları satıyor. Minimum sipariş miktarı 5 adettir. 100 veya daha fazla yazıcı kartuşu siparişlerinde %20 indirim yapılmaktadır. Sipariş edilen yazıcı kartuşu sayısı için çeşitli değerleri kullanarak test senaryoları hazırlamanız istense, aşağıdaki gruplardan hangisini kullanırsınız?

- A. 5, 6, 20
- B. 4, 5, 80
- C. 4, 5, 99
- D. 1, 20, 100





Soru Çözümleri-4

Test durumları ne zaman üretilir?

- A. Testler kayıt edilirken
- B. Testler analiz edilirken
- C. Test konfigürasyonu oluşturulurken
- D. Test belirtimleri çıkartılırken





Soru Çözümleri-5

**Hangi durumda testler için Tamam, dur
diyebiliriz ?**

- A. Zaman bittiğinde
- B. Yeterli seviyede güven düzeyine ulaştığında
- C. Daha fazla hata bulunamadığında
- D. Kullanıcılar ciddi hata bulamadığı durumda



Soru Çözümleri-6

Bir stok kontrol sistemindeki sipariş numaraları 10000 ile 99999 (sayılar dahil)arasında değişebilir. Aşağıdaki girdilerden hangisi, yalnızca eşdeğer geçerli aralık ve geçerli sınırlar için test tasarlamadan bir sonucu olabilir?

- A. 1000, 50000, 99999
- B. 9999, 50000, 100000
- C. 10000, 50000, 99999
- D. 10000, 99999, 100000

- ✓ Yazılım için **aynı sonucu verecek girdi değerleri** belirlenir.
- ✓ Genelde iki sınıf tanımlanabilir:
 - **Geçerli Giriş Sınıfı :**
Yazılımin kabul edeceği
geçerli giriş aralığıdır
 - **Geçersiz Sınıf :**
Formal olmayan veya
geçerli aralık dışı veri
kümesidir.
- ✓ Tanımlanan her kümeye 1 test
durumu yazılır.



Soru Çözümleri-7

Kod gözden geçirmelerin önemli bir faydası da:

- A. Çalıştırma ortamı hazır olmadan kodun test edilmesini sağlaması**
- B. Kodu yazan kişi tarafından yapılabilmesi**
- C. Deneyimsiz personel tarafından yapılabilmesi**
- D. Gerçeklestirmesinin ucuz olması**

Gözden Geçirmeler Sayesinde

- ✓ Daha az defect ve yazılım maliyeti
- ✓ Yazılım geliştirme faaliyetlerinde verim artar
- ✓ **Yazılım geliştirme süresini kısaltır**
- ✓ Hata seviyesini düşürür, daha az ciddi hatalar
- ✓ Müşteri ilişkilerini geliştirir
- ✓ Deneyim paylaşımı artar
- ✓ **Standartlara uyum artar**



Soru Çözümleri-8

Ne kadar yeniden test gerekeğine nasıl karar verirsiniz?

- A. Önceden yapılan benzer projelere bakarak
 - B. Geliştirme ekibi ile görüşerek
 - C. Regresyon için ayrılan zaman bakarak
 - D. a & b
-



Soru Çözümleri-9

Test tasarım tekniğinin amacı,

- A. Test senaryolarının değil, yalnızca test koşullarının belirlenmesi**
- B. Test koşullarını değil, yalnızca test senaryolarını tanımlama**
- C. Test koşullarının ve Test senaryolarının belirlenmesi**
- D. Test koşullarının veya Test senaryolarının belirlenmesi**



➤ **Test Tasarımı**

- ✓ Hatalı durumun belirlenmesi
- ✓ Koşullar
- ✓ Test oracle
- ✓ istenilen (expected result)- gerçeklesen sonuç (actual result)-> bu bağlamda senaryolar belirlenmelidir

Hatalı durumdan bağımsız olarak doğru durumun bulunması için gerekli bilgiler "testoracle" denir.

➤ **Testin Tatbiki (implementation)**

- ✓ Gruplama
- ✓ Sıralama

Test set



Soru Çözümleri-10

Gözden geçirmeler hakkında aşağıdaki ifadelerden hangisi doğrudur?

- A. Gözden geçirmeler, kullanıcı gereksinimleri üzerine yapılamaz**
- B. Gözden geçirmeler, kodun test edilmesinin en az etkili yolu**
- C. Gözden geçirmelerin, test planlarında hatalar bulması pek olası değildir**
- D. Şartnameler, kodlar ve test planları hakkında gözden geçirmeler yapılmalıdır.**



3.Techical Review

- Dokümane edilmiş, tanımlanmış defect bulma süreci olan
- Teknik kişileri ve çalışma arkadaşlarını sürece dahil eden
- Yönetimin isteği bağlı olarak katıldığı
- Bulgular listesini, yazılımın gereksinimleri karşılayıp karşılamadığı ile ilgili kararı ve uygun olan durumlarda bulgularla ilgili önerileri içeren gözden geçirme raporunun hazırlanması
- İdeal olarak eğitimli moderatör tarafından yönetilir.
- İsteğe bağlı kontrol listesi kullanılır.

Amaç: Tartışma, karar verme, alternatifleri değerlendirme, defect bulma, teknik problemleri çözme ve gereksinimlere, planlara, düzenlemelere ve standartlara uyumu kontrol etme

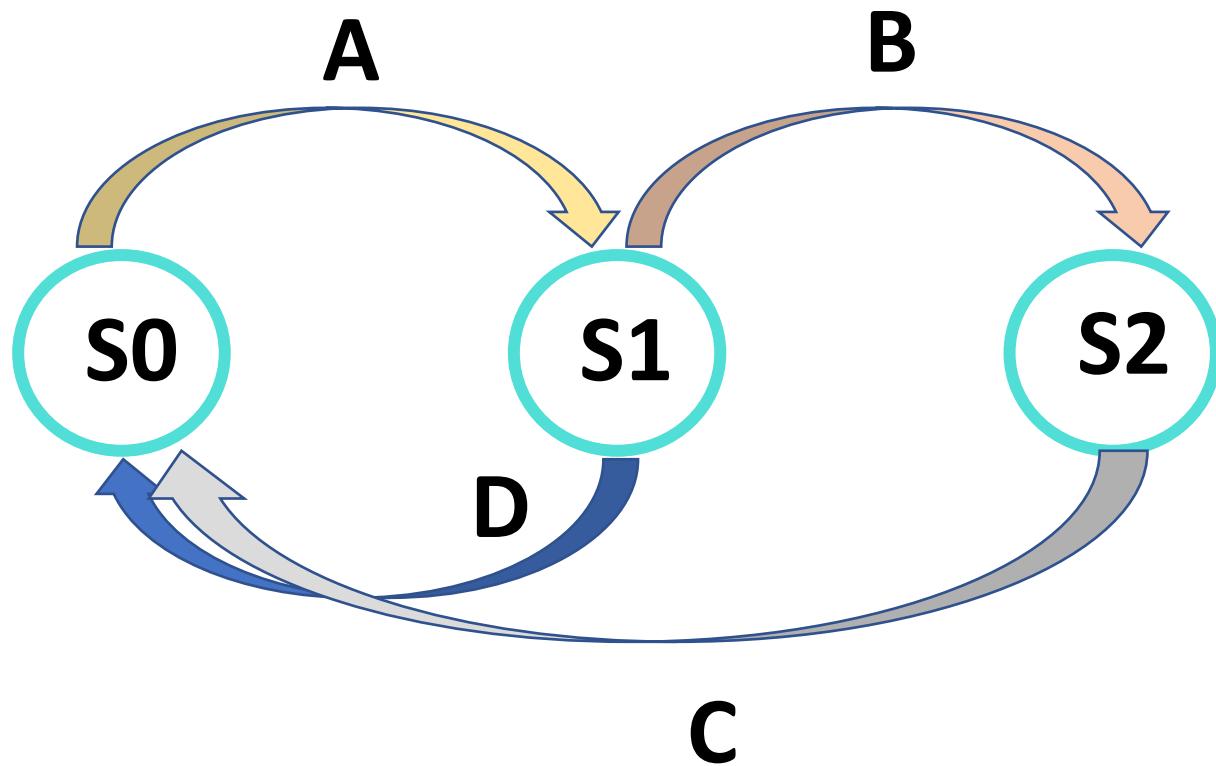


Soru Çözümleri-11

Durum geçiş tablosu göz önüne alındığında, test durumlarından hangisi aşağıdaki durum geçişleri dizisini kapsar?

S1 SO S1 S2 SO

- A. B, C, A, D
- B. A, B, C, D
- C. D, A, B
- D. D. A, B, C





Soru Çözümleri-12

Bir yazılım çıktısını gözden geçirirken ana hedef nedir?

- A. Potansiyel uygulama hatalarını bir test belirtimleri üzerinden saptamak
- B. Herhangi bir yazılım işi ürünündeki kusurları tanımlamak
- C. Gereksinim spesifikasyonundaki yazım hatalarını tespit etmek
- D. Koddaki standart tutarsızlıklarını belirlemek



Soru Çözümleri-13

Aşağıdakilerden hangisi normalde bir test için bir hedef olmamalıdır?

- A. Yazılımdaki hataları bulmak
- B. Yazılımın canlıya çıkmak için hazır olup olmadığını değerlendirmek
- C. Yazılımın çalışmadığını göstermek
- D. Yazılımın doğru olduğunu kanıtlamak.



Soru Çözümleri-14

Aşağıdaki gereksinimlerden hangisi test edilebilir?

- A. Sistem kullanıcı dostu olacaktır
- B. Sistemin emniyet açısından kritik bölümleri SIFIR arıza içermelidir
- C. Belirtilen tasarım yükü için tepki süresi bir saniyeden az olmalıdır
- D. Sistem taşınabilir olacak şekilde yapılacaktır.



Soru Çözümleri-15

Aşağıdakilerden hangisi sistem testinin bir parçası DEĞİLDİR:

- A. İş süreci tabanlı test**
- B. Performans, yük ve stres testi**
- C. Gereksinimlere dayalı test**
- D. Yukarıdan aşağıya entegrasyon testi**

Sistem (System) Testi

- ✓ Sistem testi, genellikle sistemin gerçekleştirileceği uçtan uca görevleri ve bu görevleri eksiksiz ve entegre bir sistemde gerçekleştirir. Sistem testi, bir bütün olarak sistemin uçtan uca davranışına odaklanmalıdır(**hem işlevsel hem de işlevsel olmayan**)
- ✓ Gereksinimlere göre sistemin uygunluğunun kontrol edilmesini sağlar.
- ✓ Bileşenlerin genel etkileşimini test eder.
- ✓ Yük, stres, performans, güvenilirlik ve güvenlik testlerini içerir..
- ✓ Sistem testi, sistemin şartnameye uygun olduğunu doğrulamak için yapılan son testtir.



Soru Çözümleri-16

Hata ayıklama (debugging) faaliyetlerini kim uygular?

- A. Geliştiriciler
- B. Analistler
- C. Testçiler
- D. Kaliteciler



Soru Çözümleri-17

Aşağıdakilerden hangisi doğrudur?

- A. Etki analizi, regresyon testinde bulunan bir kusurun sistem üzerindeki etkisini değerlendirir**
- B. Etki analizi, regresyon test ekibine yeni katılan bir kişinin etkisini değerlendirir**
- C. Etki analizi, regresyon testinde, hatanın doğru şekilde düzeltildip düzeltildiğini değerlendirir**
- D. Etki analizi, regresyon testinin ne kadar olacağını belirlemek için sistemdeki bir değişikliğin etkisini değerlendirir.**



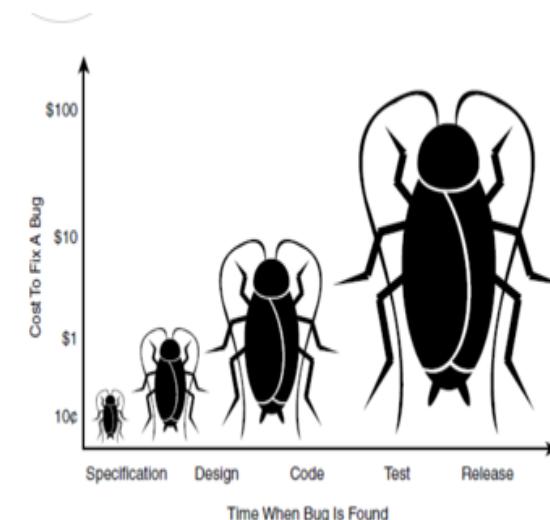
Bir proje kapsamında değişik faktörlerden dolayı projenin herhangi bir anında değişiklik talebi ortaya çıkabilir. Böyle bir durum projenin süresini ve bütçesini etkileyebilir. Bu nedenle bu değişiklikler karşısında etki analizi yapılmalıdır.



Soru Çözümleri-18

Testlerin yazılım yaşam döngüsünün erken safhalarında başlamasının ana faydası nedir?

- A. Testlerin test aşaması sırasında tasarlanmasından daha ucuzdur
- B. Kusurların koda girmesini önlemeye yardımcı olur
- C. Erken tasarlanan testler daha sonra tasarlanan testlerden daha etkilidir
- D. Test edenler meşgul olduğunda test aşamasında zaman kazandırır



Hatalar projenin erken safhalarında bulunursa, o hatayı düzeltmek daha kolay ve hatanın düzeltilme maliyeti o kadar az olur

Hatanın geç bulunması...daha fazla belgeye, koda, teste neden olur, bu yüzden maliyet artar



Soru Çözümleri-19

Sistem testi hangi gereksinimlerin gerçekleştiğini kontrol etmelidir?

- A. Yalnızca işlevsel olmayan gereksinimler,**
- B. Yalnızca işlevsel gereksinimler,**
- C. İşlevsel olmayan gereksinimler ve İşlevsel gereksinimler**
- D. İşlevsel olmayan gereksinimler veya İşlevsel gereksinimler**



Sistem (System) Testi

- ✓ Sistem testi, genellikle sistemin gerçekleştirileceği uçtan uca görevleri ve bu görevleri eksiksiz ve entegre bir sistemde gerçekleştirir. Sistem testi, bir bütün olarak sistemin uçtan uca davranışına odaklanmalıdır(hem işlevsel hem de işlevsel olmayan)
- ✓ Gereksinimlere göre sistemin uygunluğunun kontrol edilmesini sağlar.
- ✓ Bileşenlerin genel etkileşimiini test eder.
- ✓ Yük, stres, performans, güvenilirlik ve güvenlik testlerini içerir..
- ✓ Sistem testi, sistemin şartnameye uygun olduğunu doğrulamak için yapılan son testtir.





Soru Çözümleri-20

Konfigürasyon yönetimi ne zaman tanımlanır?

- A. Test planlaması sırasında
- B. Test analizi sırasında
- C. Testler yürütülürken
- D. Çıkış kriterlerini değerlendirirken



Soru Çözümleri-21

Aşağıdakilerden hangisi regresyon testinin karakteristik özelliklerindenidir?

- i) Regresyon testi SADECE bir kez çalışır
 - ii) Düzeltmeler yapıldıktan sonra regresyon testi yapılır
 - iii) Regresyon testi genellikle otomatiktir
 - iv) Regresyon testlerinin muhafaza edilmesine gerek yoktur
- A. ii, iv.
- B. ii, iii.
-
- C. i, iii, iv.
- D. iii.

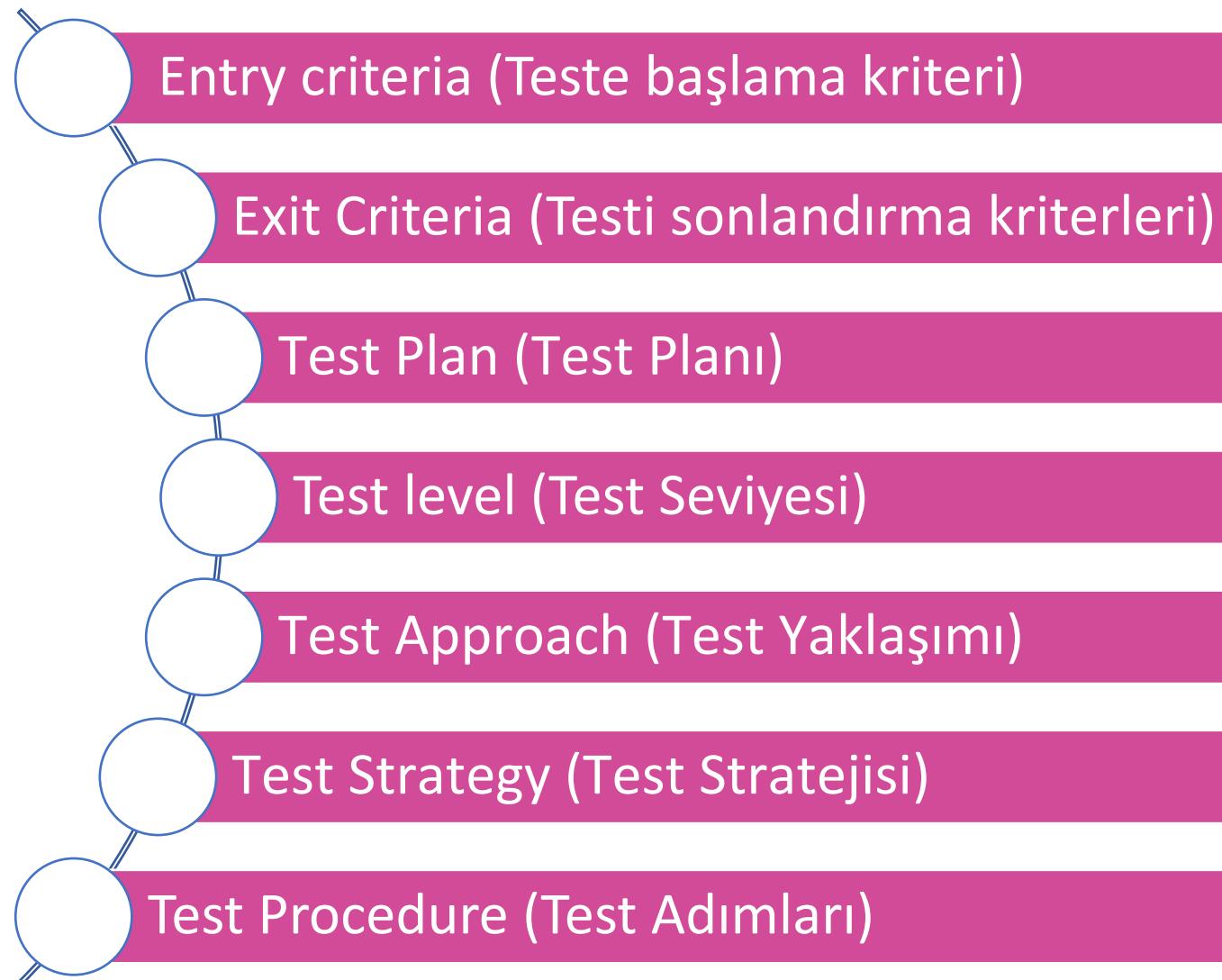


Test Planlama ve Tahminleme



Test Yönetimi - Test Planlama ve Tahminleme

Keywords





Test Yönetimi - Test Planlama ve Tahminleme

- Planlama bir master test planında yer alabileceği gibi sistem testi ve kabul testi gibi test seviyeleri için ayrı test planlarında da yer alabilir
- Test planının ana hatlarına "Yazılım Testi Dokümantasyonu Standardı" (**IEEE Std 829-1998**) kapsamında deðinilmektedir
- Test planı proje kapsamındaki test ile alakalı tüm işleri içerisinde barındıran ve mantık sıralamasına sokan proje planıdır.



Test Yönetimi - Test Planlama ve Tahminleme

Test planlama sürekli devam eden bir aktivitedir.

Planlamayı etkileyen ana faktörler;

- ✓ Organizasyonun test politikası,
- ✓ Test kapsamı
- ✓ Hedefler
- ✓ Riskler ve sınırlandırmalar
- ✓ Önem
- ✓ Test edilebilirlik
- ✓ Kaynakların elverişliliği



Test Yönetimi - Test Planlama ve Tahminleme

Test Planlama Adımları

- ✓ Kapsamın ve test hedeflerinin belirlenmesi
- ✓ Genel test yaklaşımının tanımlanması
- ✓ Test aktivitelerini yazılım yaşam döngüsü adımlarıyla entegre edilmesi
- ✓ Test organizasyonunu oluşturulması
- ✓ Test aktivitelerinin kaynak ve zaman planamasının yapılması
- ✓ Test dokümantasyon hazırlıklarının yapılması
- ✓ Test metriklerinin belirlenmesi
- ✓ Test prosedürlerinin ayrıntı seviyesinin belirlenmesi

Test planının içeriğini
de oluşturur



Test Yönetimi - Test Planlama ve Tahminleme

IEEE 829 Standard Test Plan Şablonu

Introduction

Features to be tested

Staffing and training needs

Schedule

Test deliverables

Environmental needs

Approach

Risks and contingencies

Test items

Features not to be tested

Item pass/fail criteria

Approvals

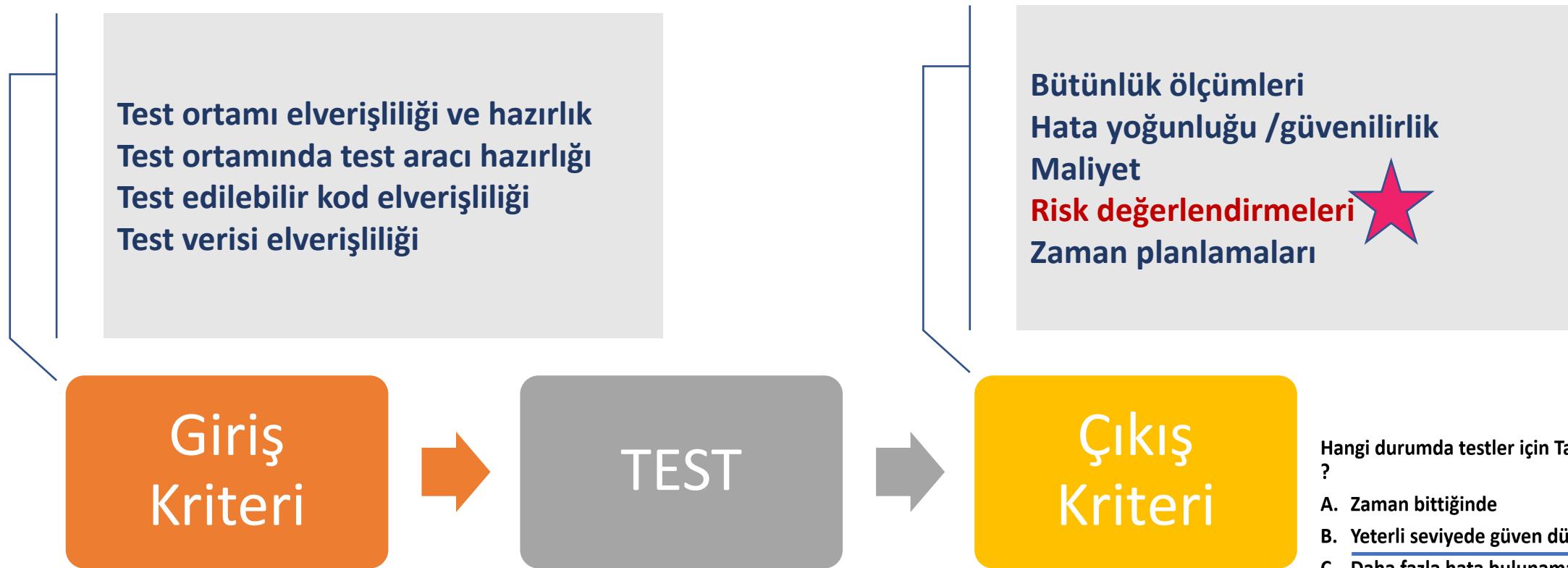
Test tasks

Responsibilities

Suspension and resumption criteria



Test Yönetimi - Test Planlama ve Tahminleme



- Hangi durumda testler için Tamam, dur diyebiliriz ?
- A. Zaman bittiğinde
 - B. Yeterli seviyede güven düzeyine ulaşıldığında
 - C. Daha fazla hata bulunamadığında
 - D. Kullanıcılar ciddi hata bulamadığı durumda



Test Yönetimi - Test Planlama ve Tahminleme

Test Başlama Kriterleri- Entry Criteria

Testin ne zaman başlayacağını belirler

- ✓ Personel, araç-gereç, materyal ihtiyaçları
- ✓ Test edilecek ürünün hazır olması

Smoke test

- ✓ Test datası olmalı
- ✓ Testi sonlandırma kriterlerinin hazır olmalıdır



Test Yönetimi - Test Planlama ve Tahminleme

Test Sonlandırma Kriterleri- Exit Criteria

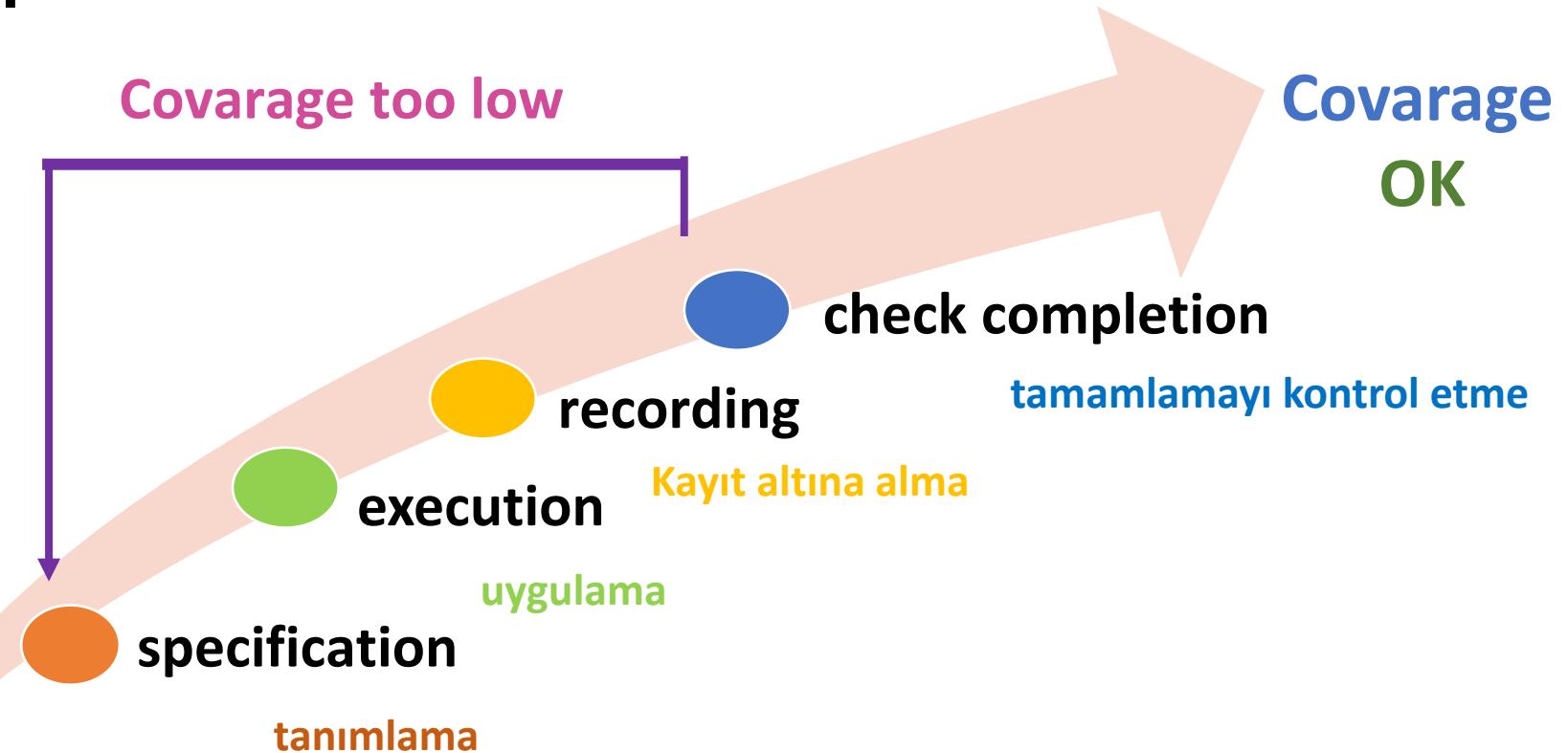
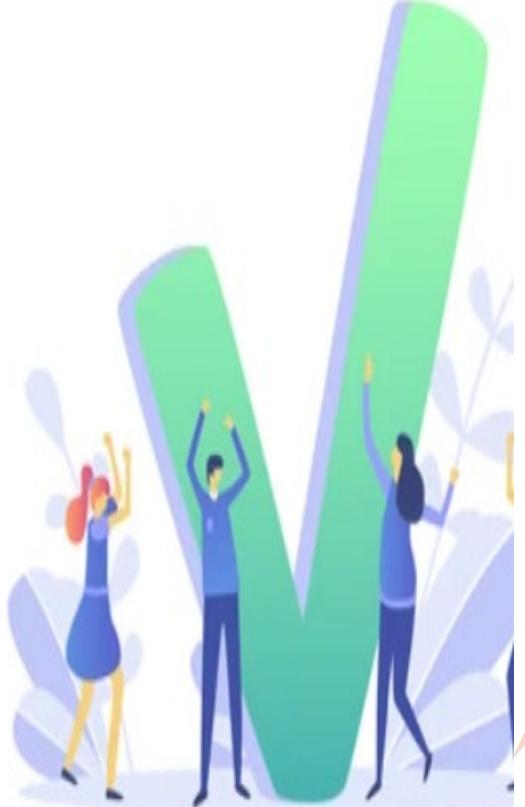
Testin ne zaman sonlandırılacağını belirler

- ✓ Bir takım testlerin başarılı olduğunda
- ✓ Belirli bir code coverage sağlandığında
- ✓ Hata dağılımı kavranıldığında
- ✓ Bütçe, zaman tamamlandığında
- ✓ Hata bulunma sıklığı azaldığında
- ✓ Kabul edilebilir riskler taşıdığında



Test Yönetimi - Test Planlama ve Tahminleme

Covarage : kapsam alanı





Test Yönetimi - Test Planlama ve Tahminleme

daha önceki veya benzer projelere ya da genel değerlere dayanarak test çabasını tahmin etme

Test Tahminlemesi

Metrik Bazlı Yaklaşım

Uzman Bazlı Yaklaşım

testte yapılacak işlerin sahibi veya uzmanlar tarafından yapılan tahminlere dayanarak görevleri tahmin etme



Test Yönetimi - Test Planlama ve Tahminleme

Test Tahminlemesi

- **Testçi sayısı**
 - ✓ Proje konusu
 - ✓ Proje riski
 - ✓ Ürün riski
- **Test süresi**
 - ✓ Test case sayısı
 - ✓ Gereksinim sayısı
 - ✓ İşletim süresi
- **Karmaşıklığı/Zorluğu**
 - ✓ Kullanılan methodlar
 - ✓ Yeni teknolojiler
 - ✓ Güvenlik açıkları



test süresi = $(\text{test_case_sayısı} \times \text{test_işletim_süresi}) / \text{testçi_sayısı}$



Test Yönetimi - Test Planlama ve Tahminleme

Test eforu tahmin edildiğinde kaynaklar belirlenebilir ve bir zaman çizelgesi çizilebilir.

Test eforu tahmini birçok faktöre bağlı olabilir:

- ✓ **Yazılımın özellikleri:** test modelleri için kullanılan gereksinim ve diğer bilgilerin kalitesi (örnek : test esası), yazılımın boyutu, problemlü alanın karmaşıklığı, güvenilirlik ve güvenlik için gereksinimler ve dokümantasyon gereksinimleri
- ✓ **Geliştirme sürecinin özellikleri:** kuruluşun kararlılığı, kullanılan araçlar, test süreci, katılan kişilerin becerileri ve zaman kısıtlaması
- ✓ **Test ürünü/çiktısı:** hataların sayısı ve gereken yeniden çalışma eforu



Test Yönetimi - Test Planlama ve Tahminleme

Test stratejisi ve Test yaklaşımı

Test Stratejisi:

- ✓ Projeye özgüdür
- ✓ Ortak kanı verir
- ✓ Süreçler belirlenir

Değerlendirilmesi gereken bileşenler:

- ✓ Yazılım geliştirme stratejileri
- ✓ Müşteri gereksinimleri
- ✓ Test amaçları
- ✓ Proje konusu

Test Yaklaşımı:

Daha küçüktür, test tiplerini gruplar

ISTQB deki tanım : Test yaklaşımı, bir test projesi için test stratejisinin uyarlanmasıdır



Test Yönetimi - Test Planlama ve Tahminleme

Test Yaklaşımı

- ✓ Test yaklaşımı, test planlarında ve test tasarımlarında tanımlanır ve düzenlenir.
- ✓ Genellikle test projesinin amacına ve risk değerlendirmesine dayanarak verilen kararları içerir.
- ✓ Test yaklaşımı test sürecini planlama, uygulanacak test tasarım tekniklerini ve test çeşitlerini seçme, giriş ve çıkış kriterini belirlemek için referans noktasıdır.



Test Yönetimi - Test Planlama ve Tahminleme

Test Yaklaşımı
Örnekler:

Analitik Yaklaşımalar

Metodik
Yaklaşımalar

Süreç veya Standart
Uyumlu Yaklaşımalar

Testin en riskli alanlara
yönlendirildiği risk bazlı test

Arıza bazlı, tecrübe dayalı,
kontrol listesi bazlı ve kalite
özelliği bazlı gibi

Özel standartlar tarafından
belirlenenler veya çeşitli Agile
metotlar gibi



Test Yönetimi - Test Planlama ve Tahminleme

Test Yaklaşımı
Örnekler:

Dinamik Yaklaşımlar

Testin önceden planlanmadığı,
keşif testleri gibi

İstisnai yaklaşımlar

Uzmanlarının önerileri ve
rehberlikler

Regresyon
hassasiyetli
yaklaşımlar

Var olan test materyalinin, test
komut dosyalarının kullanımı
gibi



Test İlerleme Gözetimi ve Kontrolü



Test Yönetimi - Test İlerleme Gözetimi ve Kontrolü

- Test gözetiminin amacı **test işlemleri hakkında geri bildirim ve şeffaflık sağlamaktır.**
- Takip edilecek bilgiler manuel veya otomatik olarak toplanabilir
- Bu bilgiler çıkış kriterini ölçmek için kullanılabilir
- Planlanan zaman çizelgesine ve bütçeye göre ilerlemeyi değerlendirmek için de metrikler kullanılabilir
- Kullanılan birçok test metrikleri vardır



Test Yönetimi - Test İlerleme Gözetimi ve Kontrolü

Yaygın kullanılan metrikler;

- ✓ Hazırlanan test senaryo sayısının planlanan sayıya oranı
- ✓ Test ortamı **hazırlığında** yapılan işin yüzdesi
- ✓ Test senaryosu **yürütmeye** (örnek. çalıştırılan/çalıştırılmayan test senaryosu sayısı ve başarılı/başarısız test senaryoları)
- ✓ Hata ile ilgili bilgi (**canlı değilse->** örnek. hata yoğunluğu, bulunan ve düzeltilen hatalar, arıza oranı ve tekrar testi sonuçları)
- ✓ Testler sonucunda gereksinimlerin, risklerin ve kodun kapsama yüzdesi
- ✓ Test uzmanlarının ürüne **subjektif güveni**
- ✓ Test **kilometre taşlarının tarihleri**
- ✓ Test maliyetleri (bir sonraki hatayı bulma avantajıyla ya da bir sonraki testi çalışma avantajıyla karşılaştırıldığında ortaya çıkan maliyet dahil)



Test Yönetimi - Test İlerleme Gözetimi ve Kontrolü

Test Durum Görüntüleme

- ✓ Plana ne kadar uyuyor
 - ✓ Hatalar
 - ✓ Saptanan
 - ✓ Raporlanan
- ✓ Çözülen
- ✓ Ne kadar daha emek gerekli
- ✓ Test sonuçları takım içinde görülebilir

Teste başladıkten sonra testin gidişatını veya yazılımın kalitesini canlı olarak görüntüleme işine **test durum görüntülemesi** (**test progress monitoring**) denir.



Test Yönetimi - Test İlerleme Gözetimi ve Kontrolü

Test Kontrolü

ISTQB dökümanında verilen örnek raporda testte o anki mevcut durumu gösteren bir çok bilgi yer almaktadır.

- ✓ Defect Density (Hata Yoğunluğu)
- ✓ Failure Rate (Hata Oranı)
- ✓ Test Coverage (Test Kapsamı)
- ✓ Test Monitoring (Test Görüntüleme)
- ✓ Test Report (Test Raporu)

Plandan **sapma olması**
durumunda test caselerin
sapmayı önlemek için
kontrol edilmesidir..



Test Yönetimi - Test İlerleme Gözetimi ve Kontrolü

Örneğin buradaki raporda

Plan Effort

planlanan test için harcanacak emeği gösteriyor ve

Actual Effort ise test işletilirken harcanan gerçek emeği göstermektedir.

System Test Case Summary											
Cycle One											
Test	ID	Test Suite/Case	Status	System	Bug	Bug	Run	Plan	Act	Plan	Actual
				Config	ID	RPN	By	Date	Date	Effort	Effort
1.000 Functionality											
1.001	File	Fail	A	701	1	LTW	1/8	1/8	4	6	6
1.002	Edit	Fail	A	709	1	LTW	1/9	1/10	4	8	8
				710	5						
				718	3						
				722	4						
1.003	Font	Pass	B			JHB	1/10	1/10	4	4	4
1.004	Tables	Warn	B	708	15	JHB	1/8	1/9	4	5	5
1.005	Printing	Skip					1/10		4		
Suite Summary											
							1/10	1/10	20	23	23
2.000 Performance/Stress											
2.001	Solaris Server	Warn	A,B,C	701	1	EM	1/10	1/13	4	8	24 Replan 1/11
2.002	NT Server	Fail	A,B,C	724	2	EM	1/11	1/14	4	4	24 Replan 1/12



Test Yönetimi - Test Raporlama

- ✓ Modül/Birim
- ✓ İhtiyaç(Gereksinim) Numarası
- ✓ Test senaryo no
- ✓ Test sonuç no
- ✓ Testin sonucu
- ✓ Testin tarihi
- ✓ Testi yapan kişi bilgisi
- ✓ Hata no
- ✓ Test ortamı
- ✓ Hata için yapılan tekrar test sayısı ya da bir testin tekrar koşulma sayısı

Test sonuçlarının raporlanması ve değerlendirilmesi için [IEEE 829-1998 standartı](#) Test Sonuç Raporu formatını önermektedir.



Test Yönetimi - Test Raporlama

Şu durumları değerlendirmek için test sırasında ve test seviyesinin sonunda metrikler toplanmalıdır;

- ✓ Test seviyesi için test hedeflerinin yeterliliği
- ✓ Benimsenen test yaklaşımlarının yeterliliği
- ✓ Hedeflere göre testin etkinliği



Yapılandırma Yönetimi



Test Yönetimi - Yapılandırma Yönetimi

Konfigürasyon Yönetimi (versiyon kontrolü)

- ✓ Sistemdeki tüm araçların sürümlerini kontrol eder
- ✓ Birimlerin fonksiyonel ve fiziksel olarak tanımlarını dökümante etmek
- ✓ Sistemdeki araçların durumunu günceller ve raporlar
- ✓ Hata oluşması durumunda hatayı bildirir
- ✓ Sistemdeki tüm araç ve yazılımların güncelliliği denetler
- ✓ Sistemdeki araçların bir birleriyle olan çalışmalardaki sorunları kontrol eder

değişimleri kontrol eden ve herhangi bir sorun çıkması durumunda en **düzgün (stable)** konfigürasyonu yükleyebilen ve **hizmet kesintisini en aza indirmeyi hedef** edinmiş bir yazılım hizmetidir



Test Yönetimi - Yapılandırma Yönetimi

Konfigürasyon Yönetimine Alınabilecek Bileşenler

- ✓ Planlar
- ✓ Süreç tanımları
- ✓ Gereksinimler
- ✓ Tasarım bilgileri
- ✓ Çizimler
- ✓ Ürün spesifikasyonları
- ✓ Kod
- ✓ Derleyiciler
- ✓ Ürün bilgisi dosyaları
- ✓ Ürün teknik yayınları



Risk ve Test Etme



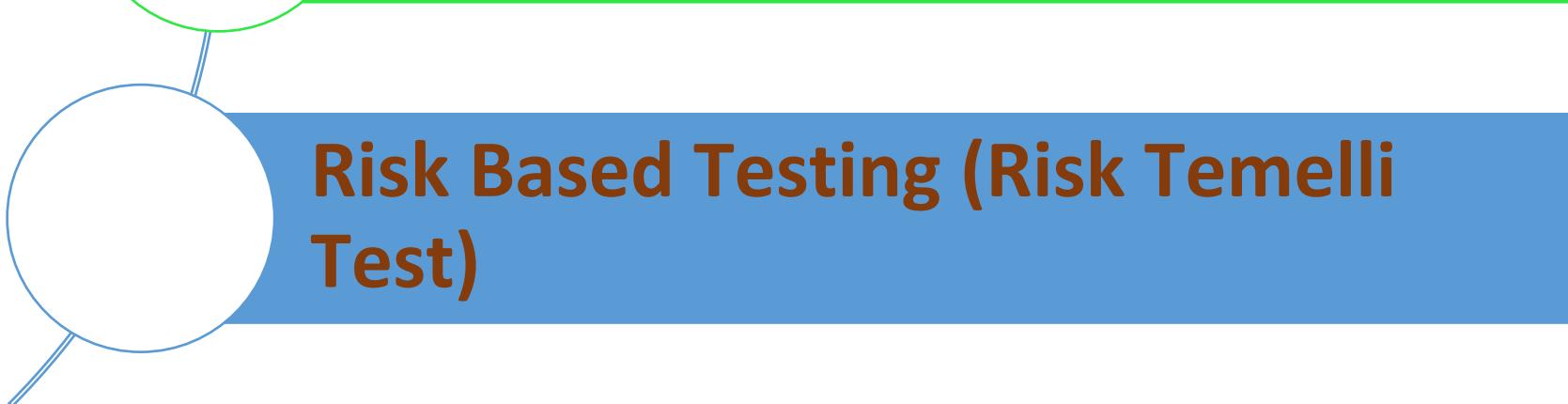
Test Yönetimi - Risk ve Test Etme

Risk Nedir?

meydana geldiğinde istenmeyen sonuçlara ya da potansiyel bir probleme yol açabilecek bir olay, tehlike, tehdit veya durumun olasılığı



Test Yönetimi - Risk ve Test Etme





Test Yönetimi - Risk ve Test Etme

Ürün Riskleri (Product Risks)

- Arızaya eğilimli yazılımın teslim edilmesi
- Yazılım/donanımın bir kişiye veya şirkete zarar verebilme potansiyeli
- Yazılım özelliklerinin zayıf olması (örnek: **fonksiyonalite, güvenilirlik, kullanılabilirlik ve performans**)
- Veri bütünlüğünün ve kalitesinin zayıf olması (örnek : veri geçişi sorunları, veri dönüştürme problemleri, veri taşıma problemleri, veri standartlarının ihlali)
- Amaçlanan fonksiyonlarını gerçekleştirmeyen yazılım

Yazılımdaki potansiyel arıza alanları (gelecekteki ters gidebilecek işler veya tehlikeler) ürünün kalitesini riske attığı için **ürün riskleri** olarak bilinir.



Test Yönetimi - Risk ve Test Etme

Ürün Riski (Product Risks)

- Riskler, testin nereden başlatılacağına ve daha fazla testin nerede yapılacağına karar vermek için kullanılır
- Test etme süreci istenmeyen sonuçların ortaya çıkma riskini azaltmak veya istenmeyen bir sonucun etkisini azaltmak için hayatı geçirilir
- Ürün riskleri, projenin başarısı ile ilgili bir risk çeşididir.
- Test etmede risk bazlı yaklaşım, bir projenin ilk aşamalarından başlayarak ürün riskinin seviyelerini azaltmak için proaktif fırsatlar sunar. Bu sayede risklerin en baştan tanımlanmasını ve tanımlanan bu riskler kullanılarak test planamasının, kontrolünün, analizinin, tasarımının, hazırlığının ve yürütülmesinin yapılmasını sağlar.





Test Yönetimi - Risk ve Test Etme

Proje Riskleri (Project Risks)

Organizasyonel faktörler:

- + Beceri, eğitim ve personel kısıtlaması
- + Personel sorunları
- + Politik sorunlar, örneğin:

-  Test uzmanlarının ihtiyaçlarını ve test sonuçlarını karşı tarafa iletmesi ile ilgili sorunlar
-  Test ve gözden geçirmelerde bulunan bilgilerin yazılım sürecinin iyileştirilmesi için kullanılamaması
-  Test ekibinden ve sürecinden yanlış bekłentiler (örnek: test sırasında hata bulmanın önemini takdir etmeme)

projenin hedeflerine ulaşmasını engelleyebilecek riskler **proje riskleri** olarak bilinir.



Test Yönetimi - Risk ve Test Etme

Proje Riskleri (Project Risks)

Teknik sorunlar:

- ✓ Doğru gereksinimleri belirleme ile ilgili problemler
- ✓ Gereksinimlerin kısıtları karşılayamaması
- ✓ Test ortamının zamanında hazır olmaması
- ✓ Geç kalınmış veri dönüştürme,
- ✓ Geç kalınmış geçiş planlaması,
- ✓ Geç kalınmış yazılım geliştirme,
- ✓ Veri dönüştürme/geçiş araçlarının geç test edilmesi,
- ✓ Tasarım ve kod kalitesinin düşük olması
- ✓ Yapılandırma ve test verisinin kalitesinin düşük olması





Test Yönetimi - Risk ve Test Etme

Proje Riskleri (Project Risks)

Tedarikçi sorunları:

- ✓ Tedarikçilerin kalitesiz iş üretmesi
- ✓ Sözleşme sorunları
- Riskleri analiz ederken, yönetirken ve azaltırken, **test yöneticisinin iyi oluşturulmuş proje yönetimi ilkelerini** izlemesi gereklidir.
- "Yazılım Testi Dokümantasyonu Standardı" (IEEE Std 829-1998)'nın test planlarıyla ilgili kısmında risklerin ve beklenmeyen olayların bildirilmesi gereği vurgulanır





Test Yönetimi - Risk ve Test Etme

Risk Based Testing - (Risk Temelli Test)

- Risk analizi yapılmalı
- ✓ Gereksinim tanım dökümanı
- ✓ Paydaşlarla durum üzerine düşünülmeli
- ✓ Konu uzmanlarından öneriler alınmalı
- Puanlandırılmalı
 - Farklı aralıklar kullanılabilir
 - ✓ 1-10 ✓ 1-5 ✓ 1-3
- Zaman planı ile riskler paylaşılmalıdır

Risk temelli testin en önemli özelliklerinde birisi **risk analizi yapılması**dır.

Risk analizi mevcut durum göz önüne alınarak hazırlanmış **test caselere verilen puanlardır**.

Risk analizi yapmak demek konu üzerine derinlemesine incelemek demektir.

Test caseler yazılırken kullanılan dökümanlar bu boyutta tekrar gözden geçirilebilir.



Test Yönetimi - Risk ve Test Etme

Risk bazlı yaklaşımın tanımlanan riskler şu amaçlarla kullanılabilir:

- ✓ Uygulanacak test tekniklerini belirleme
- ✓ Gerçekleştirilecek testin derecesini belirleme
- ✓ Önemli hataları mümkün olduğunca erken bulmak için testi **önceliklendirme**
- ✓ Riski azaltmak için test dışı aktivitelerin (örnek: yeni tasarımcılara eğitim sağlama) uygulanıp uygulanmayacağı belirleme





Test Yönetimi - Risk ve Test Etme

Risk Analiz Örneği

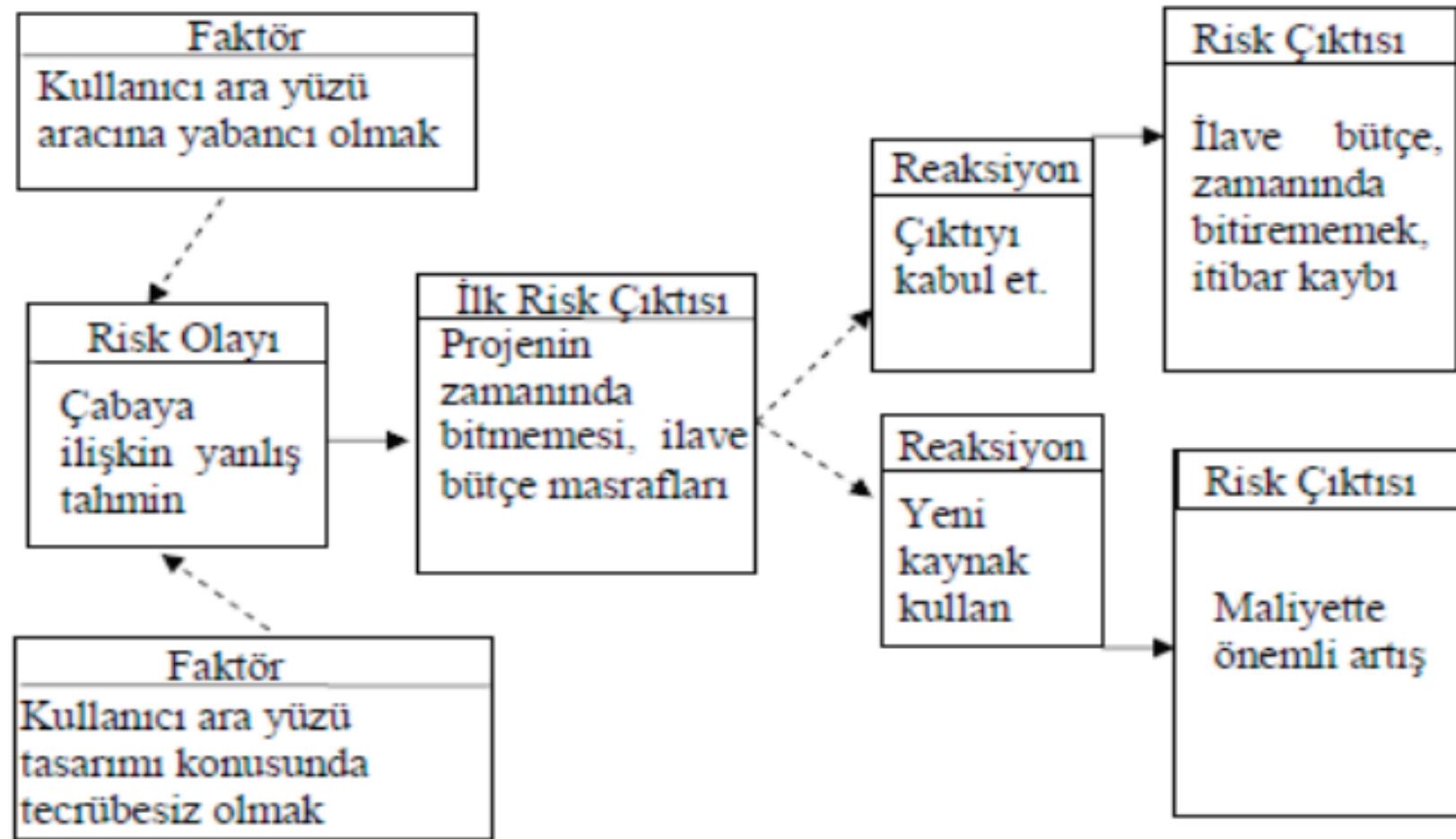
Gereksinim	Çıkma Olasılığı	Etkisi	Risk Değeri
Yeni Üyelik	3	4	$3 \times 4 = 12$
Satın alma	4	5	$4 \times 5 = 20$
Kullanıcı Bilgilerini Görme	1	2	$1 \times 2 = 2$

Risk Değeri = Çıkma Olasılığı x Etkisi

Not: 1 en küçük - 5 en büyük değere sahip



Örnek senaryo grafigi

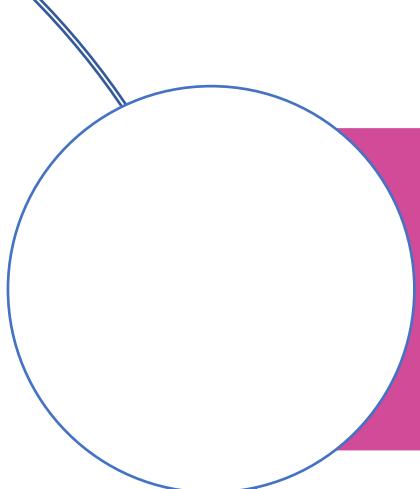




Olay/Vaka/Hata Yönetimi

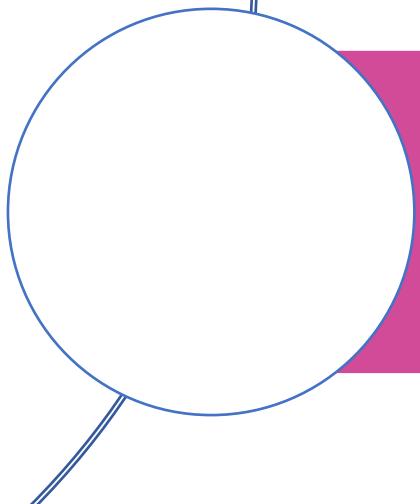


Test Yönetimi - Olay/Vaka/Hata Yönetimi



Incident Logging
(Vaka Kaydı)

Keywords



Root Cause
(Problemin Özü)



Test Yönetimi - Olay/Vaka/Hata Yönetimi

➤ Vaka Nedir?

Test sırasında beklenen sonuca uymayan her şeydir.

➤ Sebepleri nelerdir?

- ✓ Yazılımda hata var
- ✓ Test doğru işletilmedi, testte hata var
- ✓ Beklenen sonuç yanlış verilmiş
- ✓ Test ortamında sorun vardır



Hata yazılımda olabileceği gibi dökümantasyonda da olabilir!



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Vakalar ne aşamada ortaya çıkar ?

- ✓ yazılım geliştirme
 - ✓ gözden geçirme
 - ✓ test ve yazılımın kullanımı sırasında
 - ✓ kod veya gereksinim dokümanlarında
 - ✓ yazılım geliştirme dokümanlarında
 - ✓ test dokümanları ve kullanıcı bilgileri ("Yardım" veya kurulum kılavuzları) gibi dokümantasyon türlerinde

ortaya çıkabilir





Test Yönetimi - Olay/Vaka/Hata Yönetimi

- Testin hedeflerinden biri hataları bulmak olduğundan, gerçekleşen ve beklenen çıktılar arasındaki farklılıkların ilk başta **vaka- olay** olarak kaydedilmesi gereklidir.
- Bir vaka araştırıldığında olayın **hata olduğu** ortaya çıkabilir.
- Vakaları ve hataları ortadan kaldırmak için uygun eylemler belirlenmelidir.
- Vakalar ve hatalar bulunmasından, sınıflandırılmasına, düzeltilmesine ve çözüm onayına kadar izlenmelidir.
- Tüm olayların yönetilmesi için şirketlerin **vaka yönetimi süreci** ve sınıflandırma kuralları oluşturulması gereklidir.



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Vakaları neden raporlarız?

- ✓ Yazılımcılara ve diğer paydaşlara problem hakkında geri bildirim sağlayarak gerektiğinde **tanımlama, izolasyon ve düzeltme sağlama**
- ✓ Test liderlerine, test edilen **sistemin kalitesini ve testin ilerleyişini izleme yolu sunma**
- ✓ Test süreç iyileştirmesi için fikirler sunma





Test Yönetimi - Olay/Vaka/Hata Yönetimi



Vaka raporunun detayları nelerdir ?

- ✓ Yayın tarihi, yayinallyan kuruluş ve yazar
- ✓ Beklenen ve gerçekleşen sonuçlar
- ✓ Test öğesinin (yapılardırma ögesi) ve ortamın tanımlanması
- ✓ Olayın gözlemlendiği yazılım yaşam döngüsü süreci
- ✓ Kayıtlar, veri tabanı dökümleri ve ekran görüntüleri de dahil olmak üzere gerçekleşen olayın yeniden üretiminin ve çözümün sağlamak için olayın detaylandırılması
- ✓ Etkinin paydaşlar çıkarları üzerindeki etki kapsamı veya derecesi
- ✓ Etkinin yazılım üzerindeki önem derecesi



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Vaka raporunun detayları nelerdir ?

- ✓ Aciliyet / düzeltme önceliği
- ✓ Olayın durumu (açık, ertelenmiş, kapalı)
- ✓ Sonuçlar, öneriler ve onaylar
- ✓ Olay nedeniyle yapılan bir düzeltmeden/değişiklikten etkilenebilen diğer alanlar gibi genel sorunlar
- ✓ Olayın izole edilmesi
- ✓ Problemi ortaya çıkaran test senaryosu



Not : Olay raporu yapısının ana hatları "Yazılım Testi Dokümantasyonu Standardı" (IEEE Std 829-1998) kapsamında yer almaktadır.



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Test incident report

- > Vaka raporu ile
ilgili bir şablon

10. Test incident report

10.1 Purpose

To document any event that occurs during the testing process that requires investigation.

10.2 Outline

A test incident report shall have the following structure:

- a) Test incident report identifier;
- b) Summary;
- c) Incident description;
- d) Impact.

The sections shall be ordered in the specified sequence. Additional sections may be included at the end. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test incident report or available to users of the incident report.

Details on the content of each section are contained in the following subclauses.

10.2.1 Test incident report identifier

Specify the unique identifier assigned to this test incident report.

10.2.2 Summary

Summarize the incident. Identify the test items involved indicating their version/revision level. References to the appropriate test procedure specification, test case specification, and test log should be supplied.

10.2.3 Incident description

Provide a description of the incident. This description should include the following items:

- a) Inputs;
- b) Expected results;
- c) Actual results;
- d) Anomalies;
- e) Date and time;
- f) Procedure step;
- g) Environment;
- h) Attempts to repeat;
- i) Testers;
- j) Observers.



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Vaka Raporu ile ilgili Örnekler

Accident / Incident Report Form Template

EMPLOYEE NAME:	TITLE / ROLE:	DATE OF REPORT:	
EMPLOYEE SIGNATURE:	LENGTH OF TIME IN CURRENT ROLE:	DATE OF INCIDENT:	
LOCATION OF INCIDENT:	TIME OF INCIDENT:		
RESULT OF ACCIDENT / INCIDENT			
HEAD	SHOULDER	LEFT	RIGHT
FACE	ARM PIT		
NECK	UPPER ARM		
UPPER BACK	LOWER ARM		
LOWER BACK	ELBOW		
CHEST	WRIST		
ABDOMEN	HAND		
PELVIS / GROIN	BUTTOCKS		
LIPS	HIP		
TEETH	THIGH		
TONGUE	LOWER LEG		
NOSE	KNEE		
FINGERS	ANKLE		
TOES	EYES		
OTHER:	EARS		
INCIDENT INFORMATION			
INCIDENT DESCRIPTION			
TASKS LEADING TO INCIDENT			
ADDITIONAL INFO			
OSHA REPORTING			
WITNESS NAME AND CONTACT			
VERIFICATION			
SUPERVISOR NAME:	REPORTED TO:	DATE OF REPORT:	
SUPERVISOR SIGNATURE:	BUREAU:	WORK UNIT:	
ADDITIONAL INFO:			

Employee Incident Report Template

REPORTED BY:	DATE OF REPORT:
TITLE / ROLE:	INCIDENT NO.:
EMPLOYEE INCIDENT INFORMATION	
EMPLOYEE NAME:	EMPLOYEE TITLE / ROLE:
DATE OF INCIDENT:	TIME OF INCIDENT:
LOCATION:	
SPECIFIC AREA OF LOCATION:	
ADDITIONAL PERSON(S) INVOLVED:	
WITNESSES:	
INCIDENT DESCRIPTION INCLUDING ANY EVENTS LEADING TO OR IMMEDIATELY FOLLOWING THE INCIDENT	
EMPLOYEE EXPLANATION OF EVENTS / CIRCUMSTANCES	
RESULTING ACTION EXECUTED, PLANNED, OR RECOMMENDED	
EMPLOYEE NAME:	EMPLOYEE SIGNATURE:
REPORTING STAFF NAME:	REPORTING STAFF SIGNATURE:
HR REP NAME:	HR REP SIGNATURE:



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Hata Önem Seviyeleri

Büyük	Kritik	Ölümcul	Orta	Küçük	Kozmetik
<ul style="list-style-type: none">• Testler devam edebilir.• Ürün bu hata ile teslim edilebilir.• Ancak yazılım kullanıldığında telfisi zor sonuçlar doğurabilir.	<ul style="list-style-type: none">• Testler devam edebilir.• Ancak bu hata derecesi ile yazılım teslim edilemez.	<ul style="list-style-type: none">• Testlerin devam etmesini engelleyecek hataları belirtmek için kullanılan derecedir.• Eğer bu türden bir hata bulunmuş ile testlerin devam etmesi imkânsızdır.	<ul style="list-style-type: none">• Testler devam edebilir.• Ürün bu hata ile teslim edilebilir.• Ancak yazılım kullanıldığında telfisi mümkün sorunlar çıkartabilir.	<ul style="list-style-type: none">• Testler devam edebilir.• Ürün bu hata ile teslim de edilebilir.• Yazılımın işleyişinde ortaya çıkabilecek olan hatalar önemli bir sonuç doğurmaz	<ul style="list-style-type: none">• Yazılım üzerindeki renk, font, büyülüklük gibi görsel hatalardır.• Olması durumunda ne testi durdurur ne de ürünün teslimini engeller



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Türkçesi **vaka yönetimi** olarak çevrilen **incident management**

Defect - tracking

Defect - management

Bug - tracking

Bug – management tool



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Vaka Yönetim Süreci

Vaka tespit edilip bildirilir

Düzelme faaliyetleri gerçekleştirilir

Yeni yazılım yükü hazırlanır

Yineleme testleri gerçekleştirilir

Vaka yeni yükte düzeltmiş ise kapatılır

INCIDENT MANAGEMENT



INCIDENT



PROCESS



DETECTION



ANALYSIS



INITIAL SUPPORT



RESTORE

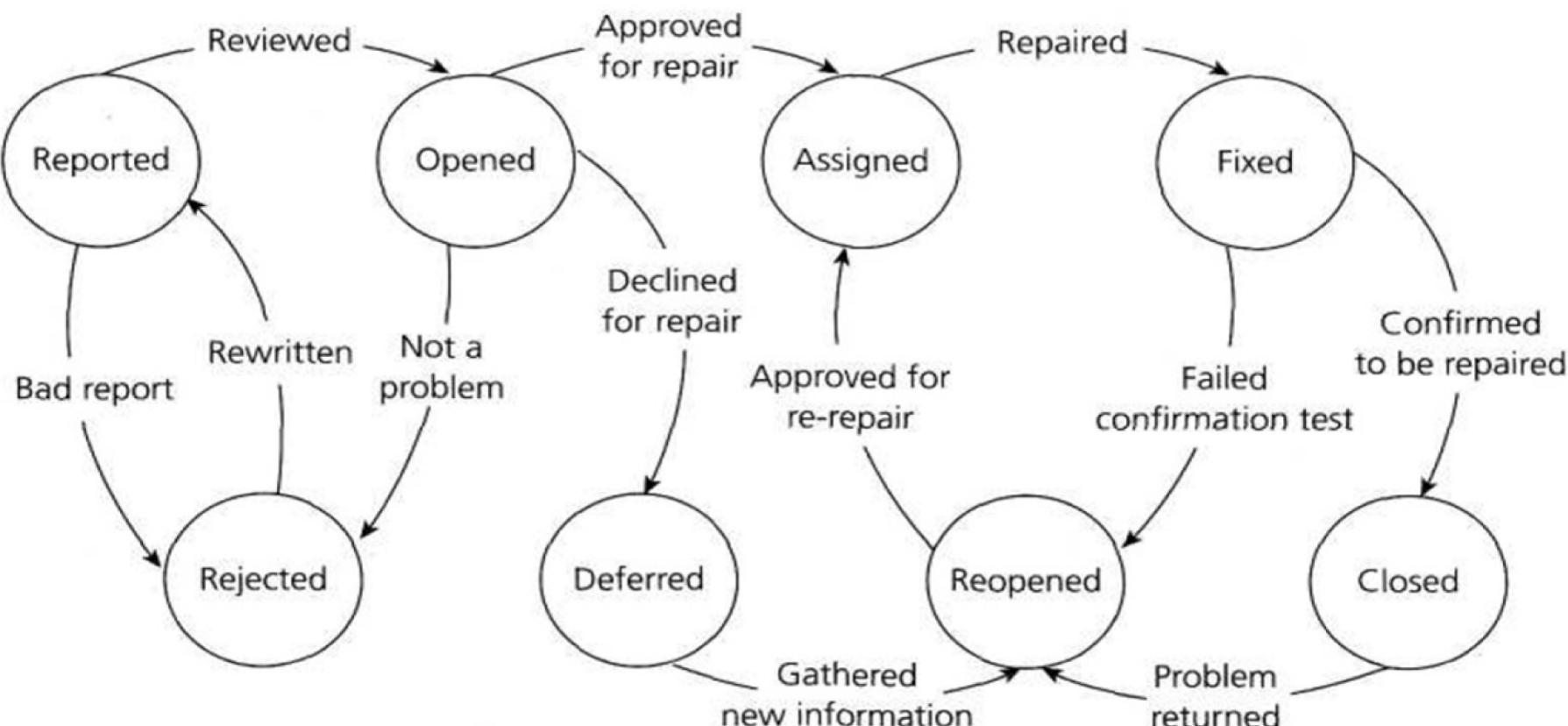


REPORTING



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Vaka Yaşam Döngüsü

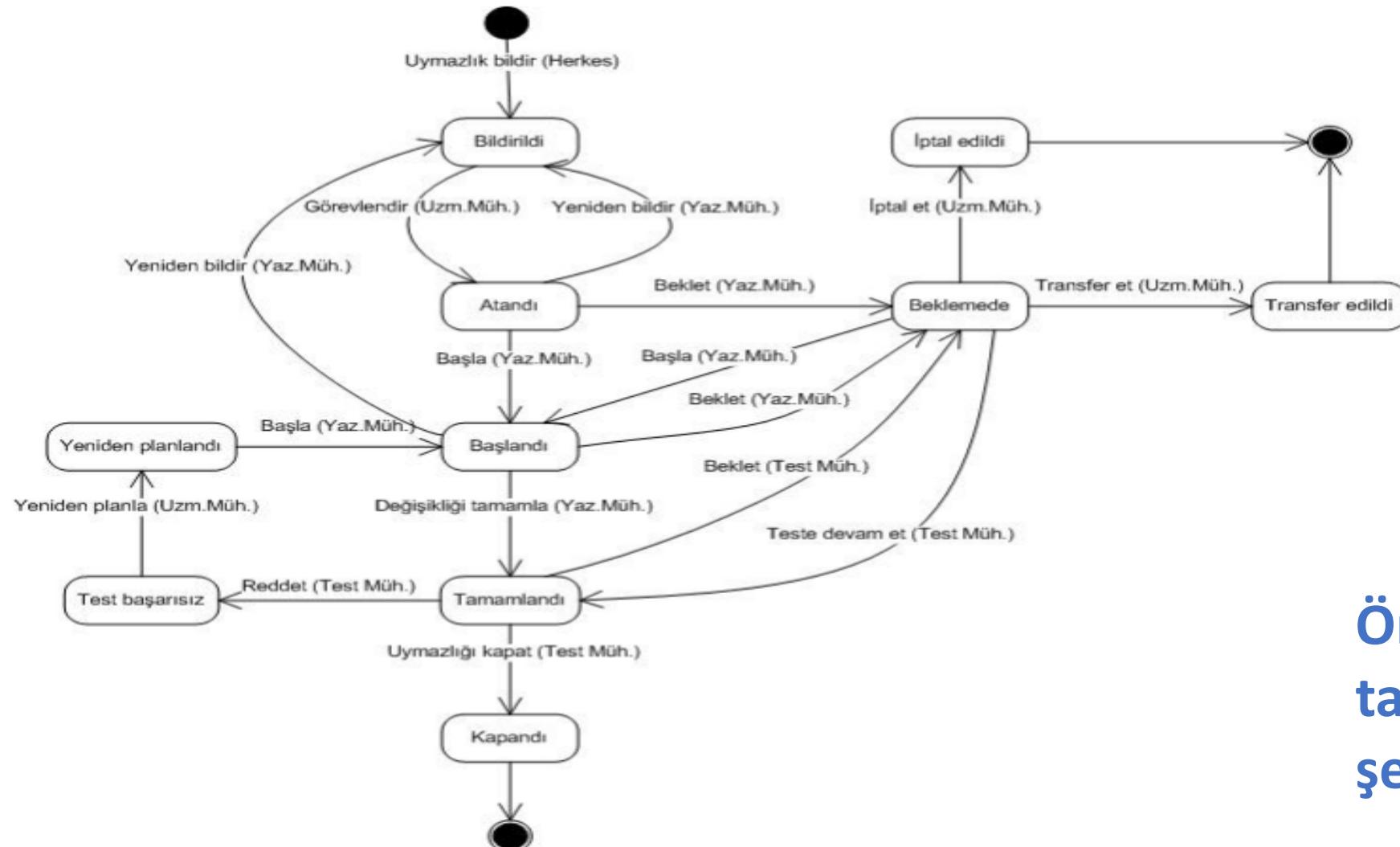


Hata statüleri

Reported
Opened
Assigned
Fixed
Closed
Reopened
Deffered
Rejected



Test Yönetimi - Olay/Vaka/Hata Yönetimi



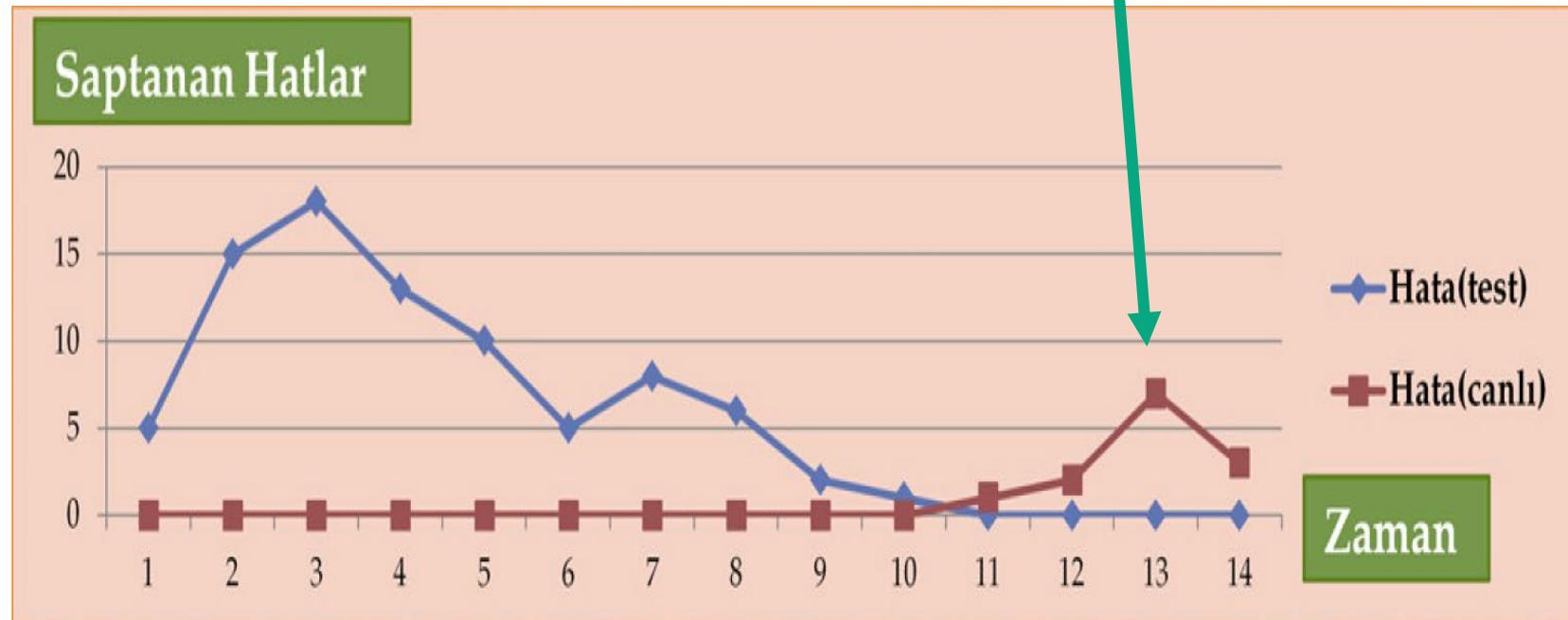
Örnek bir vaka takip akış şeması



Test Yönetimi - Olay/Vaka/Hata Yönetimi

Hata Yakalama Oranı (Defect Detection Percentage)

- ✓ Test sırasında yakalanana hatanın tüm hatalara oranıdır..
- ✓ Test ortamında gözden kaçan hataları da gösterir



$$DDP = \frac{hata(test)}{hata(test) + hata(canlı)}$$



Test Araçları Desteği



Test Aracı Çeşitleri



Test Aracı Çeşitleri

Keywords





Test Aracı Çeşitleri

Test için Araç Desteği Nerelerde Kullanılır ?

- Test planlama, test tasarıımı, test raporlama ve gözlemeleme gibi manuel test işlemlerini destekleyerek test işlemlerinin verimliliğini artırma
- Manuel olarak yapıldığında önemli oranda kaynak gerektiren işlemleri otomasyona geçirme (örnek: statik test)
- Manuel olarak yürütülemeyen işlemleri otomasyona geçirme (örnek: istemci-sunucu uygulamalarının geniş ölçekli performans testleri)
- Testin güvenilirliğini artırma (örnek: veri karşılaştırmalarının otomasyona geçirilmesi veya davranışın simülasyonu)



Test Aracı Çeşitleri

Test Aracı Sınıflandırılması

Araçlar çeşitli kriterlere göre sınıflandırılabilir:

- ✓ amaç, lisanslı
- ✓ ücretsiz
- ✓ açık kaynak kodlu
- ✓ paylaşımı
- ✓ kullanılan teknoloji



Test Aracı Çeşitleri

Test Aracı Sınıflandırılması

- Bazı araçlar yalnızca bir aktiviteyi desteklerken, diğerleri **birden fazla aktiviteyi destekleyebilir**, ancak **en yakından ilişkili oldukları aktiviteler altında sınıflandırılır**.

- Tek bir sağlayıcıdan gelen araçlar, özellikle de **birlikte çalışmak için tasarlanmış olanlar tek bir grupta toplanabilir**.

Probe Effect (Ölçücü Etkisi): Test amacıyla kullanılan bir araç test edilen biriminin üzerine bir etki bırakır. Gerçek davranış ile test davranışları arasında fark yaratan bu duruma, test sonuçlarına yansyan bu etkiye **ölçücü (probe) etkisi** denir.





Test Aracı Çeşitleri

Test Yönetimi ve Testler için Araç Desteği

Yönetim araçları tüm yazılım yaşam döngüsü boyunca tüm test aktivitelerine uygulanır

- ✓ Test Yönetim Araçları
- ✓ Gereksinim Yönetim Araçları
- ✓ Vaka-Olay Yönetim Araçları (Hata Takip Araçları)
- ✓ Yapılandırma Yönetim Araçları





Test Aracı Çeşitleri - Test Yönetimi ve Testler için Araç Desteği

Test Yönetim Araclarının Özellikleri

Test Yönetimi

- * Test Verisi Takibi
- * Test Koşturulacağı Ortamlar
- * Planlanan, Yazılan, Koşturulan, Başarılı, Başarsız Test Case Sayısı

Çalıştırılacak Testlerin Planlanması

- * Manuel * Test Araçlarıyla

Test Aktivitelerinin Yönetimi

- * Harcanan Zaman * Plana Uyumluluk

testlerin işleyışı ile ilgili genel
ve detay seviyede bilgiler
veren araçlardır

Test manager



Test Aracı Çeşitleri - Test Yönetimi ve Testler için Araç Desteği

Test Yönetim Araçlarının Özellikleri

Diger Test Araclarina Baglanti

- * Test Execution Araçları
- * Vaka Yönetimi
- * Gereksinim Yönetimi
- * Konfigürasyon Yönetimi

Izlenebilirlik

- * Test Sonuçları * Hatalar



Test Sonuçlarına Erisim

Raporlama

- * Başarılı / Başarısız Test Run * Çözülmüş / Bekleyen / Kapatılan Hataların Oranı



Test Aracı Çeşitleri - Test Yönetimi ve Testler için Araç Desteği

Gereksinim Yönetimi Araclarının Özellikleri

- ✓ Gereksinimleri depolar
- ✓ Gereksinimlerin özelliklerini ile bilgiler
- ✓ Gereksinimlerin tutarlığını kontrol eder
- ✓ Önceliklendirme
- ✓ Durumlarını gösterme
- ✓ İzleneme
- ✓ Test yönetim araçlarıyla entegrasyon
- ✓ Gereksinimlerin kapsamını (coverage) gösterme

Gereksinim yönetim araçları yazılı gereksinimleri depolayarak gerekli hallerde durumlarına, özelliklerine, öncelik (priority) değerine göre çeşitli filtrelemeler yapan test aracıdır. Gereksinim yönetim toollarının genel amacı gereksinimleri toplamak ve test case hazırlamasına yardımcı olmaktadır.

Business Analyst



Test Aracı Çeşitleri - Test Yönetimi ve Testler için Araç Desteği

Vaka Yönetim Araçlarının Özellikleri

Vakanın özelliklerini kayıt eder

Dosya ekleme olanağı

Hataları önceliklendirme

Atama özelliği

Durum

* Open

* Rejected

* Duplicate

Raporlama

Saptanan her türlü vakanın kaydedilmesine, takip edilmesine ve raporlanması olanak sağlamaktır.

Vaka yönetimi araçları vasıtasıyla saptanan hatalar çözülmek üzere ilgili kişilere atama yapmaya olanak sağlamalıdır

Tester



Test Aracı Çeşitleri - Test Yönetimi ve Testler için Araç Desteği

Konfigürasyon Yönetimi Araçlarının Özellikler

Testware ve software sürüm kontrolü

Farklı sürümler arasında izlenebilirlik

Sürüm ve konfigürasyon bağlantısını kurmak

Sürüm ve sürüm adayı (release candidate) yönetimi

Baselining (özel sürümler üretme)

Yazılıma müdahaleleri kontrol eder

* Check in * Check out * Merge * Push istekleri

Test açısından bakıldığında konfigürasyon yönetimi tüm test araçlarının (testware) ve test edilen yazılım ürünlerinin (software) sürümlerini yönetmeye yarayan sistem araçlarıdır.

Developer &
Tester



Test Aracı Çeşitleri

Statik Test için Araç Desteği

Statik test araçları, geliştirme sürecinde erken safhalarda daha fazla hata bulmanın uygun maliyetli bir yolunu sunar

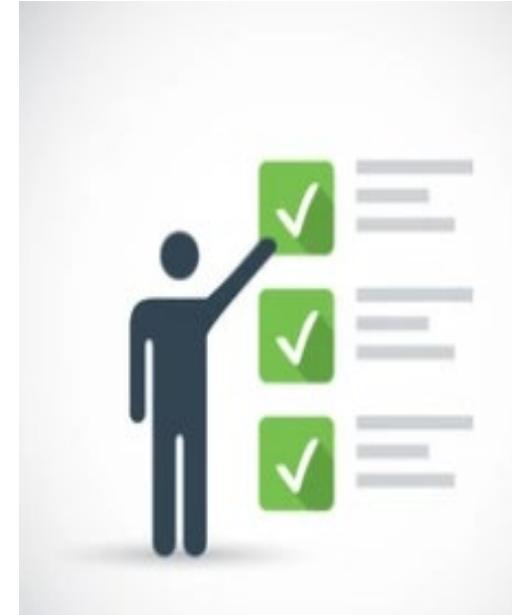
- ✓ Gözden Geçirme Araçları
- ✓ Statik Analiz Araçları
- ✓ Modelleme Araçları



Test Aracı Çeşitleri - Statik Test için Araç Desteği

Gözden Geçirme Araçları

- ✓ Bu araçlar, gözden geçirme süreçlerinde, kontrol listelerinde, gözden geçirme kılavuzlarında destek sağlar ve gözden geçirme yorumlarını saklamak ve iletmek, ayrıca hataları ve eforu rapor etmek için kullanılır.
- ✓ Büyük veya coğrafi olarak farklı alanlarda bulunan ekipler için gözden geçirme süreçlerine yardımcı olur.

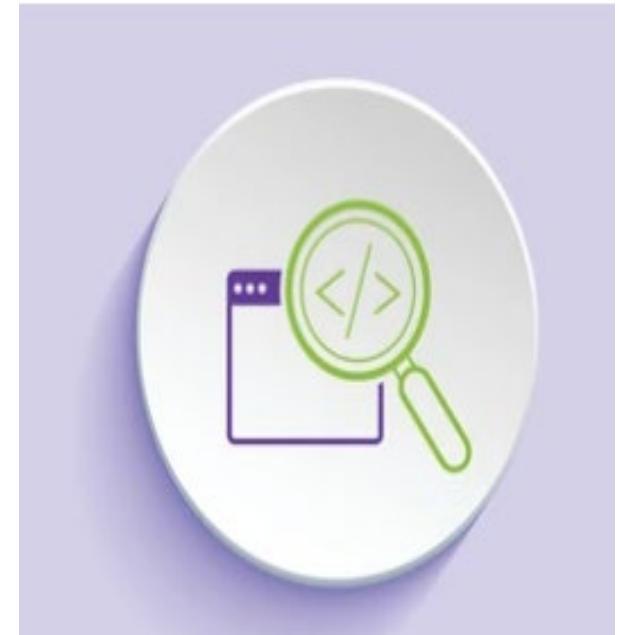




Test Aracı Çeşitleri - Statik Test için Araç Desteği

Statik Analiz Araçları

- ✓ Bu araçlar **kodlama standartlarını** (güvenli kodlama dahil), **İç yapı mimarisinin analizini** sağlayarak dinamik testten önce yazılımcıların ve test uzmanlarının **hataları** bulmasını sağlar.
- ✓ Koda yönelik metrikler (örnek : karmaşıklık) sağlayarak planlamada ve risk analizinde de yardımcı olurlar.





Test Aracı Çeşitleri - Statik Test için Araç Desteği

Modelleme Araçları

- ✓ Bu araçlar, tutarsızlıklarını belirleyip hataları bularak yazılım modellerini (örnek : ilişkisel veri tabanı için fiziksel veri modeli PDM) doğrulamak için kullanılır.
- ✓ Bu araçlar, modele bağlı olarak test senaryoları üretmede kullanılabilir.





Test Aracı Çeşitleri –Test Yürütmeye ve Kayıt Tutma İçin Araç Desteği

Test Yürütmeye Araçları

testlerin otomatik veya yarı otomatik olarak yürütülmesini sağlar ve genellikle her bir test koşumu için bir test kaydı tutar, ayrıca testleri de kaydeder



Test Kuluçkası / Birim Testi Çerçevesi Araçları

Birim testi kuluçkası veya çerçevesi, sahte nesnelerin taklit uygulama veya sürücü olarak sağlanması yoluyla test nesnesinin çalışacağı ortamı simüle ederek bileşenlerin ya da sistem bölümlerinin test edilmesini kolaylaştırır



Test Karşılaştırıcıları

dosyalar, veri tabanları veya test sonuçları arasındaki farkları belirler, bir test karşılaştırıcı, özellikle otomasyonda bir test sonucunu bilen olarak kullanabilir



Kapsam Ölçüm Araçları

koda dahil olarak veya olmayarak, bir dizi testle çalıştırılmış kodun yüzdesini ölçer, örnek :komutlar, dallar veya kararlar ve bir modül ya da fonksiyon çağrıları



Güvenlik Test Araçları

yazılımın güvenlik özelliklerini değerlendirmek için kullanılır, güvenlik araçları çoğunlukla belirli bir teknolojiye, platforma ve amaca odaklanır



Test Aracı Çeşitleri –Performans-Monitörleme için Araç Desteği

✓ Dinamik Analiz Araçları

Dinamik analiz araçları yalnızca yazılım yürütülürken görülebilen zamana (performans gibi) veya bellek sizıntıları gibi hataları bulur. Genellikle bileşen ve bileşen entegrasyon testinde kullanılır.

Performans testi araçları, eş zamanlı kullanıcı sayısı, artış deseni, sıklık ve ilgili işlem yüzdesi bağlamında bir yazılımın simüle edilen kullanım koşulları altında nasıl davranışacağını monitörler ve raporlar.

✓ Performans Testi / Yük Testi / Stres Testi Araçları

✓ İzleme Araçları

İzleme araçları sürekli olarak belirli sistem kaynaklarının kullanımını analiz eder, doğrular ve raporlar, ayrıca olası hizmet problemleri ile ilgili uyarılar verir.



Test Aracı Çeşitleri –Özel Test İhtiyaçları için Araç Desteği



Veri Kalitesi Değerlendirmesi

Veri, veri dönüştürme/geçiş projeleri ve veri ambarı benzeri uygulamalar **ve projelerin merkezinde** bulunur. Bu tür projelerde, işlenen verilerin doğru ve eksiksiz olduğundan ve proje ihtiyaçlarına uyduğundan emin olmak amacıyla **veri dönüştürme ve geçiş kurallarını gözden geçirmek ve doğrulamak üzere araçların veri kalitesi değerlendirmesi** için kullanılması gereklidir.



Kullanılabilirlik testleri



Araçların Etkin Kullanımı: Potansiyel Avantajlar ve Riskler



Araçların Etkin Kullanımı- Test Araçlarının Olası Faydaları

Test araçlarının sıkıcı olan ve manuel işletilmesi çok zaman alan, hata riski fazla olan veya manuel işletilmesine olanak olmayan işler için sağlayacağı avantajlar vardır.

- Yinelenen işlerde azalma
- Daha fazla tutarlılık
- Tekrar çalışabilir işler
- Nesnel değerlendirme
- Testle ilgili bilgilere kolay erişim





Araçların Etkin Kullanımı- Test Araçlarının Olası Riskleri

- Gerçekçi olmayan bekentiler
- Aracın ilk kullanımına fazla zaman harcamak
- Araç kullanımının devamlılığı için fazla zaman harcamak
- Araçlara fazla güvenmek
- Test araçlarının ürettiği hedefler için fazla zaman harcamak



Araçların Etkin Kullanımı- Test Araçlarının İlk Kez Kullanımı

Test aracı seçimi ve önemli noktalar

- ✓ Organizasyon değişime hazırlığı
- ✓ Aracın organizasyonda geliştireceği süreçler
- ✓ Aracın tarafsız değerlendirilmesi
- ✓ Aracın beklenilere cevabı
- ✓ Tedarikçinin kalitesi
 - *Eğitim *Destek *İtibar *Kullanıcı forum/yorumlan (açık kaynaklar için)
- ✓ İlk kullanım
 - *Destek *Sıkıntılı noktalar *Montör



Etik Kurallar



Etik Kurallar

- Yazılım testine dahil olmak, test uzmanının müşterilerin gizli ve özel bilgilerine ulaşmasını sağlayabilir.

- Bu bilgilerin uygunsuz bir şekilde kullanılmaması için etik kurallarına uyulması gereklidir.





Etik Kurallar

- KAMU** - Test uzmanları, **kamu yararını gözeterek** hareket etmelidir
- MÜŞTERİ VE İŞVEREN** - Test uzmanları, kamu yararını gözeterek, **müşteri ve işverenlerinin çıkarlarına en uygun** şekilde hareket etmelidir
- ÜRÜN** - Test uzmanları, sağladıkları çıktıların mümkün olan **en yüksek profesyonel standartları karşıladığından emin** olmalıdır
- KARAR** - Test uzmanları, profesyonel kararlarında **tutarlılıklarını ve bağımsızlıklarını** korumalıdır





Etik Kurallar

- YÖNETİM** - Test yöneticileri ve liderleri, **test projelerinin yönetiminde** etik bir yaklaşım benimsemeli ve uygulamalıdır
- MESLEK** - Test uzmanları, kamu yararını gözterek **mesliğin dürüstüğünü ve itibarını** korumalıdır
- MESLEKTAŞLAR** - Test uzmanları, meslektaşlarına karşı **adil olmalı, onları desteklemeli** ve yazılımcılarla işbirliklerini geliştirmelidir
- SÜREKLİ GELİŞİM** - Test uzmanları, mesleklerinin içrasıyla ilgili **yaşam boyu öğrenmeli** ve bu konuda etik bir yaklaşım benimsemelidir





Etik Kurallar



Eğer bu kurallar test uzmanları tarafından ihlal edilirse..

ISTQB tarafından..

- ✓ mesleki anlamada cezai işlemlere maruz kalabilirler
- ✓ mesleki yaptırımlar uygulanabilir



Kaynakça

The Standish Group, "CHAOS Manifesto 2013: Think Big, Act Small," 2013.

<http://www.turkishtestingboard.org/>

http://www.emo.org.tr/ekler/f966d9ba0bf9787_ek.pdf

<https://www.profelis.com.tr/false-positive-true-negative-neler-oluyor/>

https://www.tutorialspoint.com/software_testing_dictionary/review.htm

<https://ahmeteminege1.medium.com/black-box-ve-white-box-test-c3850bd8f4d9>

https://www.emo.org.tr/ekler/adf71fe88613d84_ek.pdf

<https://www.mobilhanem.com/beyaz-kutu-test-teknikleri-ve-deneyim-temelli-test-teknikleri/>

http://ieee.metu.edu/iffet/blog/Konfigurasyon_Yonetimi_CMII-V1.2.pdf

<https://standards.ieee.org/ieee/829/1218/>

Kaynak: <http://www.slideshare.net/onsoftwaretest/istqb-iseb-lecture-notes-2-presentation>

<https://dergipark.org.tr/tr/download/article-file/165606>

<https://www.smartsheet.com/free-incident-report-templates>

https://www.emo.org.tr/ekler/ec563ab2a2c9be4_ek.pdf

Gürbüz, Ali, Yazılım Test Mühendisinin El Kitabı ISTQB Temel Seviye Test – ISTQB İleri Seviye – ISTQB Test Yöneticiliği – Yazılım Test Mühendisliği, Seçkin Yayıncılık, Şubat 2020

Bu
kaynaklardan
çok istifade
ettim çok
teşekkür
ederim 😊