



**Richmond**  
College

Your Access To Future  
With Advanced Ed.

**ISTQB**  
**31/07/2023**  
**Ders -1**

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

 **Richmond College**  
Advanced Education

Cooperation with  
**TECHPRO**  
EDUCATION



# ISTQB

## Foundation Level



Affiliated with Advanced education

# ISTQB Sertifikasının Önemi?

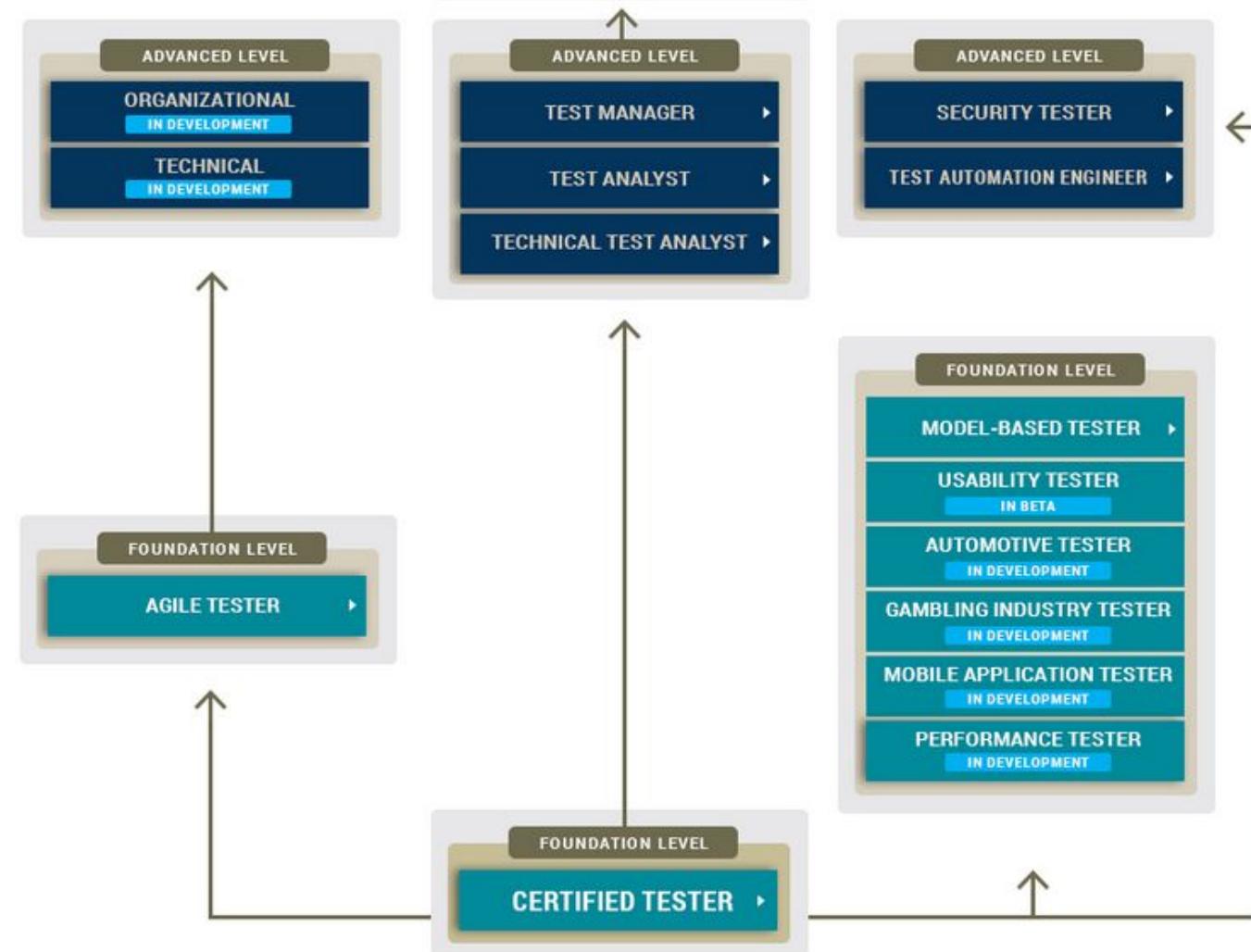
- **ISTQB** uluslararası geçerliliği olan test mühendisliği alanındaki en **önemli** sertifikasyon programıdır.
- **Yazılım testi** sertifikasyonu alanında **uluslararası standarttır**.
- İster test etme kariyerine yeni başlıyor olun, ister birkaç yıldır bu alanda çalışıyor olun, bir **ISTQB sertifikası kazanmak** önemlidir.



## ISTQB NEDİR ?

- International Software Testing Qualifications Board
- 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.





- Sınavda 40 tane çoktan seçmeli soru var.
- Sınav süresi: 1 Saat
- Sertifika almaya hak kazanmak için 40 sorudan min. 26 doğru cevap gerekmektedir.
- Güncel bilgileri resmi web sitesinden takip edebilirsiniz

<https://www.turkishtestingboard.org>

# Foundation Level içeriği

## ISTQB® - FOUNDATION LEVEL

Fundamentals of Testing	Testing Throughout the Software Development Lifecycle	Static Testing	Test Techniques	Test Management	Tool Support for Testing
What is Testing?	Software Development Lifecycle Models	Static Testing Basics	Categories of Test Techniques	Test Organisation	Test Tool Considerations
Why is Testing Necessary?	Test Levels	Review Process	Black-box Test Techniques	Test Planning and Estimation	Effective Use of Tools
Seven Testing Principles	Test Types		White-box Test Techniques	Test Monitoring and Control	
Test Process	Maintenance Testing		Experience-based Test Techniques	Configuration Management	
The Psychology of Testing				Risk and Testing	
				Defect Management	

# 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 ürünü**.

## YAZILIM HATASI



Therac-25 cihazı, kanser hastalarının tedavisinde kullanılmak için tasarlanmıştı.

yazılımında bulunan hata (bug) yüzünden yaklaşık 5 hastanın ağır dozda radyasyon sebebiyle hayatını kaybettiği raporlandı.

## YAZILIM TESTİNİN AMACLARI NELERDIR?

- Geliştirilen bir yazılımda hata olmadığını göstermek için değil, hataların varlığını göstermek ve bu hataları bulmak amacıyla gerçekleştirilir.
- Bu nedenle ne kadar çok hata tespit edilmiş ve bunlar düzelttilerek yazılım daha olgun hale getirilmiş ise test eylemleri o derece başarılı olmuş demektir.



# YAZILIM TESTİ



**Yazılım testi bir süreçtir.**

- Yazılım test süreci önce planlanan sonra icra edilip sonuçları kayıt altına alınarak belgelendirilen bir dizi eylemden oluşur.
- Ve Bu süreçteki faaliyetlerimiz geliştirilen yazılımdaki hataların varlığına odaklanır

# Testin Hedefleri



# Testing & Debugging & Confirmation Testing



## Test Neden Gerekli?

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



## Test Neden Gerekli?

- Gereksinim incelemelerine dahil olan test uzmanları bu iş ürünlerindeki hataları tespit edebilir.
- Gereksinim kusurlarının tanımlanması ve ortadan kaldırılması, yanlış veya test edilemeyen işlevsellik geliştirme riskini azaltır



# Test Neden Gerekli?

- Yazılım tasarılanırken test uzmanlarının tasarımcılarla yakın bir şekilde çalışması, her bir tarafın tasarım ve sistemin nasıl test edileceği konusundaki anlayışını artırabilir.



## Test Neden Gerekli?

- Test uzmanlarının, kod geliştirme aşamasındayken developerlar ile yakın bir şekilde çalışmasını sağlamak, kod ve testlerdeki hata riskini azaltabilir



# Test Neden Gerekli?



Yazılımın güvenilirliğini kontrol etmek için yazılım testi gereklidir. Yazılım testi olmadan yazılım istenen kaliteye ulaşamaz



Yazılım testi, sistemin arızaya neden olabilecek herhangi bir hata içermemesini sağlar.



Yazılım testi, ürünün müşterinin gereksinimine uygun olmasını sağlar

## Quality Management > Quality Assurance > Quality Control



Kalite yönetimi, bir kuruluş kalite ile ilgili olarak yönlendiren ve kontrol eden tüm faaliyetleri içerir.

# Quality Management > Quality Assurance > Quality Control

## Kalite Guvencesi;

Tüm yazılım geliştirme sürecini içerir süreçin izlenmesi ve iyileştirilmesi, üzerinde anlaşılan süreçlerin, standartların ve prosedürlerin takip edilmesini, sorunların bulunmasını ve ele alınmasını sağlar.



# Quality Management > Quality Assurance > Quality Control

**Kalite Kontrol;**  
uygun kalite seviyelerine  
ulaşılmasını destekleyen test  
faaliyetleri de dahil olmak üzere  
çeşitli faaliyetleri içerir.

Çeşitli test teknikleri  
gizli kalmayı da test tekniklerini  
test növüne göre sınımlı tutar.  
Bu tekniklerin  
sayısı sınırsızdır.



# Error-Defect-Failure



## Yanlış/Hata (Error)

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



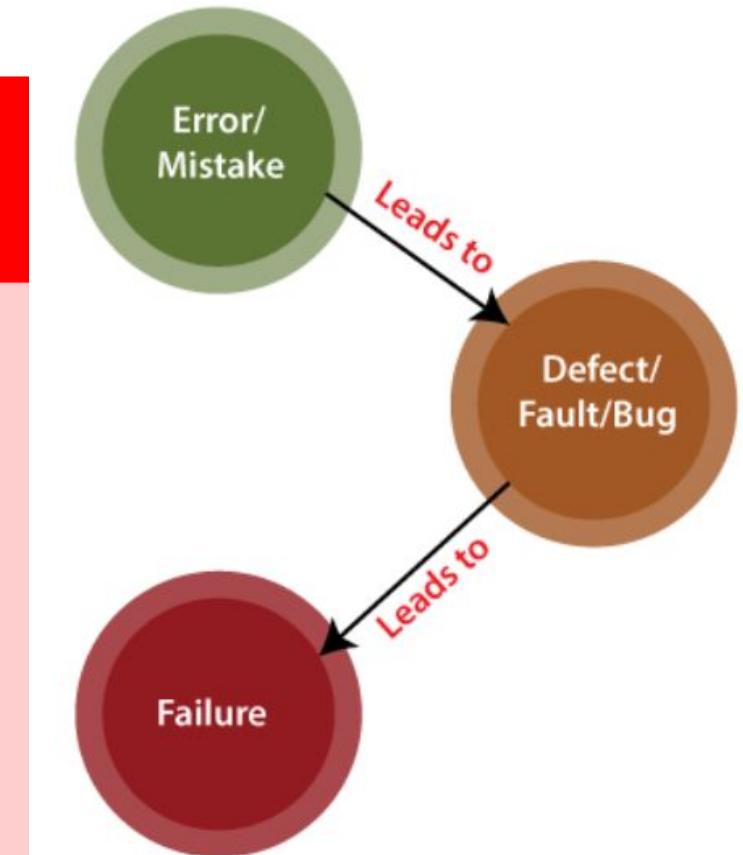
## Kusur/Hata (Fault-Defect-Bug)

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



## Arıza (Failure)

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

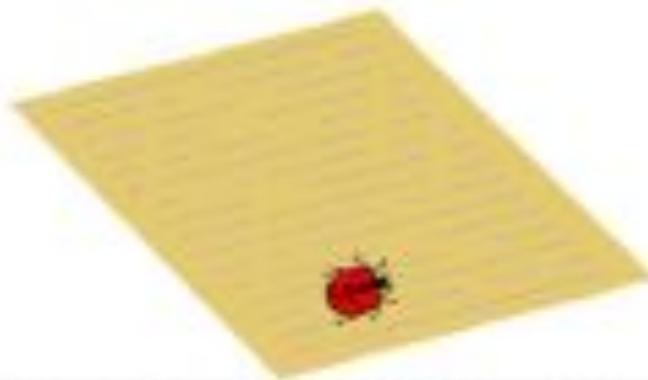


## Error-Defect-Failure

A person makes  
an error ...



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

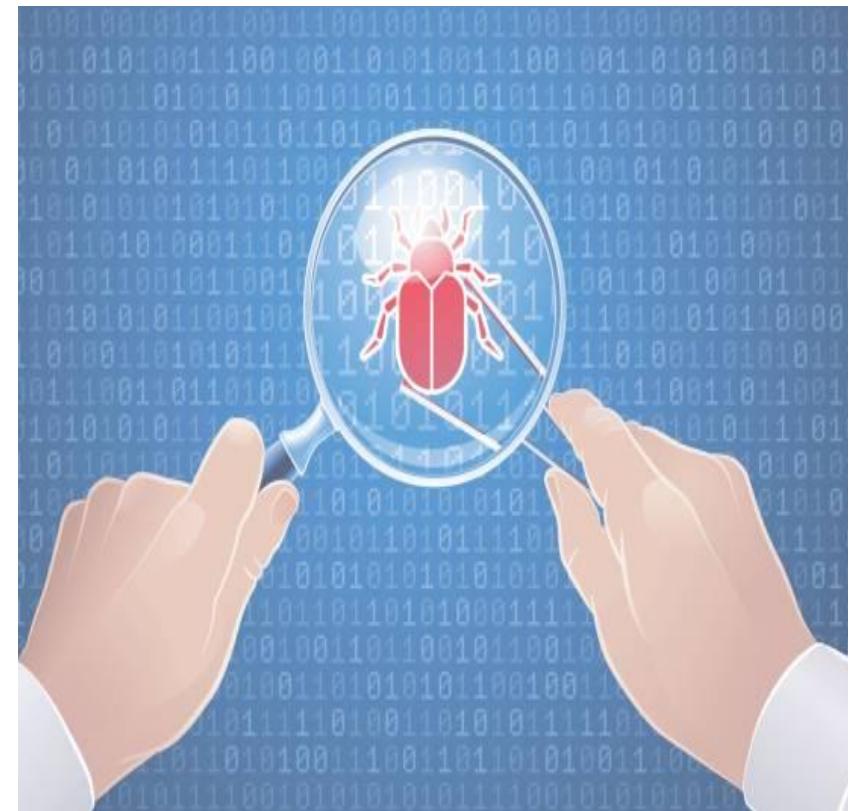


... that can cause  
a failure  
in operation



## Hatalar neden olur?

- Zaman baskısı
- İnsan hataları
- Deneyimsiz veya yetersiz vasıflı proje çalışanları
- Gereksinimler ve tasarım hakkında yanlış bilgilendirme dahil olmak üzere proje çalışanlar arasındaki iletişimsızlık
- Kodun, tasarımin, 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
- Yeni, tam bilinmeyen teknolojiler



# False positive & False negative

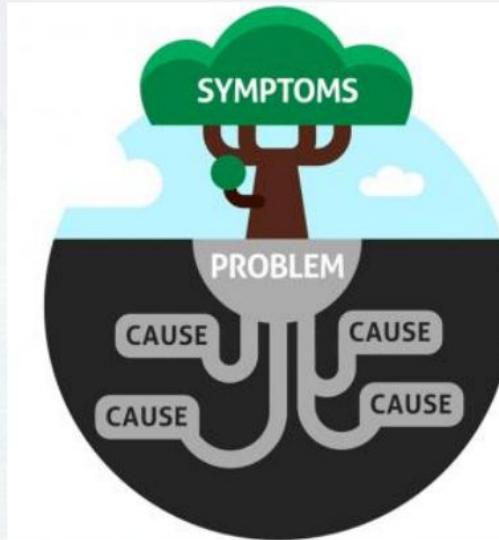


- ✓ Beklenmeyen(unexpected) test sonuçlarının tümü başarısızlık(failure) değildir.
- ✓ Testlerimizin görevi hatayı bulmak olsa da bazen testler hatalı bir ürünündeki hatayı tespit edemeyip, hatasız ürün sonucu verebilir(**false negatives**).
- ✓ Bu durum testlerin çalıştırılma yöntemlerindeki hatalardan veya test verilerindeki, test ortamındaki veya diğer test yazılımındaki kusurlardan veya başka nedenlerden kaynaklanabilir.
- ✓ Bazen de bunun tersi bir durumda hatasız olan bir ürün hatalı olarak tanımlanabilir. (**False positives**).

# Kok Neden Analizi

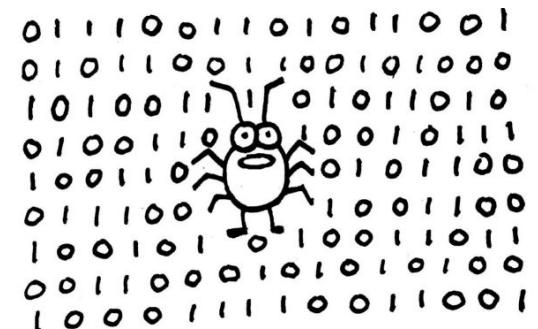
Kusurların temel nedenleri, kusurların oluşmasına katkıda bulunan en eski eylemler veya koşullardır.

## Root Cause Analysis



## 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.





### 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.



## 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.



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

- ✓ Bir uygulama içerisindeki az sayıda modül release(sürüm öncesi testler sırasında keşfedilen **kusurlarınlığını içerir** veya operasyonel arızaların çoğunundan sorumludur.
- ✓ Öngörülen hata kümeleri ve test veya operasyonda gerçek gözlemlenen kusur kümeleri, test çabasına odaklanmak için kullanılan bir risk analizine önemli bir girdidir



## 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.



### 6. Testing, koşullara(context) bağlılıdır

- ✓ 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.



### 7. Hataların olmaması bir yanlışıdır (hata yokluğu yanlışısı)

- ✓ 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).





**Richmond**  
**College**  
Your Access To Future  
With Advanced Ed.

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

# Test Süreci

 **Richmond College**  
Advanced Education

Cooperation with  
**TECHPRO**  
EDUCATION

# Test Süreci



# Test Planlama



- Tüm planlar bu aşamada yapılır
- 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



- 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

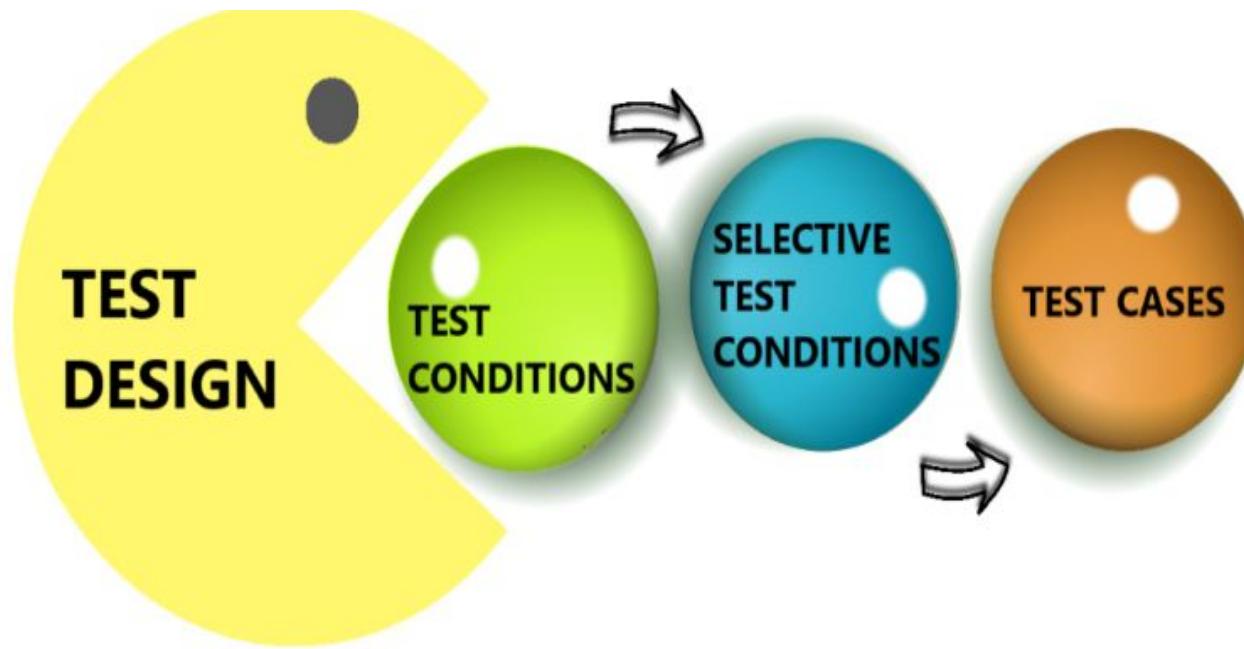


- Testin altyapısının ve ortamının belirlendiği aşamadır.
- Test edilmesi planlanan sistemlerin ihtiyaçlar kapsamında test edilebilirliği belirlenir.
- **Gereksinimler belirlenir.**
- **Test edilemeyecek** olan modüller belirlenir.
- İş analisti ile beraber çalışarak yazılımın içinde bulunduğu alan analiz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

# Requirement Traceability Matrix

Req No	Req Desc	Testcase ID	Status
123	Login to the application	TC01,TC02,TC03	TC01-Pass TC02-Pass
345	Ticket Creation	TC04,TC05,TC06, TC07,TC08,TC09 TC010	TC04-Pass TC05-Pass TC06-Pass TC06-Fail TC07-No Run
456	Search Ticket	TC011,TC012, TC013,TC014	TC011-Pass TC012-Fail TC013-Pass TC014-No Run

# Test Tasarımı



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



**Richmond  
College**

Your Access To Future  
With Advanced Ed.

# **ISTQB**

## CERTIFICATION PREPARATION COURSE

**ISTQB**  
**01/08/2023**  
**Ders -2**

# Test Süreci



# Test Oluşturma

**Testleri yapmak için hersey yerli yerinde mi?**

- 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 Environment

- ✓ Dev Environment  
(Development)
- ✓ Test / QA Environment  
(Test ortamı)
- ✓ Stage Environment  
(sahne) –Pre Prod (UAT)
- ✓ Prod – Canlı (uretim)



# Test Koşturma



**Test execution**

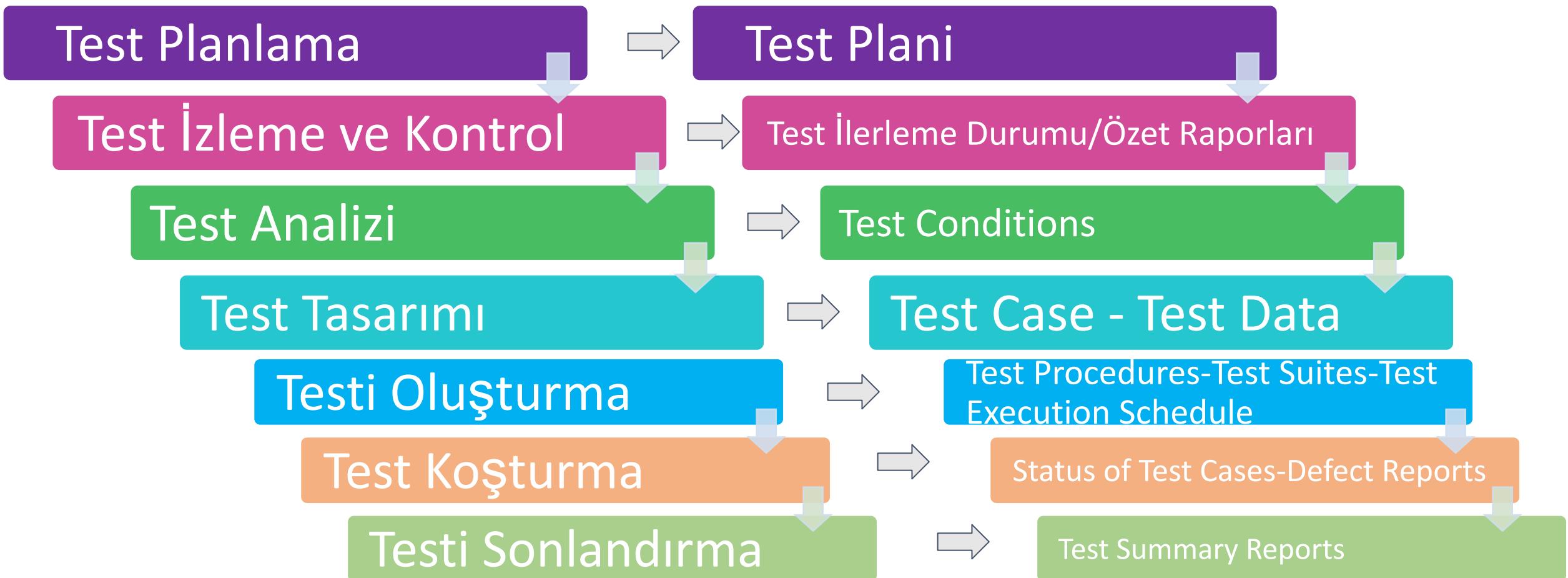
- Belirlenen sıra ile test senaryoları manuel yada otomasyon aracı kullanılarak çalıştırılır.
- Test sonucunda elde sonuçlar ile beklenen sonuçlar karşılaştırılır.
- Eğer karşılaştırmada bir hata bulunursa bildirilir.
- Çıkan hatalara neden olan temel sorunların tespit edilmelidir..
- Hatalar düzeltildikçe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

# Test İş Urunleri (Test Work Products )



## 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.





## 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



**Richmond**  
**College**  
Your Access To Future  
With Advanced Ed.

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

## 2. Bolum

# SDLC Modelleri ve Test

# Software Development Life Cycle

## SDLC

### Yazılım Geliştirme Yaşam Döngüsü

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



## Planning and Analysis

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



### Defining Requirement

- ✓ İhtiyaç analizi yapıldıktan sonra ürün gereksinimleri tanımlanır.
- ✓ Gereksinimleri tanımlama aşamasında BRD ve FRD hazırlanır.  
(Business Analyst hazırlar)
- ✓ Detaylı dokümantasyonun yapıldığı aşamadır.
- ✓ BRD (Business Requirement Document)
- ✓ FRD (Functional Requirement Document)



## Designing

- ✓ Görseller ve tasarım oluşturulur
- ✓ Birden fazla öneri sunarlar.
- ✓ Sunulan öneriler DDS' de belgelenir.  
(Design Document Specification)
- ✓ Yapılan öneriler BRD' ye göre yapılır.



## Development- Building

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



## Testing

- ✓ Bu aşamada, geliştirilen yazılım kapsamlı bir şekilde test edilir ve bulunan tüm kusurlar, düzeltmeleri için geliştiricilere atanır.
- ✓ ürün BRD'de tanımlanan kalite standartlarına ulaşıncaya kadar, ürün kusurlarının (bug) rapor edildiği, izlendiği, düzelttiği ve tekrar test edildiği aşamadır.



# 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.



## MAINTENANCE

- ✓ Sahadan veya kullanıcılarından gelen geri bildirim ile yazılım üzerinde düzeltici, uyarıcı, iyileştirici ve önleyici bakımlar gerçekleştirilmektedir.



# SDLC Modeli

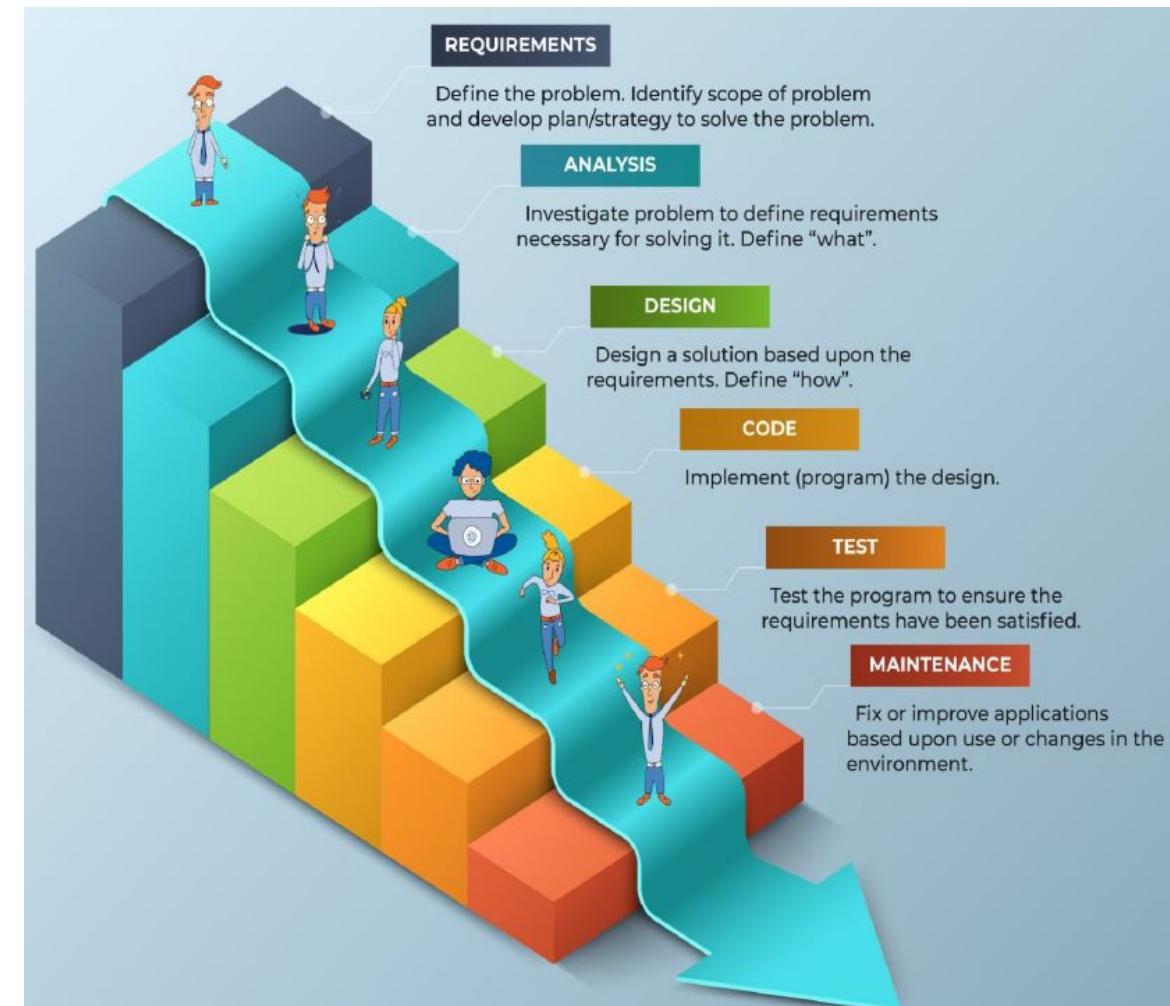


- ✓ Yazılım geliştirilirken bir dizi aktiviteyi ve olayları içeren bir yaklaşım şeklinde özetleyebiliriz
- ✓ En klasik yazılım geliştirme modeli, şelale modelidir. Şelale modeli dışında, v modeli, prototip modelleme, spiral model, çevik geliştirme gibi pek çok model bulunmaktadır.

# 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 girdisi**dir. Bir sonraki aşamanın geliştirilmesi ancak önceki aşama tamamlandığında başlar.



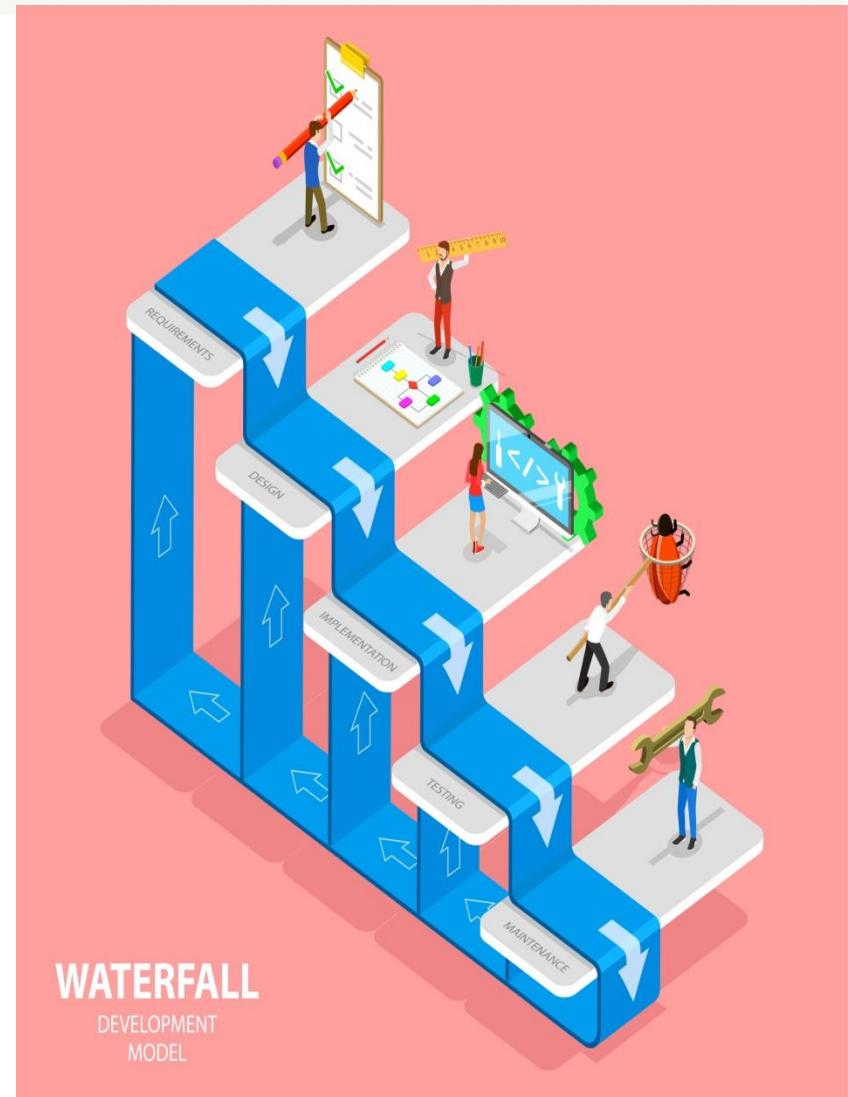
# Waterfall Model

## AVANTAJLARI

- Kullanımı ve yönetimi kolaydır.
- Gereksinimler iyi anlaşılır.
- Proje bilgisini aktarmak daha kolaydır.
- Küçük projeler için daha iyidir
- Görevler mümkün olduğunca sabit kalır
- Kapsamlı dökümanlar oluşturulur.

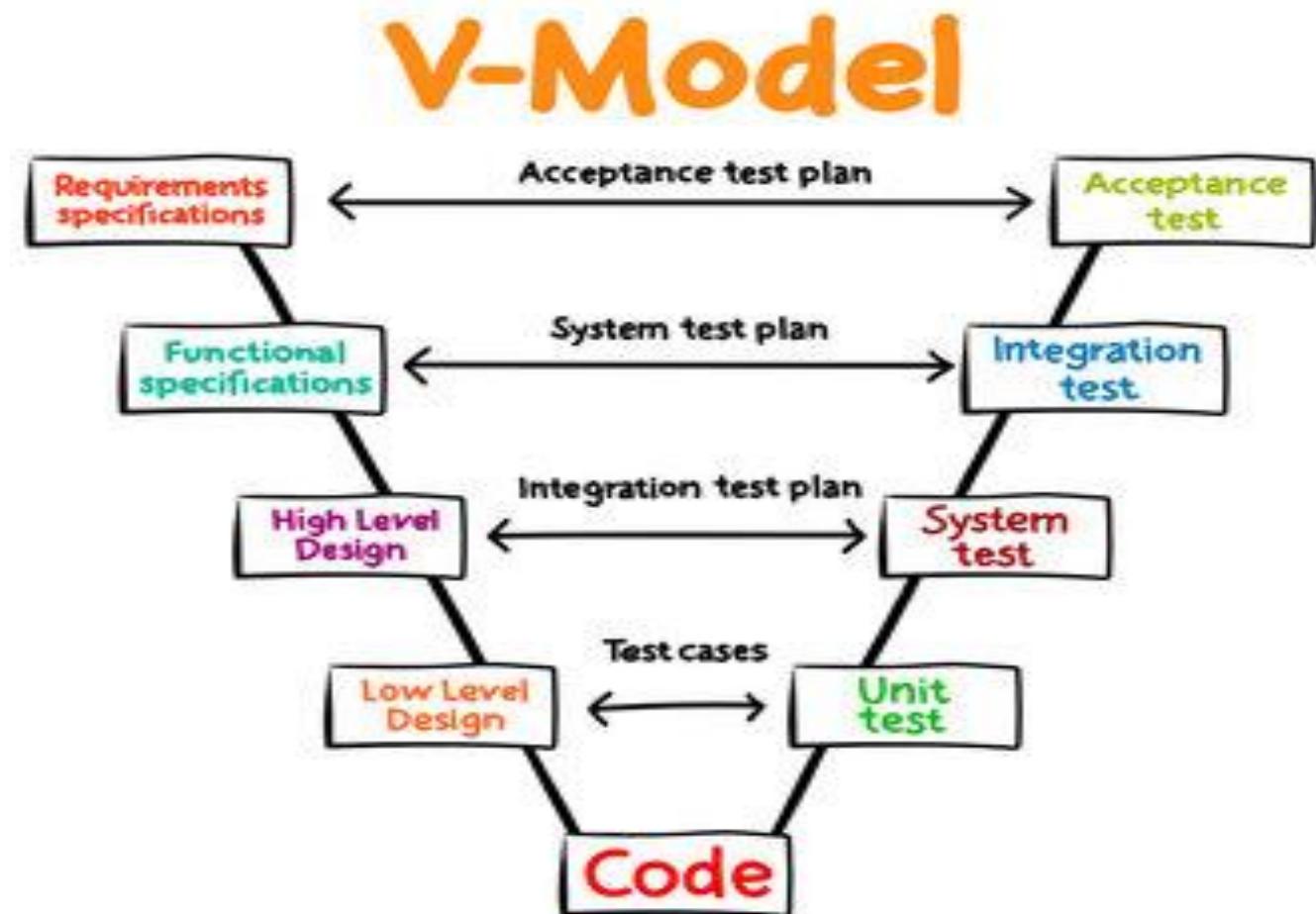
## DEZAVANTAJLARI

- Değişim ve yenilik zordur.
- Müşteri öngörü ve önerileri önemsenmez.
- Projenin bitimine kadar çalışan ürün yok.
- Beklenmedik riskleri kolayca ele alamıyor.

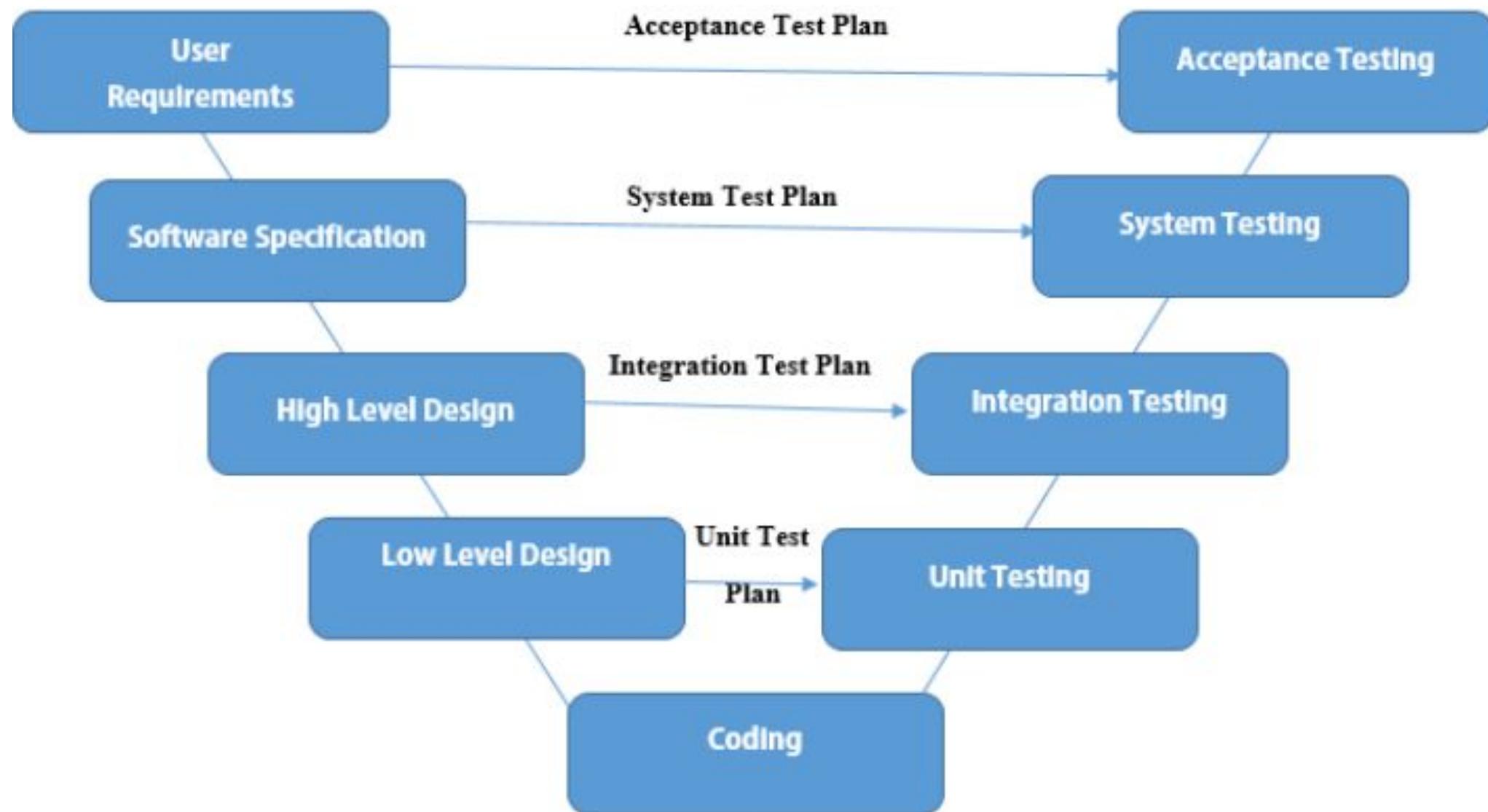


# V - Model

Şelale modelinden farklı olarak V modeli, erken test ilkesini uygulayarak test sürecini geliştirme süreci boyunca entegre eder.



# V – Model



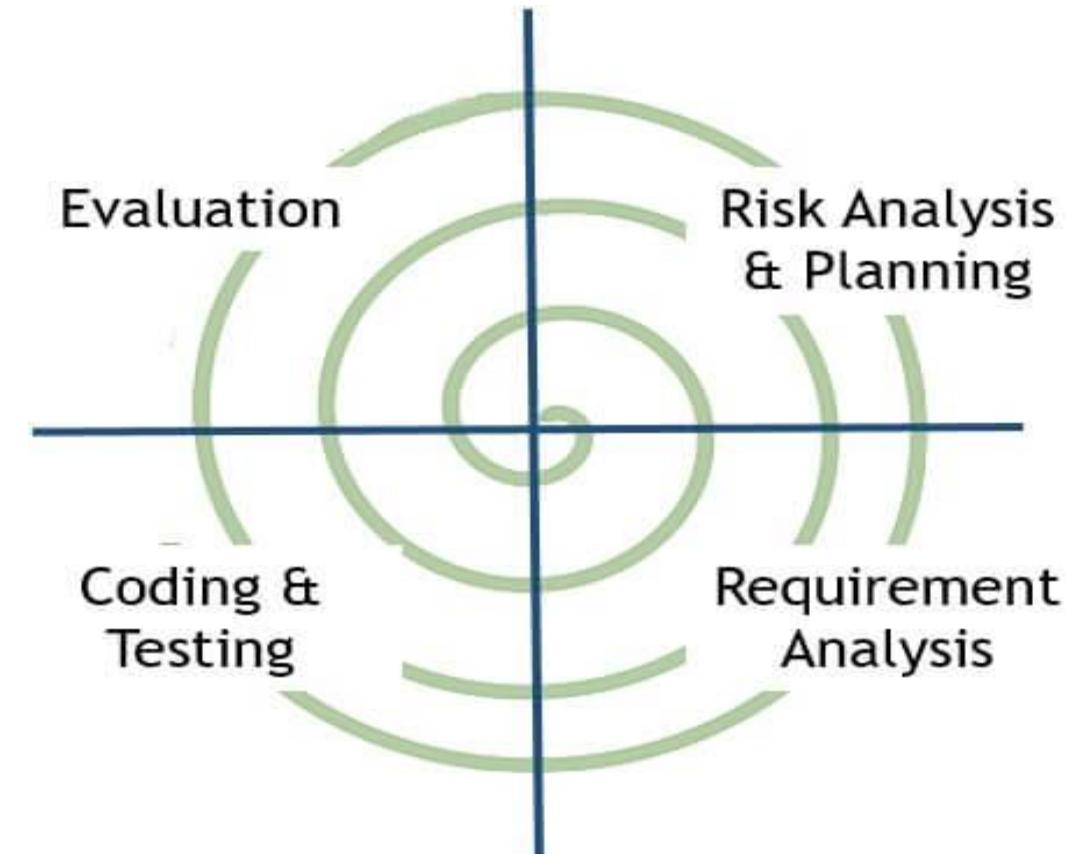
# Spiral Model

## Avantajları

- 1.Risk analizi yapmaktadır.
- 2.Bu yazılım tasarım modeli, büyük yazılım projelerini tasarlamak ve yönetmek için daha uygundur.

## Dezavantajları

- 1.Risk analizi yüksek uzmanlık gerektirir.
- 2.Kullanması pahalı model
- 3.Küçük projeler için uygun değildir.



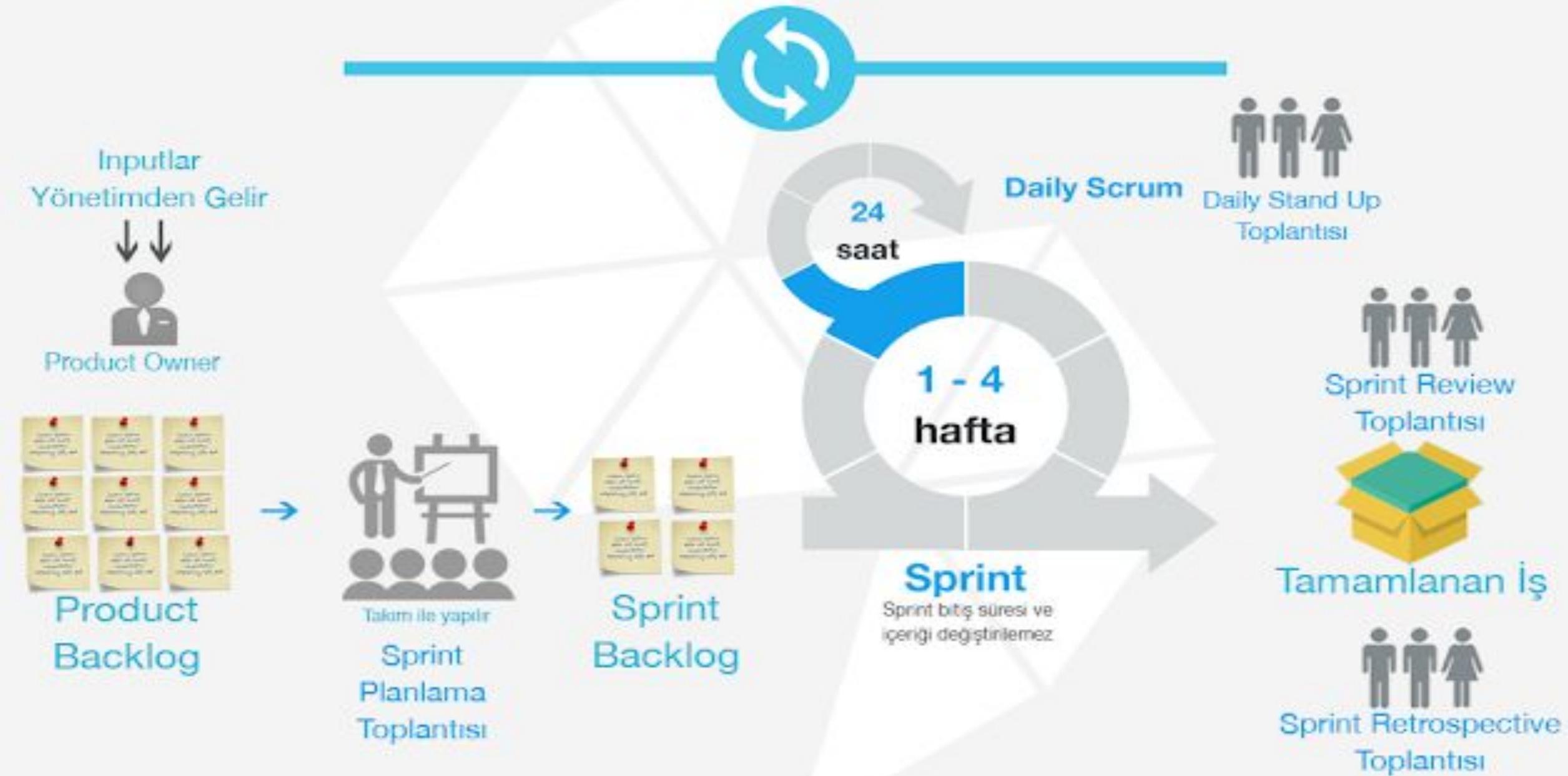
**Spiral Process Model**



## Agile(Çevik) Methodology

- Herhangi bir projenin SDLC süreci devam ederken, geliştirme ve testin sürekli olarak etkileşimde bulunmasını destekler.
- Yazılım sistemlerini etkili ve verimli bir şekilde modellemeye ve dokümantasyonunu yapmaya yönelik, pratiğe dayalı bir yöntemdir.

# SCRUM PROJE YÖNETİMİ SÜRECi



# KANBAN PRATİKLERİ NELERDİR?

1

GÖRSELLEŞTİR



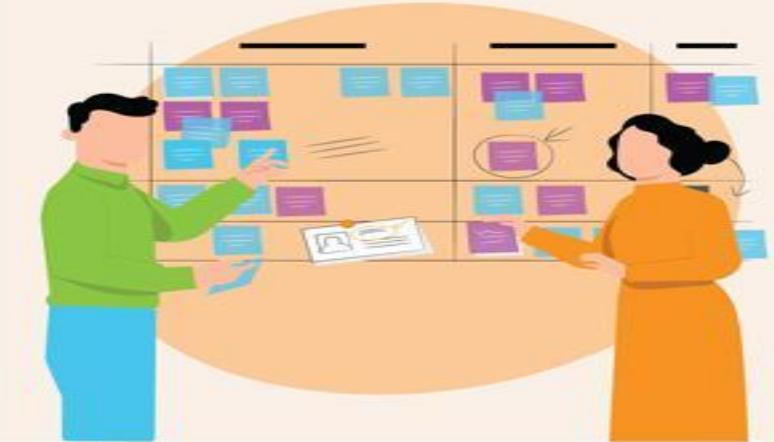
2

DEVAM EDEN İŞLERİ  
LİMİTLE



3

AKIŞI YÖNET



4

POLİTİKALARINI  
AÇIK HALE GETİR



5

GERİ BİLDİRİM  
DÖNGÜLERİ KUR



6

İŞBİRLİĞİ İLE İYİLEŞ,  
DENEYSELLİKLE EVRİMЛЕŞ  
(BİLİMSEL YÖNTEM VE MODELLERİ KULLAN)



# KANBAN

## Kanban panosu

Her sütun iş akışınızdaki bir adımı temsil eder ve her kart bir iş öğesini temsil eder.



## SDLC Model Summary



Methodoloji yönünden iyi yada kötü gibi karşılaştırma, doğru değildir, her methodolojinin kendine göre avantajları yada dezavantajları vardır,



The word "SUMMARY" is spelled out using colorful, three-dimensional blocks. Each letter is a different color: S (purple), U (orange), M (red), M (green), A (teal), R (blue), and Y (yellow). The blocks are arranged horizontally and have a slight shadow underneath them, giving them a 3D appearance.



**Richmond**  
College  
Your Access To Future  
and with Advanced IT

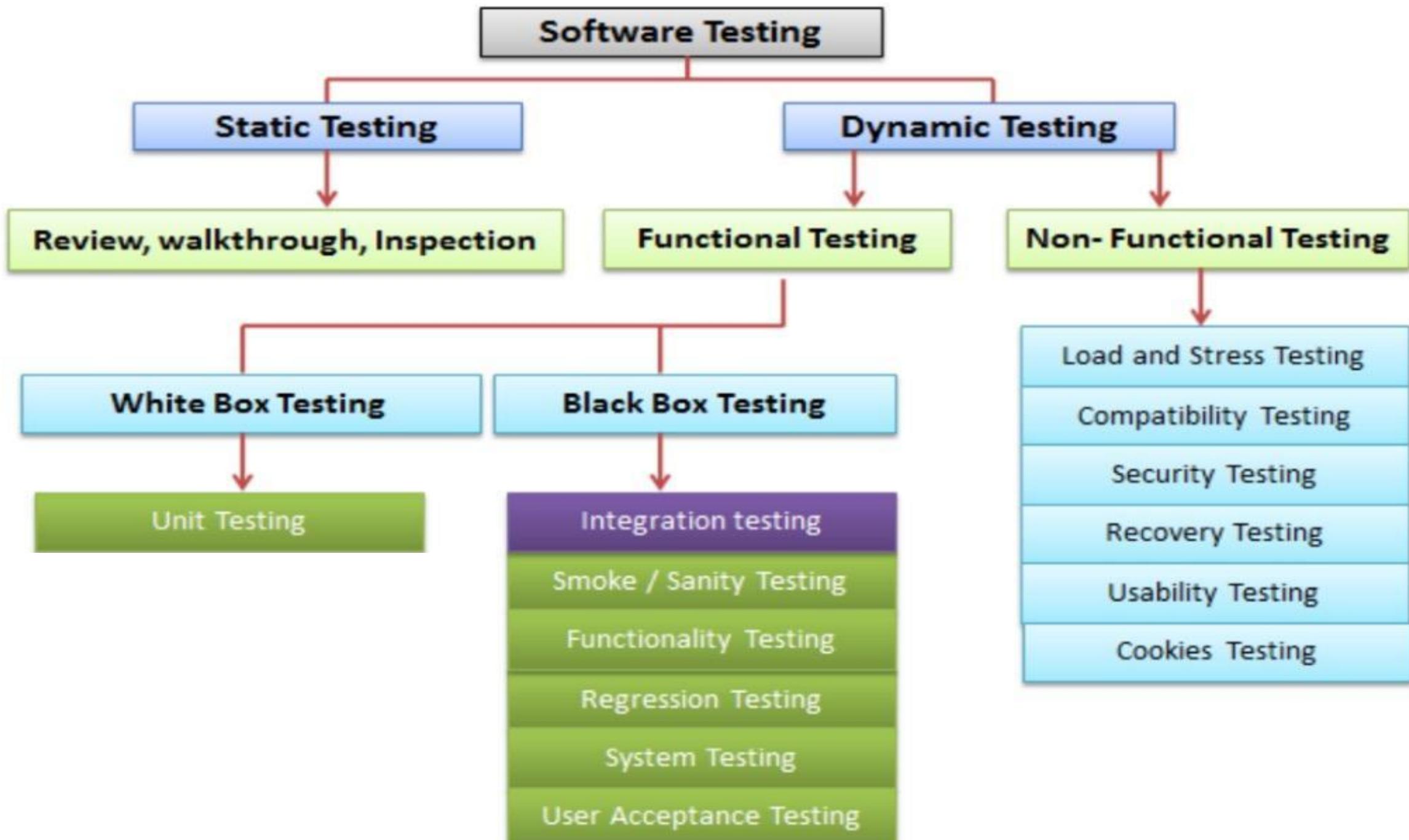
**ISTQB**  
CERTIFICATION  
PREPARATION  
COURSE

## SDLC Modelleri ve Test

# Test Çeşitleri

 **Richmond College**  
Advanced Education

Cooperation with  
**TECHPRO**  
EDUCATION





**Richmond  
College**

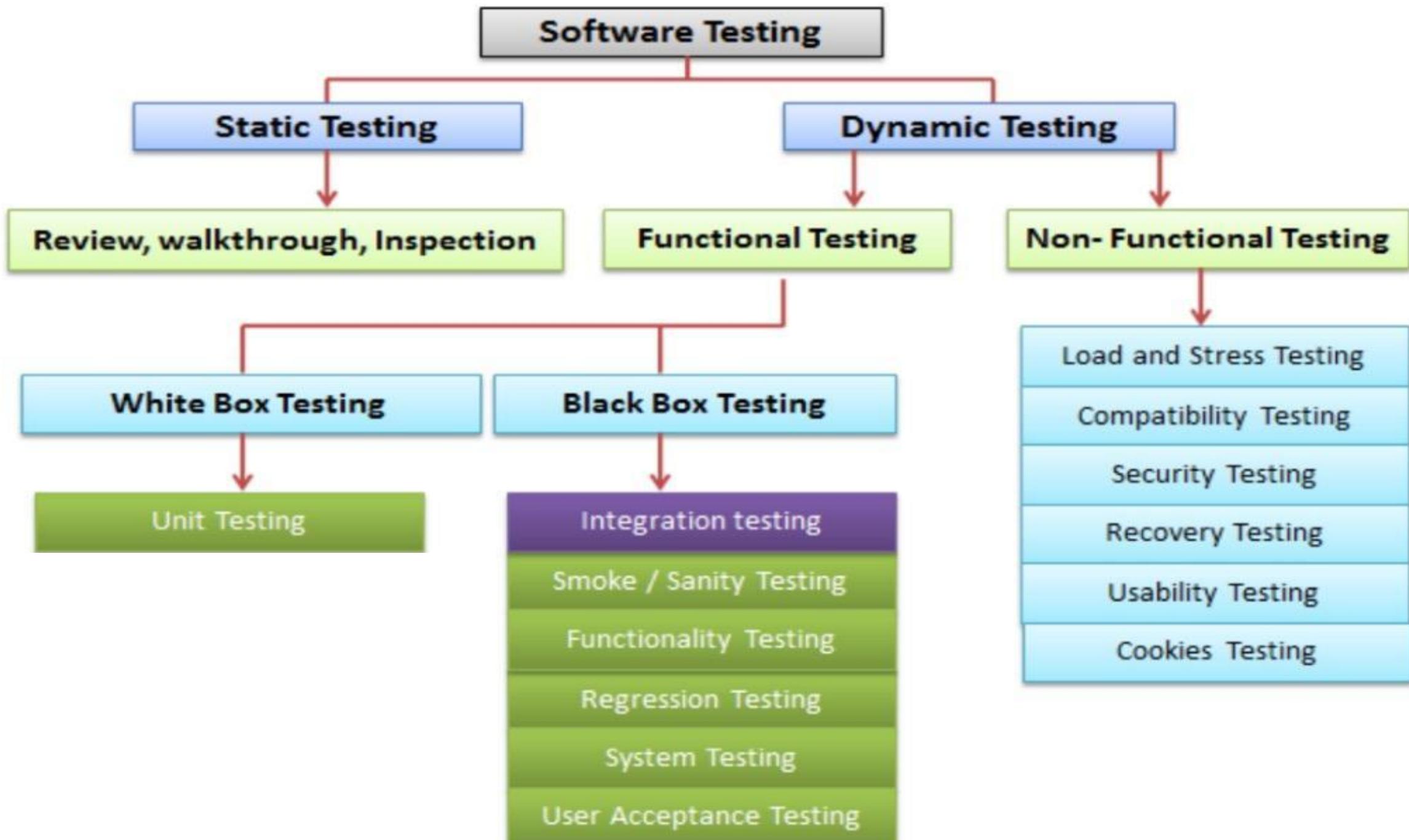
Your Access To Future

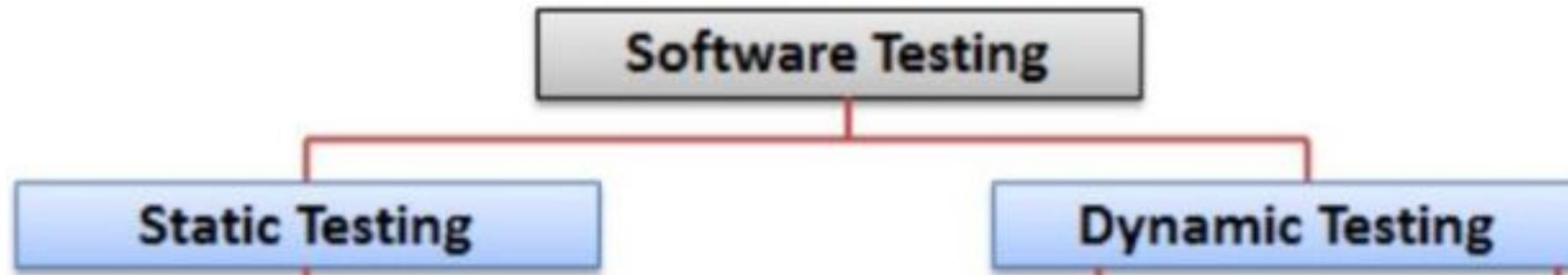
www.richmond.edu.tr

# **ISTQB**

## CERTIFICATION PREPARATION COURSE

**ISTQB  
02/08/2023  
Ders -3**





- **Statik test** kod yürütülmeden kodun veya diğer proje dokümanlarının manual olarak gözden geçirilmesidir.
- **Dinamik test** bileşenin veya sistemin yürütülmesini içeren dinamik testtir.

# Static Test



- 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.**

# Dynamic Test



## Dinamik Test:

- yazılım ürününü çalıştırarak yaptığımız testlerdir,
- Kodun bütününe çeşitli yöntemlerle test etmeye yarayan metottur.

## Verification

Doğrulama

Onaylama

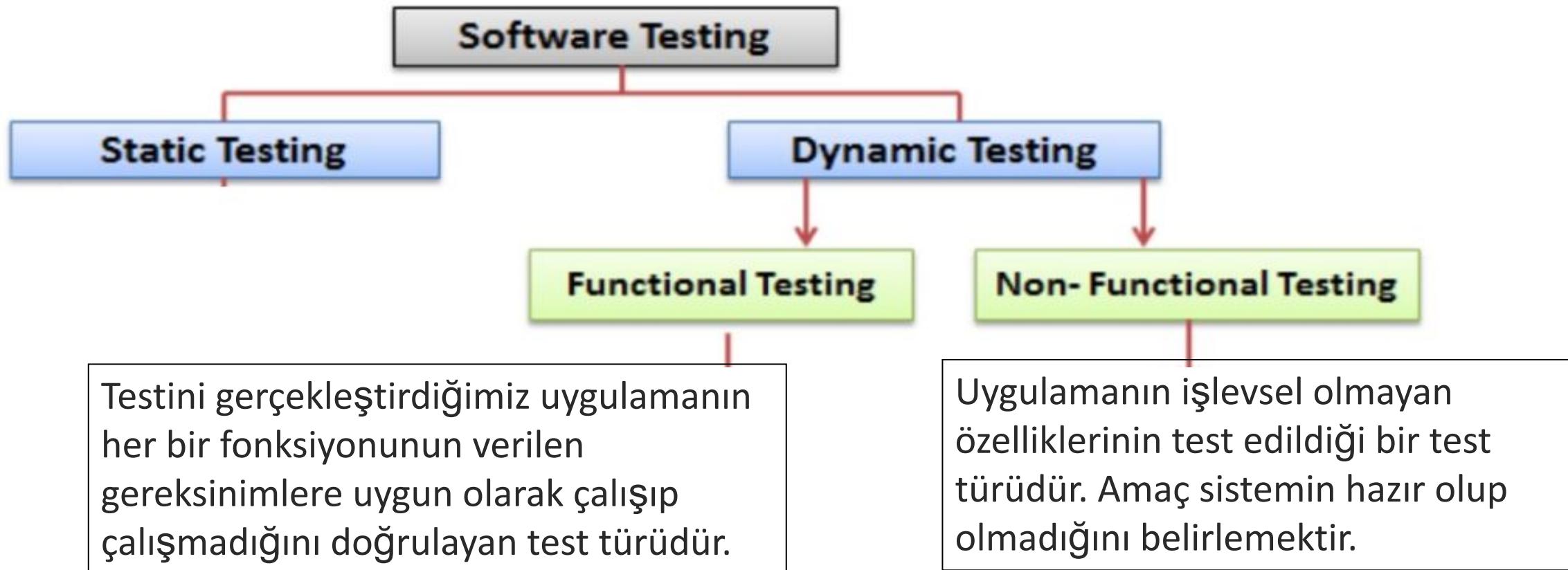
## Validation

# Verification & Validation

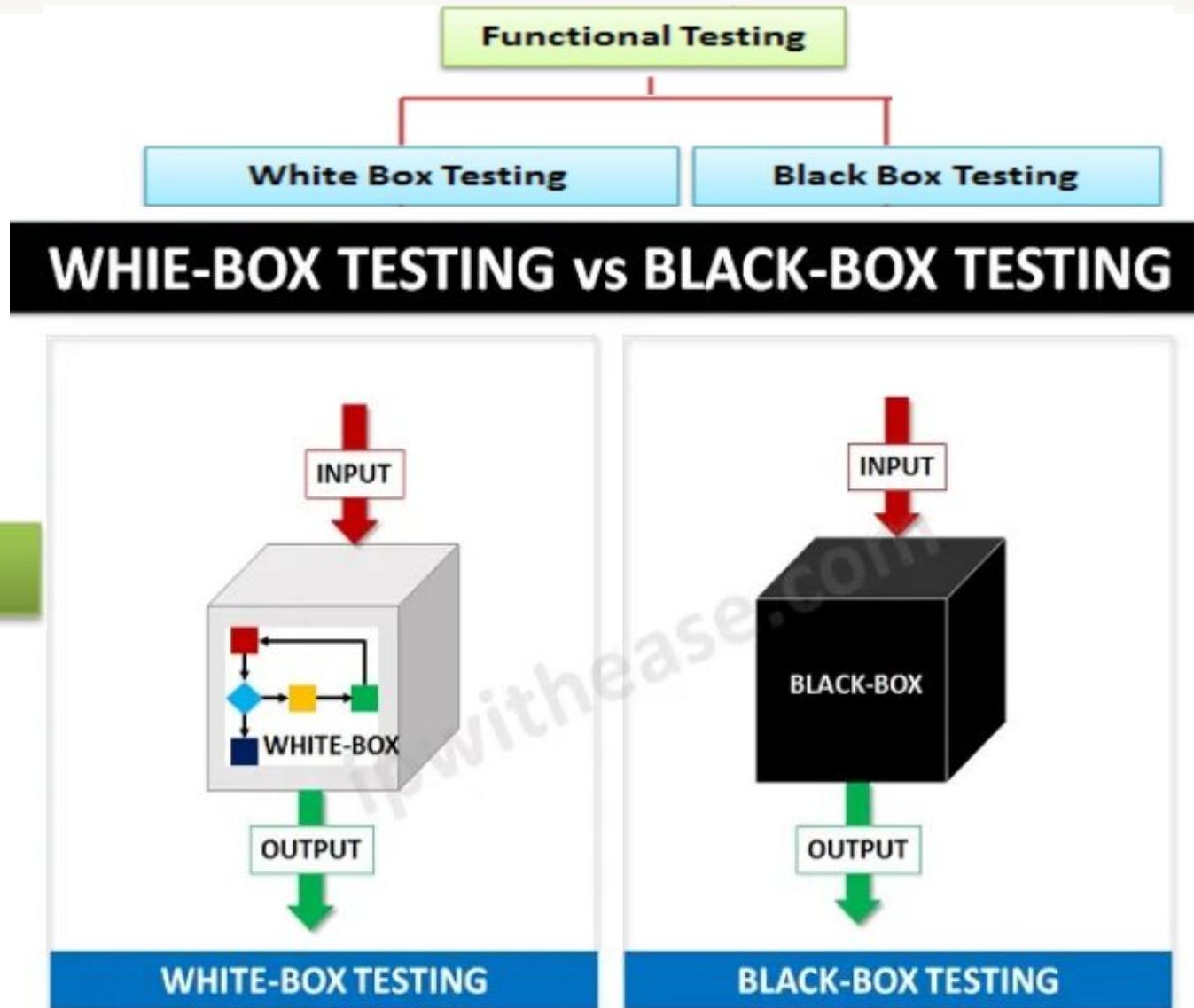
- **Verification:** Yazılımın doğru şekilde üretilmesini sağlamaktır. Geliştirme sürecinde her aşamanın çıktısı, o aşamanın gereklerine göre kontrol edilir. Doğrulama ile «Ürün doğru mu geliştirildi?» sorusuna cevap aranır.

- **Validation:** Geliştirilen yazılımın kullanım amacına uygunluğun gösterilmesidir. Onaylama ile «Doğru ürün mü geliştirildi?» sorusuna cevap aranır.

# Fonksiyonel Test & Fonksiyonel Olmayan Test



# Fonksiyonel Testler

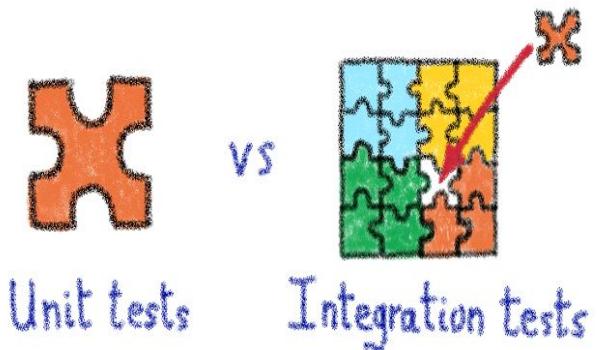


## Birim Testleri-Unit Test

- ✓ Genellikle Yazılım ekibi tarafından yapılan beyaz kutu testleridir.
- ✓ Metot, yada fonksiyon gibi **kod parçalarının** kendisinden beklenen işlevselliği **doğru olarak yerine getirip getirmedığını** ölçen ve **yazılım geliştiriciler** tarafından gerçekleştirilen test tipidir.



# Integration Test

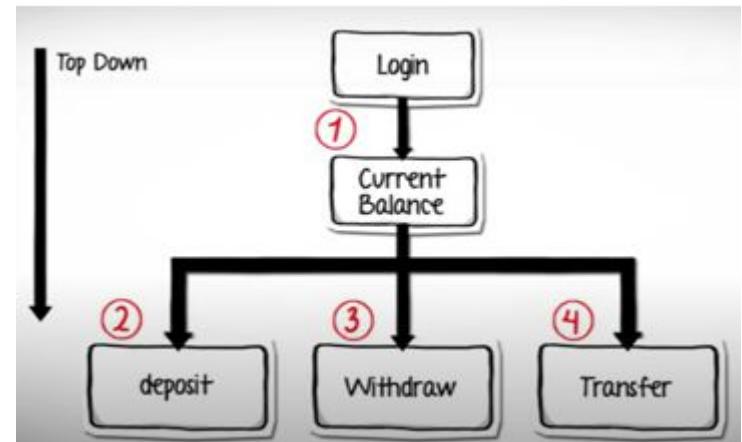


- Her modülü test ettikten sonra, entegre modüllerin istenen çıktıyi verdiğinde emin olmak için bir araya getirilir ve test edilir.
- Tipik olarak her yazılım, farklı yazılım geliştiriciler tarafından kodlanmış farklı yazılım modüllerinden oluşur.
- Bu test her bir modülün herhangi bir kusur olmadan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

# Integration Test

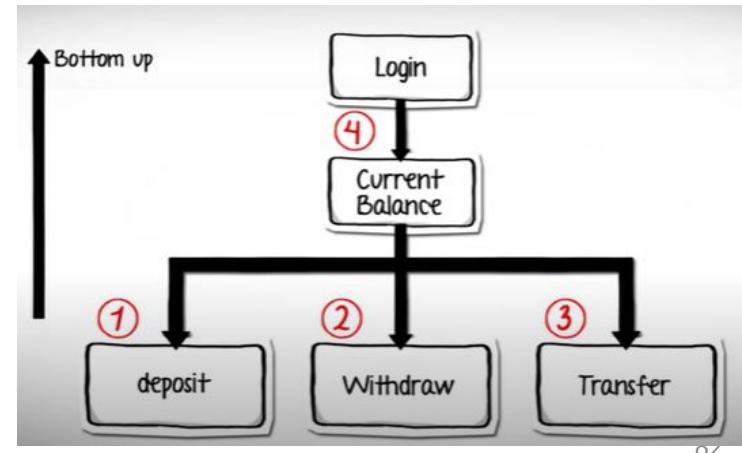
## Büyükten Küçüğe (Top-down):

Bu yaklaşımın, test süreci en üst seviyedeki bileşenlerden başlayarak alt seviye bileşenlere doğru ilerlenir. Üst seviye bileşenler kullanılarak alt seviye bileşenlerin çalışması test edilir.



## Küçükten Büyüğe (Bottom-up):

Bu yaklaşımın, test süreci Yani, alt seviyedeki bileşenlerden başlayarak üst seviye bileşenlere doğru ilerlenir. Alt seviye bileşenler kullanılarak üst seviye bileşenlerin çalışması test edilir.



# Change-related Testing

## Regresyon ve Yeniden Onaylama Testleri(Regression-Confirmation )

### Regression

Yazılım üzerinde yapılan değişiklik sonrasında mevcut özelliklerin çalışıp çalışmadığını kontrol etmek için yapılan test türüdür.

### Confirmation

Onaylama testinin amacı asıl hatanın başarıyla çözülmüş çözülmemişini onaylamaktır.

# Change-related Testing

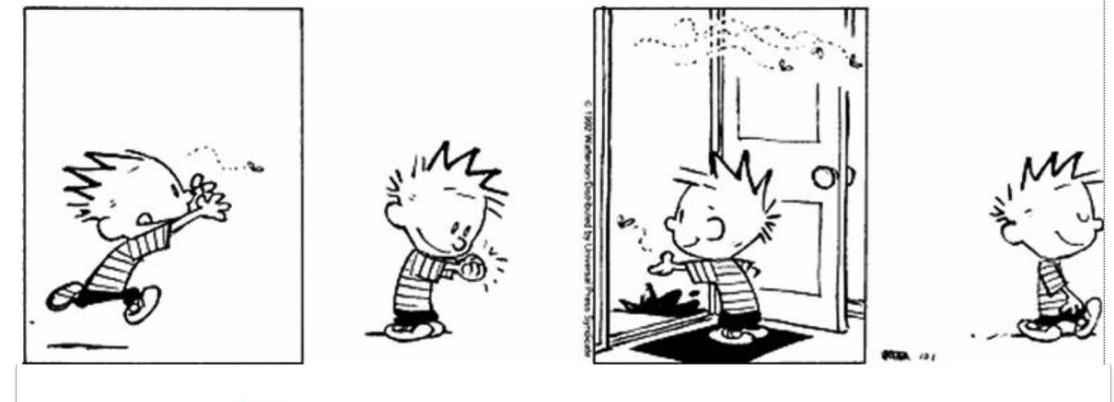
## Regression

Kes~~e~~ression

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

Regression:

"when you fix one bug, you introduce several newer bugs."



## Sistem (System) Testi



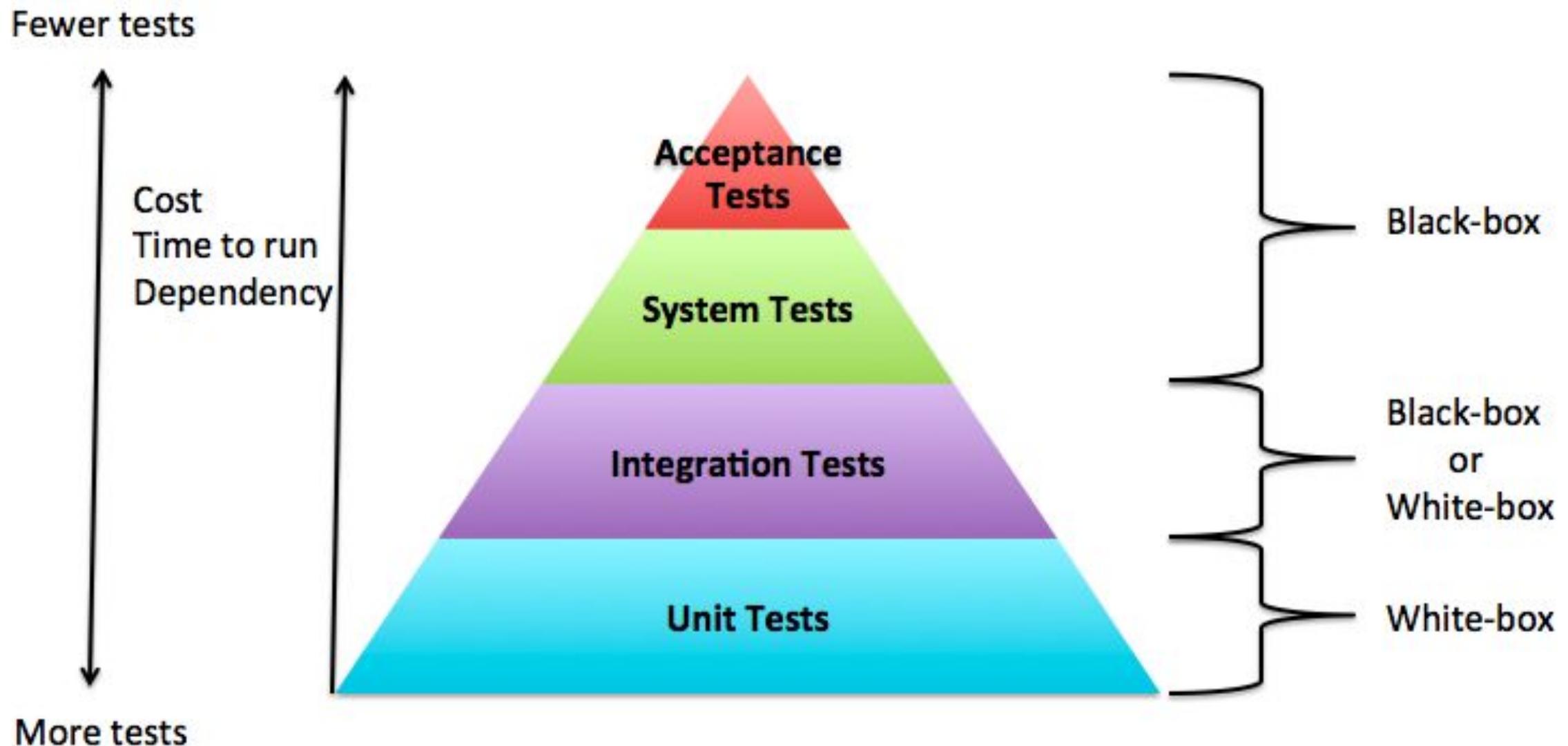
- ✓ **Bütün bir sistemin** veya **ürünün** davranış, ve yeteneklerine odaklanır ve genellikle sistemin gerçekleştirebileceği **uçtan uca(end to end)** işleri ve bu işleri gerçekleştirirken gösterdiği fonksiyonel olmayan davranışları ele alır.
- ✓ **Gereksinimlere göre** sistemin uygunluğunun kontrol edilmesini sağlar.
- ✓ **Yük, stres, performans, güvenilirlik ve güvenlik** testlerini içerir..
- ✓ Sistem testi, sistemin gereksinimlere **uygun olduğunu doğrulamak** için **yapılan son testtir.**

## Kabul (Acceptance) Testi

- ✓ 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 göstermektir**.



# Test Seviyeleri



# Yazılım Testi Neyi Amaçlar ?





**Richmond**  
**College**  
Your Access To Future  
With Advanced Ed.

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

# Test Teknikleri

 **Richmond College**  
Advanced Education

Cooperation with  
**TECHPRO**  
EDUCATION

# Test Tasarım Teknikleri

Test tasarım teknikleri

Black Box –  
Kara Kutu Test Teknikleri

White Box –  
Beyaz Kutu Test Teknikleri

Experience Based –  
Deneyim Temelli Test Teknikleri

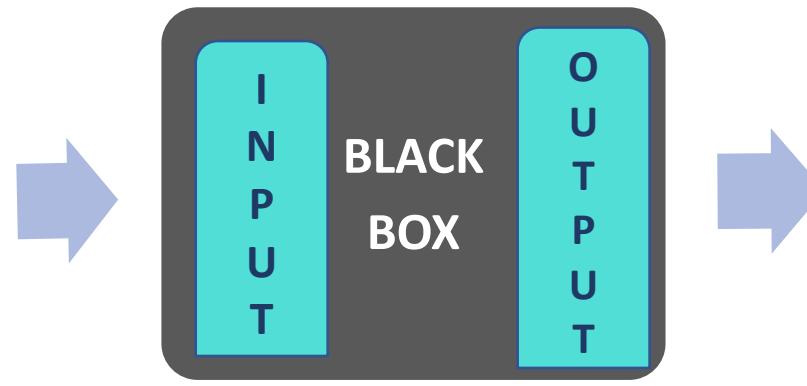
gereksinim bazlı  
test tasarım  
teknikleri

yapı bazlı  
test tasarım  
teknikleri

deneyim bazlı  
test tasarım  
teknikleri

# Kara kutu Test Tasarım Teknigi

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 tasarımı tekniği:** Bileşenin veya sistemin iç yapısı bilinmeden, bileşenin veya sistemin işlevsel veya işlevsel olmayan gereksinimi analiz edilerek test durumları hazırlamak.

# Kara Kutu Test Tasarım Teknikleri

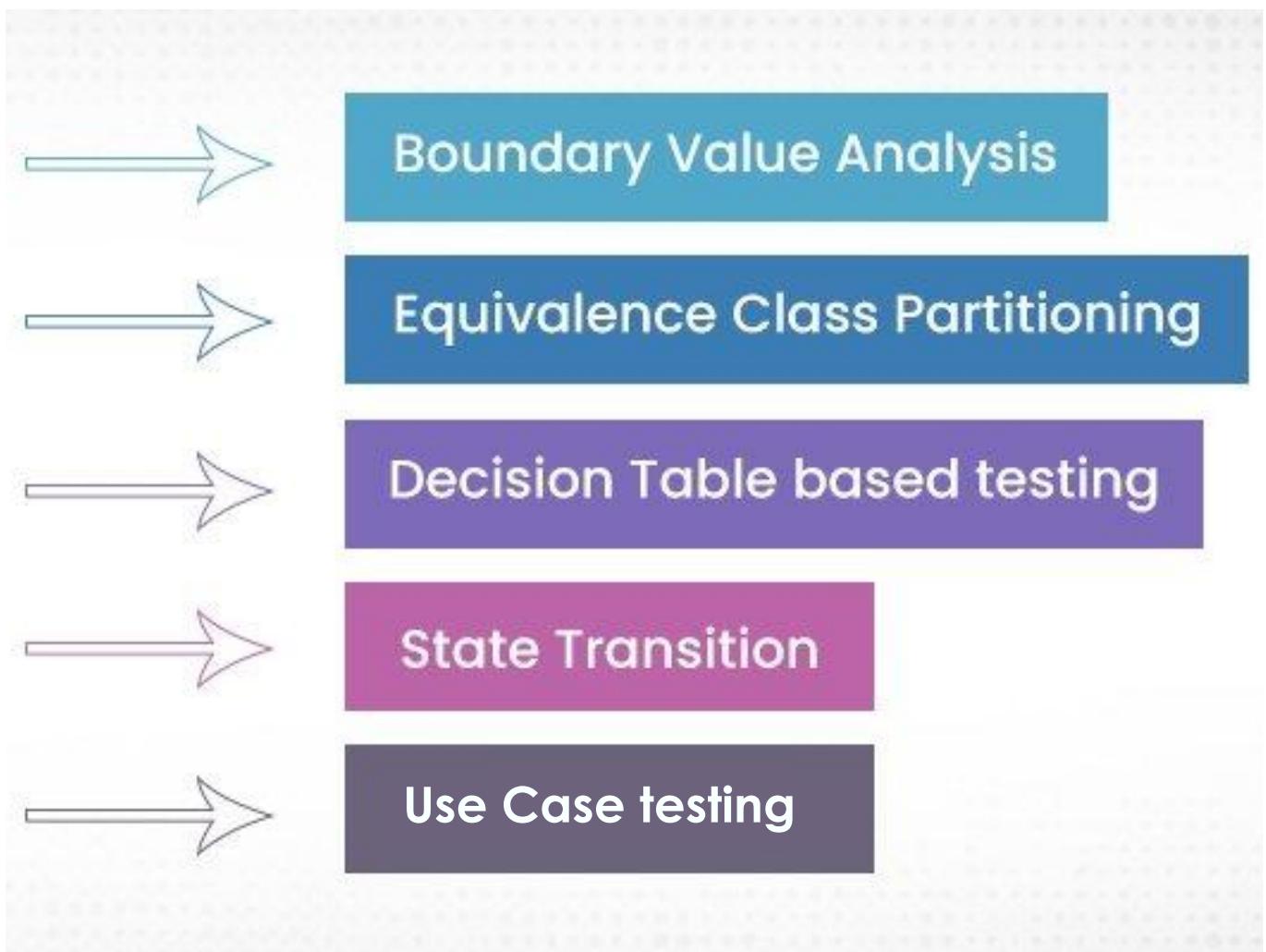
Sınır Değerleri  
Analizi

Denklik Paylarına Ayırma  
Tekniği

Karar Tablosu

Durum Geçiş Tablosu

Kullanım Durumları Testi



# Denklik Paylarina Ayırma (Equivalence Partitioning)

Lütfen yaşınızı girin ve Hesapla butonuna tıklayın.

(\*) 1-120 arasında değer girin.

Yaşınız\*:

Hesapla

Geçerli ve geçersiz test girdilerinin (denklik sınıflarının) aşağıdaki gibi olduğunu söyleyebiliriz.

- Geçersiz Aralık: 0 ve 0'dan küçük sayılar
- Geçerli Aralık: 1-120 arasındaki sayılar
- Geçersiz Aralık: 121 ve 121'den büyük sayılar

Denklik sınıflarının doğru belirlenmesi test tasarımanızı ve dolayısıyla yapacağınız testlerin kalitesini olumlu şekilde etkileyecektir.

# Sınır Deger Analizi – Boundary Value Analyses

Lütfen yaşınızı girin ve Hesapla butonuna tıklayın.

(\*) 1-120 arasında değer girin.

Yaşınız\*:

Hesapla

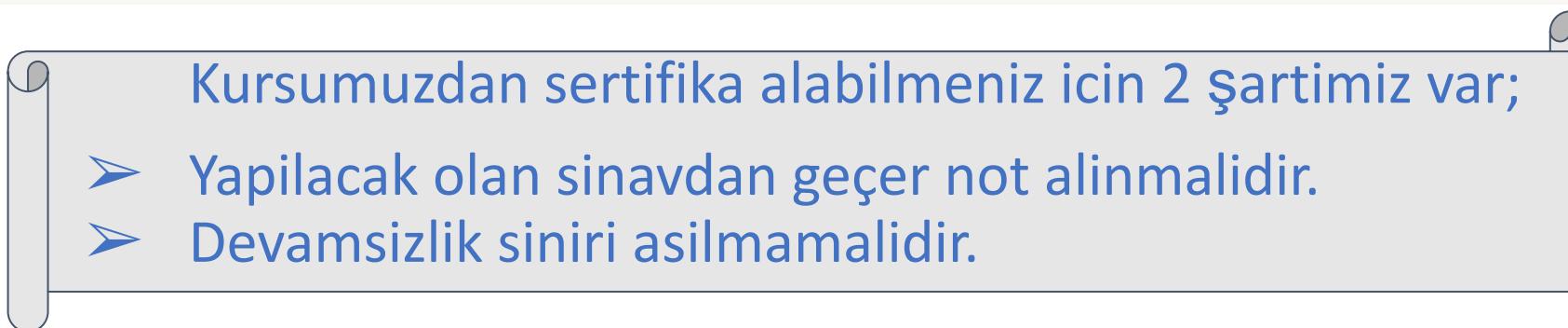
Formül	Test Girdisi
Geçersiz Aralık: Alt sınır değeri – 1	0
Geçerli Aralık: Alt sınır, Alt sınır + 1 ve Üst Sınır -1, Üst Sınır	1,2 – 119,120
Geçersiz Aralık: Üst Sınır + 1	121

## Karar Tablosu (Decision Table)

- ✓ Eğer kontrol edilecek çok sayıda durum var ise her bir durumu tablo içerisine alınır ve **her bir duruma karşı yazılımın vereceği cevaplar** yine tabloya yerleştirilerek karar tablosu oluşturulur.
- ✓ Karar tablosu testinde **tabloda yer alan sütunlardan her biri için bir test** yapılır.
- ✓ Bu sayede **test sırasında gözden kaçabilecek koşul kombinasyonlarının net bir şekilde listelenmesi**dir.
- ✓ Yazılım davranışının birden fazla mantıksal karara bağlı olduğu **tüm durumlarda uygulanabilir**.

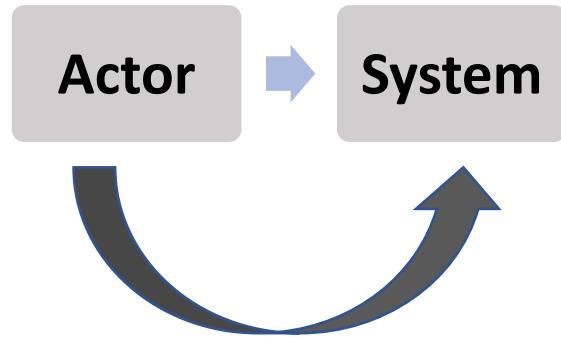
Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Minimum 8 Karakter	True	True	True	True	True	True
Rakam içermesi	False	True	True	True	False	False
Harf içermesi	False	False	False	True	True	False
Actions						
Geçerli Şifre	HAYIR	HAYIR	EVET	EVET	EVET	EVET

# Karar Tablosu (Decision Table)



Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Geçer not alındı	True	True	False	False
Devamsızlık sınırı aşılmadı	True	False	True	False
Actions				
Sertifika alabilir	True	False	False	False

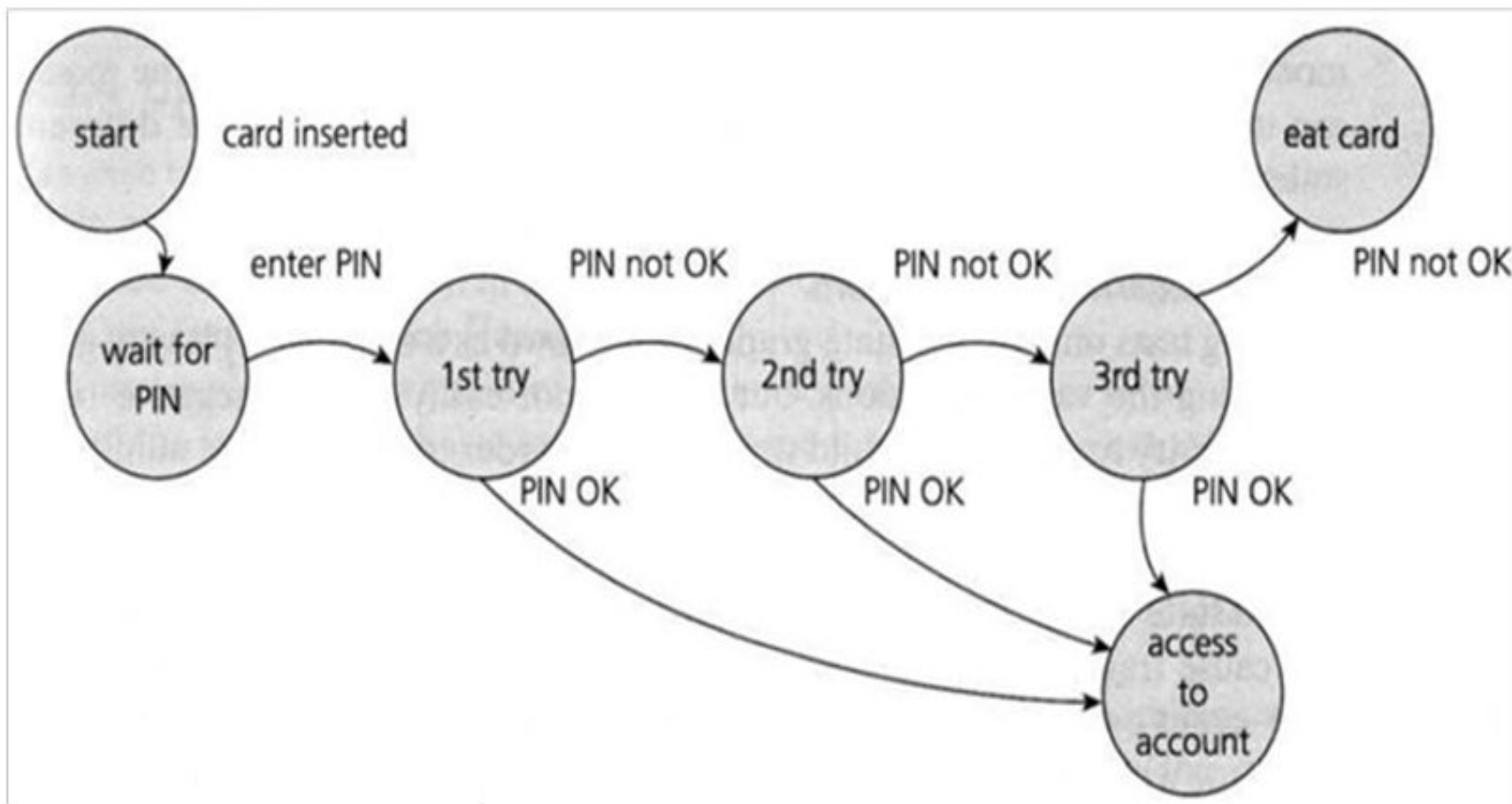
# Kullanici Durum Testi (Use Case)



**HAPPY PATH**

	<b>Step</b>	<b>Description</b>
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

# Durum Gecis Testi (State Transition)



1. ATM örneğinde ilk olarak PIN kodu bekleniyor
2. daha sonra girilen PIN kodu kontrol ediliyor eğer PIN kodu doğruysa kullanıcı hesabına erişebiliyor
3. hatalı PIN kodu girişinde ise ATM kartı yutuyor.



**Richmond**  
**College**  
Your Access To Future  
With Advanced Ed.

# **ISTQB**

## CERTIFICATION PREPARATION COURSE

**ISTQB**  
**03/08/2023**  
**Ders -4**

# 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



**Richmond**  
**College**  
Your Access To Future  
and with Advanced Ed.

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

# Beyaz Kutu Test Teknikleri

 **Richmond College**  
Advanced Education

Cooperation with  
**TECHPRO**  
EDUCATION

# Beyaz Kutu Test Tasarım Teknikleri

İfade Kapsama Testi  
(Statement Coverage Testing )

Karar Kapsama Testi  
(Decision CoverageTesting)

Diğer Yapısal Testler

# Ifade Kapsami (Statement Coverage)

- Aşağıdaki kod parçası için:
  - %100 ifade kapsamı (statement coverage) elde etmek için gereken minimum test case sayısı kaçtır?

A) 1  
B) 2  
C) 3  
D) 4

```
If x = 3  
Print ("hello")
```

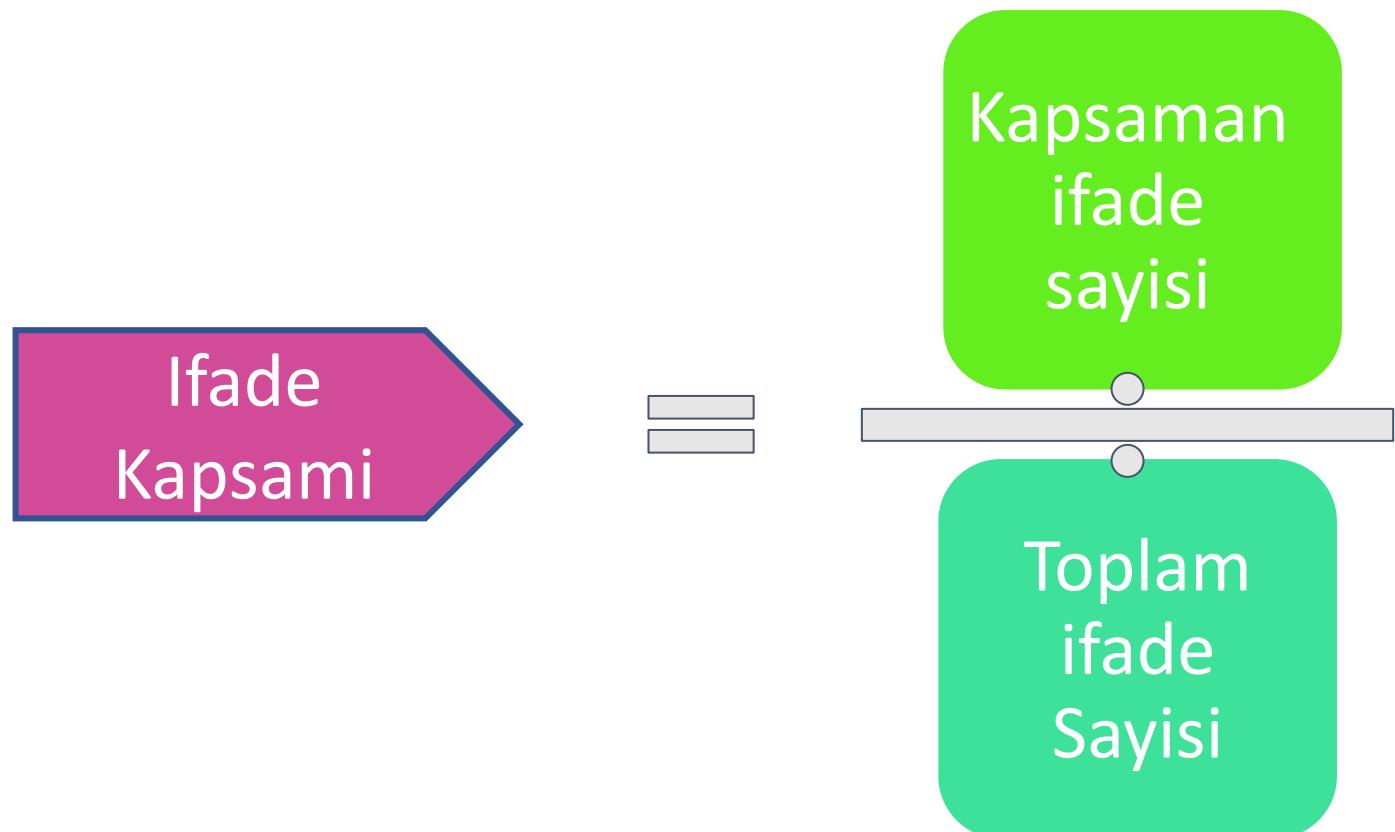
# Ifade Kapsamı (Statement Coverage)

- Aşağıdaki kod parçası için:
  - %100 ifade kapsamı (statement coverage) elde etmek için gereken test case sayısı kaçtır?

- A) 1
- B) 2
- C) 3
- D) 4

```
Print sum (int a, int b) {  
    int result = a + b;  
    if (result > 0)  
        print ("red", result)  
    else if (result < 0)  
        print ("blue", result)  
}
```

# Ifade Kapsami (Statement Coverage) Nasil Olculur?

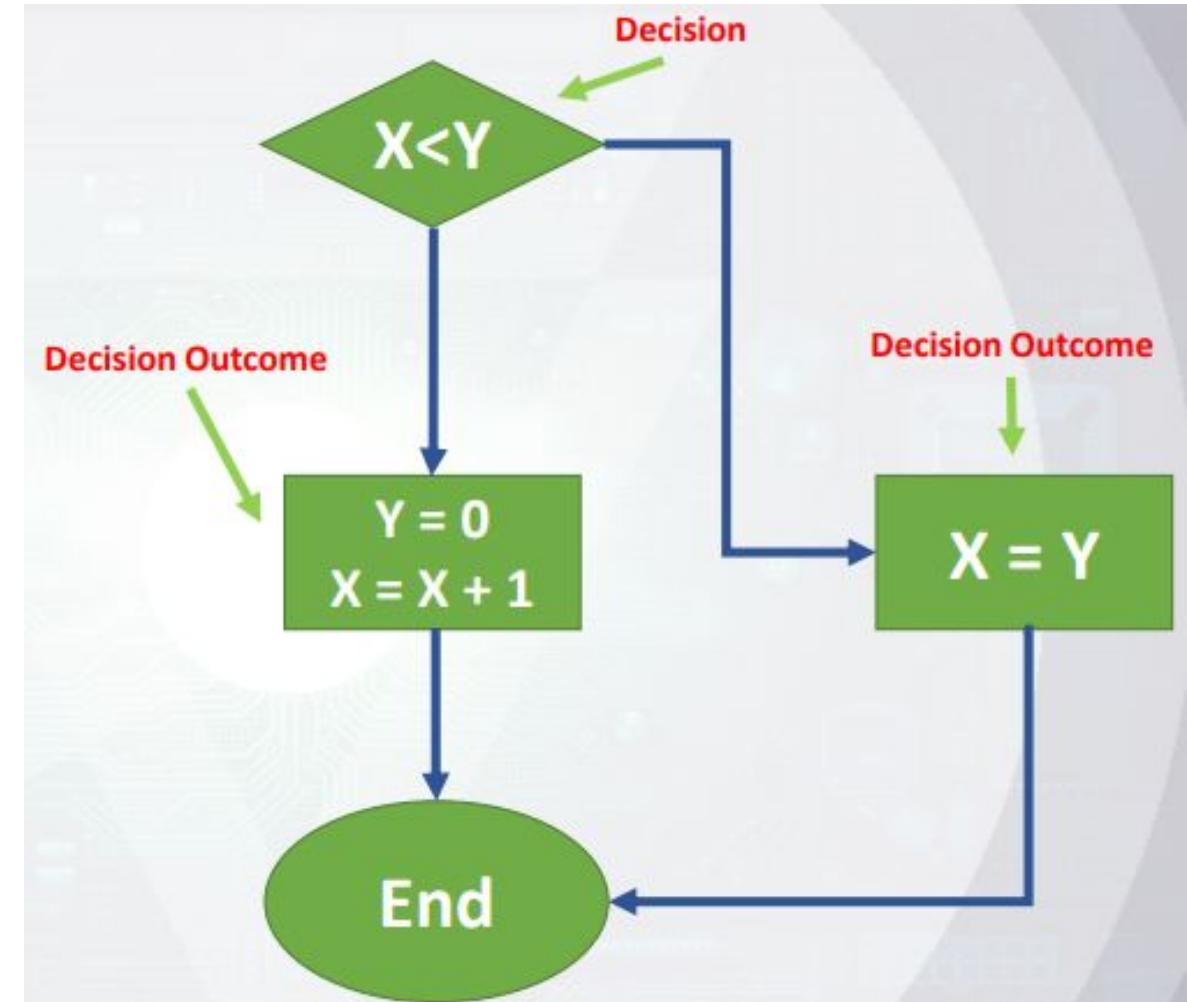


→ ifade kapsamını olcmek için kapsanan ifade sayısını toplam ifade sayısına böleriz.

## Karar Kapsami (Decision Coverage)

- ✓ Tüm (branch) dalların ya da algoritmada ki döngülerin işletilmesi için gerekli olan en az test sayısı bulunur.
- ✓ Bu örnekte  $X < Y$  vermemiz gereken bir karardır, bu kararın true yada false olmasına göre yapılan işlemler karar sonucudur

```
If X < Y  
Y = 0;  
X = X + 1 ;  
Else  
X = Y ;
```





**Richmond**  
**College**  
Your Access To Future  
and with Advanced Ed.

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

# **Deneyimsel Temelli Test Tasarım Teknikleri**

 **Richmond College**  
Advanced Education

Cooperation with  
**TECHPRO**  
EDUCATION

# Tecrubeye Dayalı Test Teknikleri(Experience-based)



Testi yapan kişinin yaklaşımına ve deneyimine bağlı olarak, bu teknikler çok çeşitli derecelerde kapsam ve etkililik sağlayabilir.

Hata tahminleme (Error Guessing), test uzmanın bilgilerine dayalı olarak, hataların bulunması için kullanılan bir tekniktir;

- Uygulamanın geçmişte nasıl çalıştığı
- Yazılımcıların yapmaya eğilimli oldukları hata çeşitleri
- Diğer uygulamalarda oluşan arızalar

# Tecrubeye Dayalı Test Teknikleri(Experience-based)

## 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

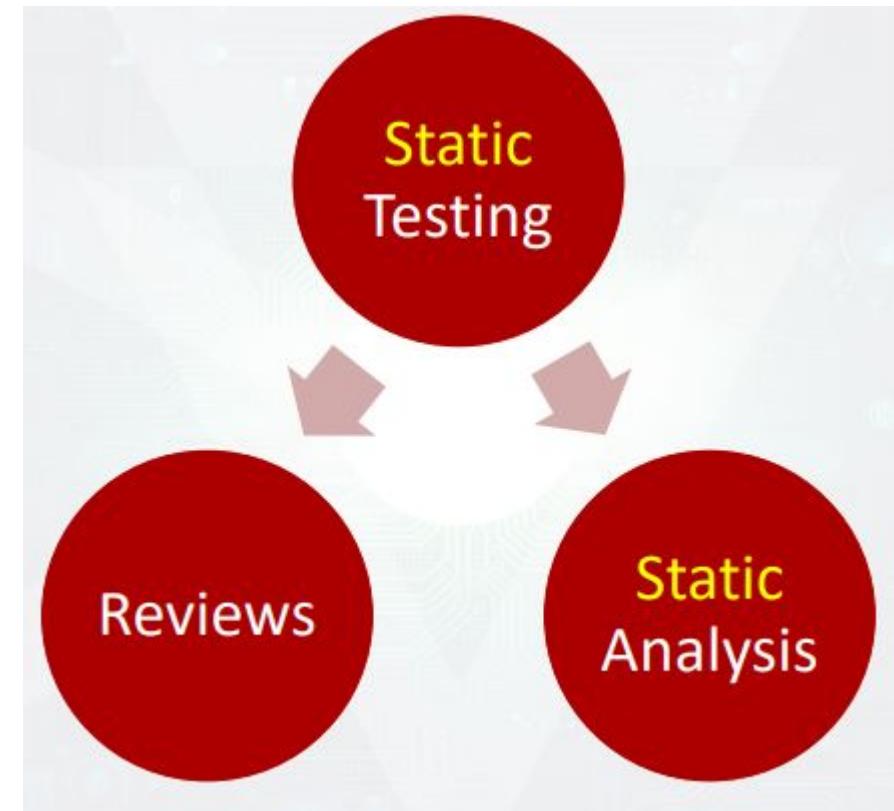


## Kontrol Listesine Dayalı Testler

- Test uzmanları bir kontrol listesinde bulunan test koşullarını kapsayacak şekilde testler tasarlar, uygular ve koşturur.
- Bu kontrol listeleri tecrübe, kullanıcı için neyin önemli olduğu veya yazılımın niçin ve nasıl başarısız olabileceği bilgisine dayanarak oluşturulabilir.

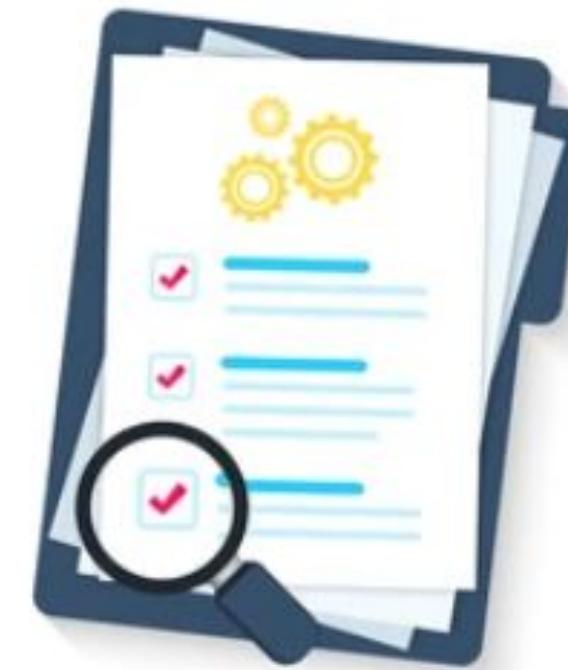
## Static Test Türleri

- Statik testlerde **herhangi bir kod çalıştırması olmaz** ve yazılı olan her şey **gözden geçirme sürecine** dahil edilerek **hata bulmak amaçlı okunur.**



## 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**.
- **Gereksinimler, tasarımlar, kod, test planları, test gereksinimleri, test senaryoları, test komut dosyaları, kullanım kılavuzları veya web sayfaları** gibi tüm yazılım ürünlerinin gözden geçirilmesidir.



# Gözden Geçirme Çeşitleri

## Gözden Geçirme Aşamaları

- Planlama, "planning,"
- İlk inceleme, initial review,
- Bireysel inceleme, individual review,
- Bulguların iletilmesi ve analiz issue communication and analysis
- Düzeltme ve raporlama". Fixing and reporting



# Resmi Gözden Geçirme İşlemleri



## 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 İşlemleri

## 2. Başlama

- ✓ Belgelerin dağıtılması
- ✓ Katılımcılara kapsam, hedefler, süreç, roller ve çalışma ürünlerinin açıklanması
- ✓ Katılımcıların incelemeyle ilgili olabilecek sorularını yanıtlamak



# Resmi Gözden Geçirme İşlemleri



## 3. Bireysel hazırlık

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

## 4. Bulguların iletilmesi ve Analiz

- ✓ Tespit edilen potansiyel kusurların iletilmesi
- ✓ Potansiyel hataları analiz etmek,
- ✓ Kalite özelliklerinin değerlendirilmesi ve belgelenmesi



## 5. Düzeltme ve Raporlama

- ✓ Hata raporları oluşturma
- ✓ Bulunan kusurların düzeltilmesi
- ✓ Kusurları uygun kişi veya ekibe iletmek
- ✓ Hataların güncel durumunu kaydetme
- ✓ Çıkış kriterlerinin karşılanması  
karşılanmadığının kontrol edilmesi
- ✓ Çıkış kriterlerine ulaşıldığındá iş  
ürününün kabul edilmesi



## Roller ve Sorumluluklar

### 1. Yazar (Author)

- ✓ İncelenmekte olan iş ürününü oluşturur
- ✓ İncelenmekte olan iş ürünündeki kusurları düzeltir (gerekirse)



### 2. Yönetim (Management)

- ✓ İncelenmekte olan iş ürününü oluşturur
- ✓ İncelenmekte olan iş ürünündeki kusurları düzeltir (gerekirse)



### 3. Moderator(Facilitator)

- ✓ Gözden geçirme toplantılarının etkili bir şekilde yürütülmesini sağlar
- ✓ Gerekirse çeşitli bakış açıları arasında aracılık yapar
- ✓ Genellikle incelemenin başarısının bağlı olduğu kişidir.



### 4. Lider(Review Leader)

- ✓ İnceleme için genel sorumluluğu alır
- ✓ Kimin dahil olacağına karar verir ve ne zaman ve nerede gerçekleşeceğini organize eder



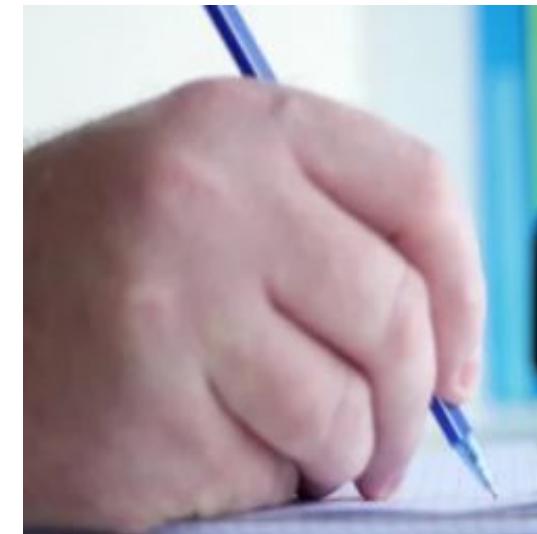
### 5. Gözden Geçiriciler(Reviewers)

- ✓ Konu uzmanları, proje üzerinde çalışan kişiler, iş ürünüğe ilgi duyan paydaşlar ve/veya belirli teknik veya iş geçmişine sahip kişiler olabilir.
- ✓ İncelenmeye olan iş ürünündeki potansiyel kusurları belirler
- ✓ Farklı bakış açılarını temsil edebilir

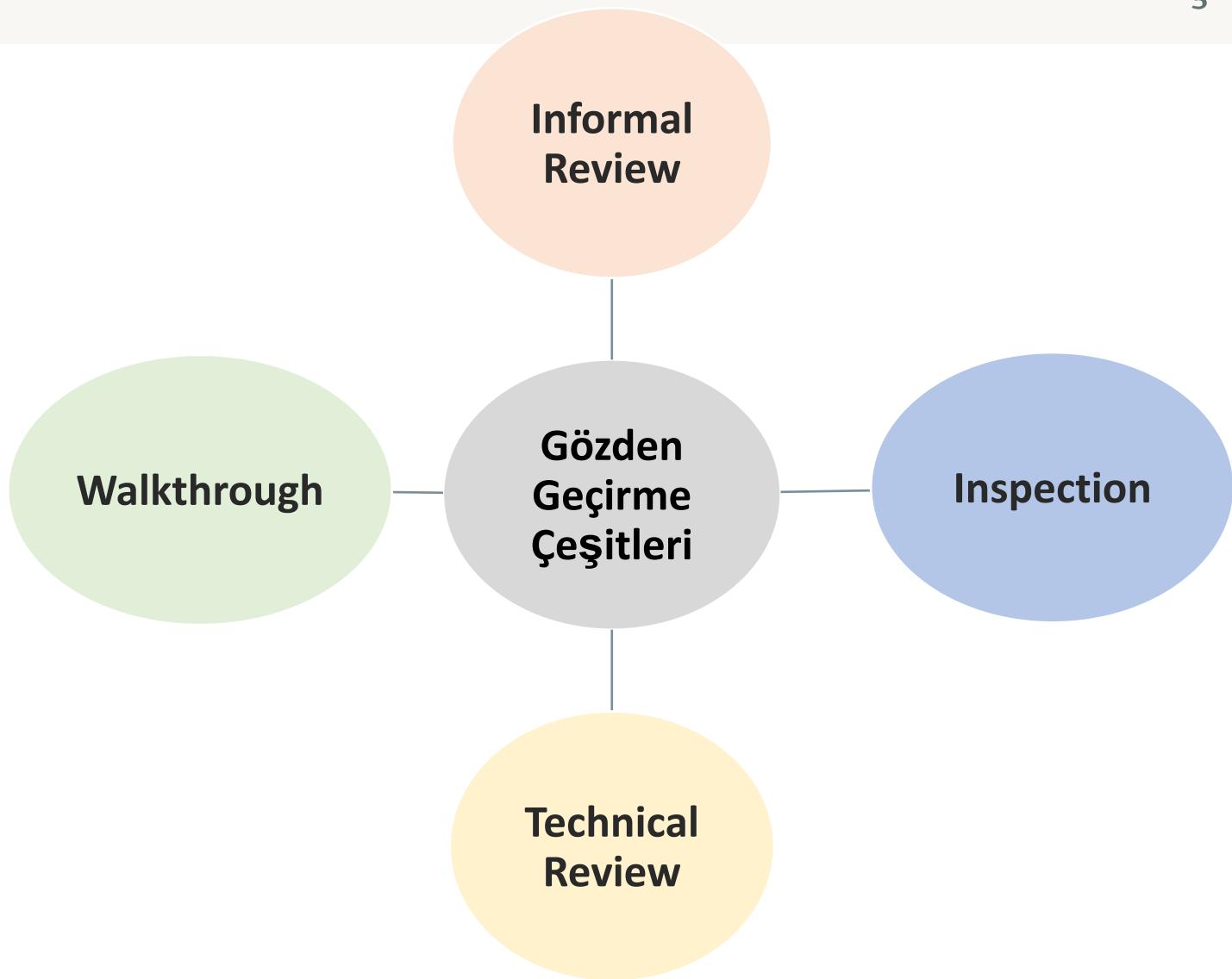


### 6. Yazıcı veya Kaydedici (Scribe or Recorder)

- ✓ Her bir gözden geçirme faaliyeti sırasında bulunan bulguları bir araya getirir ve sıraya koyar
- ✓ Gözden geçirme toplantılarında belirlenen yeni potansiyel hataları, açık noktaları ve kararları kaydeder



# Gözden Geçirme Turleri



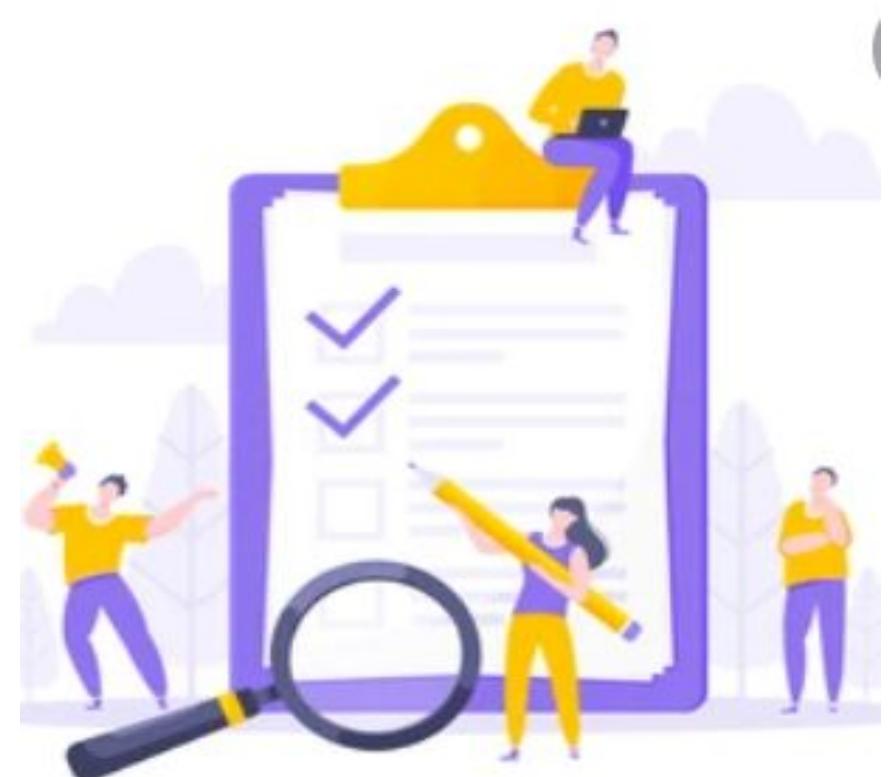
## 1. Informal Review

- 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**



## 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 değişimelidir.
- **Amaç: Öğrenme, anlama, defect bulma**



## 3.Techical Review

- Gözden geçiriciler yazarın teknik olarak dengi olan diğer veya aynı disiplinlerdeki teknik uzmanlardan seçilmelidir.
- **Amaç: Tartışma, karar verme, alternatifleri değerlendirme, defect bulma, teknik problemleri çözme ve gereksinimlere, planlara, düzenlemelere ve standartlara uyumu kontrol etme**



## 4. Inspection

- Kurallara ve kontrol listelerine dayanan resmi süreç
- Yazılım ürününün kabulu 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**



## 1. Kurgusuz (Ad hoc)

- Ad hoc incelemede, gözden geçenlere bu görevin nasıl yerine getirilmesi gerektiği konusunda çok az rehberlik sağlanır veya hiç rehberlik edilmez.
- Gözden geçenler genellikle iş ürününü sırayla okurlar, karşılaştıkları sorunları belirleyip belgelendirirler.
- Bu teknik büyük ölçüde gözden geçenin becerilerine bağlıdır ve birçok mükerrer sorunun farklı gözden geçenler tarafından raporlanması yol açabilir



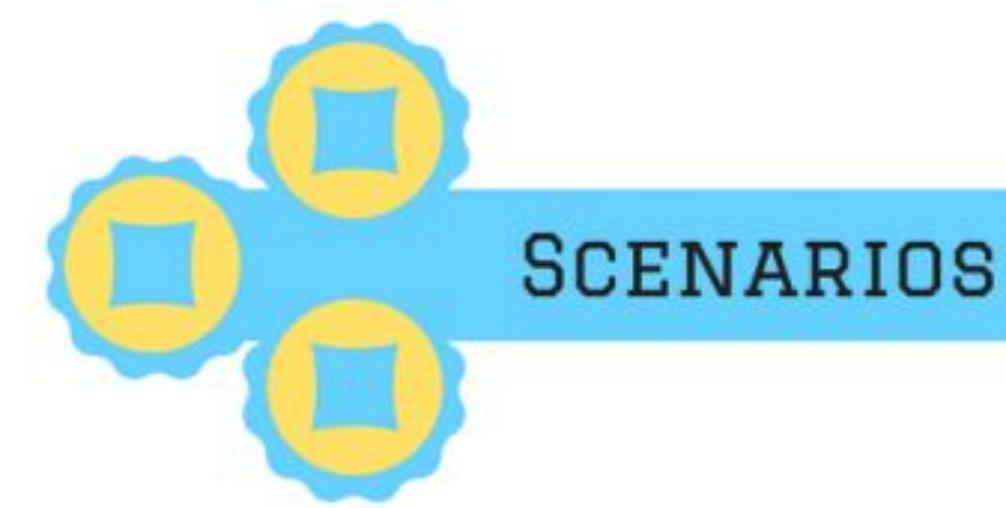
## 2. Kontrol listesine dayalı(Checklist-based)

- Gözden geçenlerin sorunları kontrol listelerine dayalı olarak tespit ettiği sistematik bir tekniktir.
- Gözden geçirme kontrol listesi, tecrübeye dayalı potansiyel hatalara iliskin bir listedir.
- Kontrol listesi tabanlı tekniğin ana avantajı, tipik kusur türlerinin sistematik olarak kapsamasıdır.
- Gözden geçen kişi, kontrol listesi dışındaki kusurları da aramalıdır.



## 3. Senaryolar ve provalar (Scenarios and dry runs)

- Gözden geçirenlere iş ürününün nasıl okunacağına ilişkin rehberler sağlanır.
- Senaryo tabanlı bir yaklaşım, çalışma ürünü üzerinde “provalar” yapılmasını sağlayarak gözden geçiricilerin işini kolaylaştırır.
- Bu senaryolar, gözden geçiricilere hata çeşitlerini bulma konusunda basit kontrol listelerinden daha iyi rehberlik sağlar.

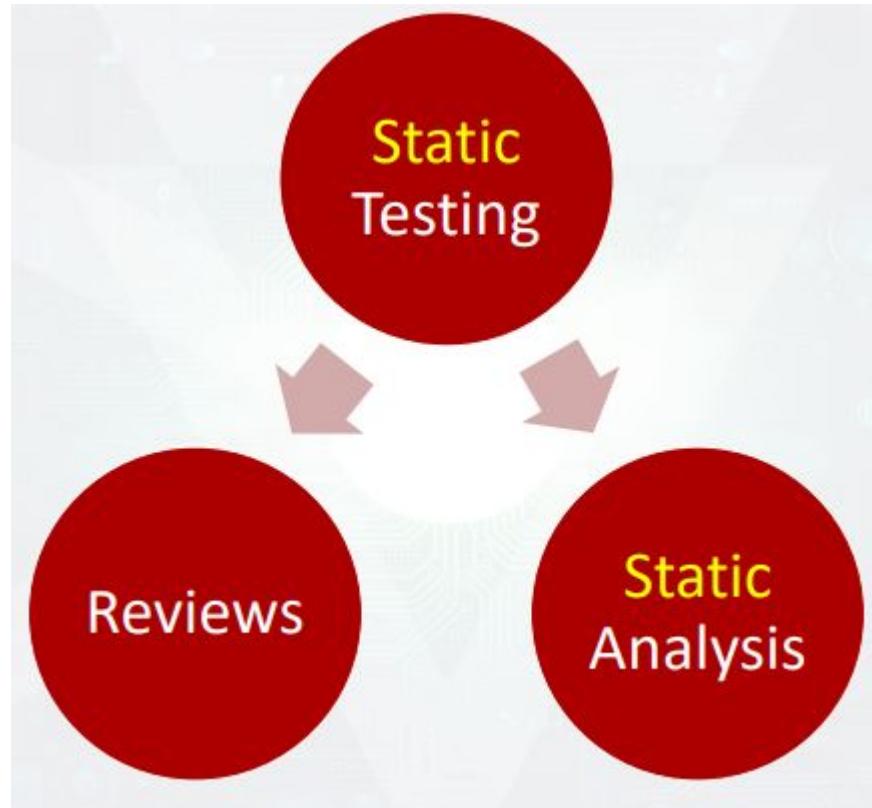


## 4. Bakış açısına dayalı(Perspective-based)

- Review esnasında farklı paydaş(Stakeholder) bakış açılarını dikkate alır.
- Farklı paydaş bakış açılarının kullanılması daha iyi neticelere ulaşılmasına ve reviewerlar tarafından benzer sorunların bulunma olasılığını azaltır.
- Bir son kullanıcı gibi düşünüp Taslak kabul testleri hazırlanır. Ayrıca bu review tekniğinde kontrol listelerinin kullanılması beklenir.
- Gereksinimleri ve teknik çalışma ürünlerini gözden geçirmek için kullanılan en etkili tekniktir



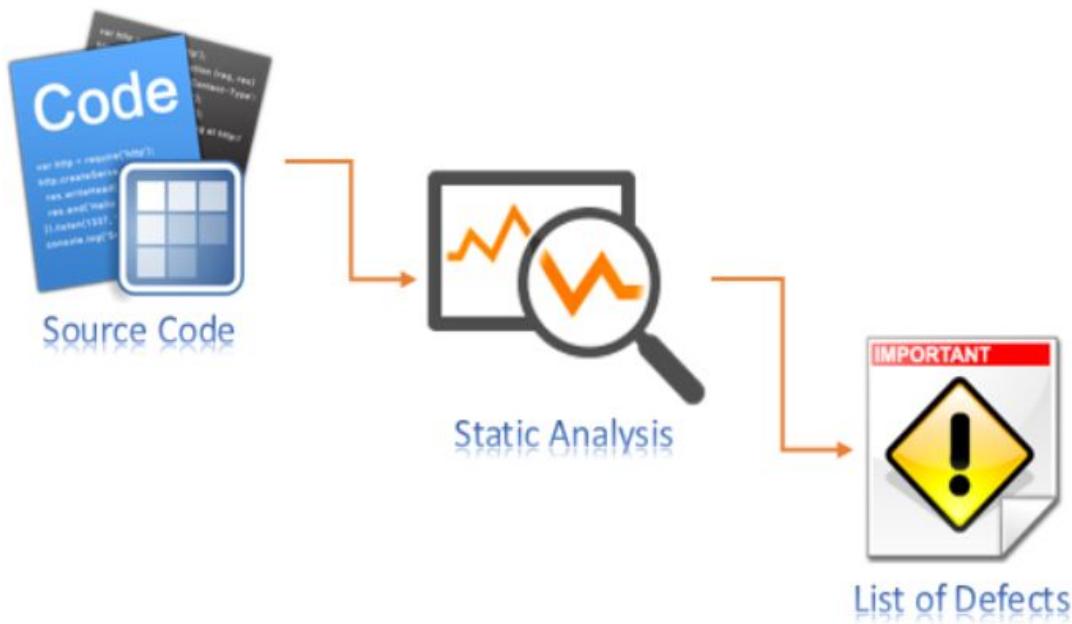
# Static Test Türleri



## 2) Statik Analiz

# Static Test Türleri

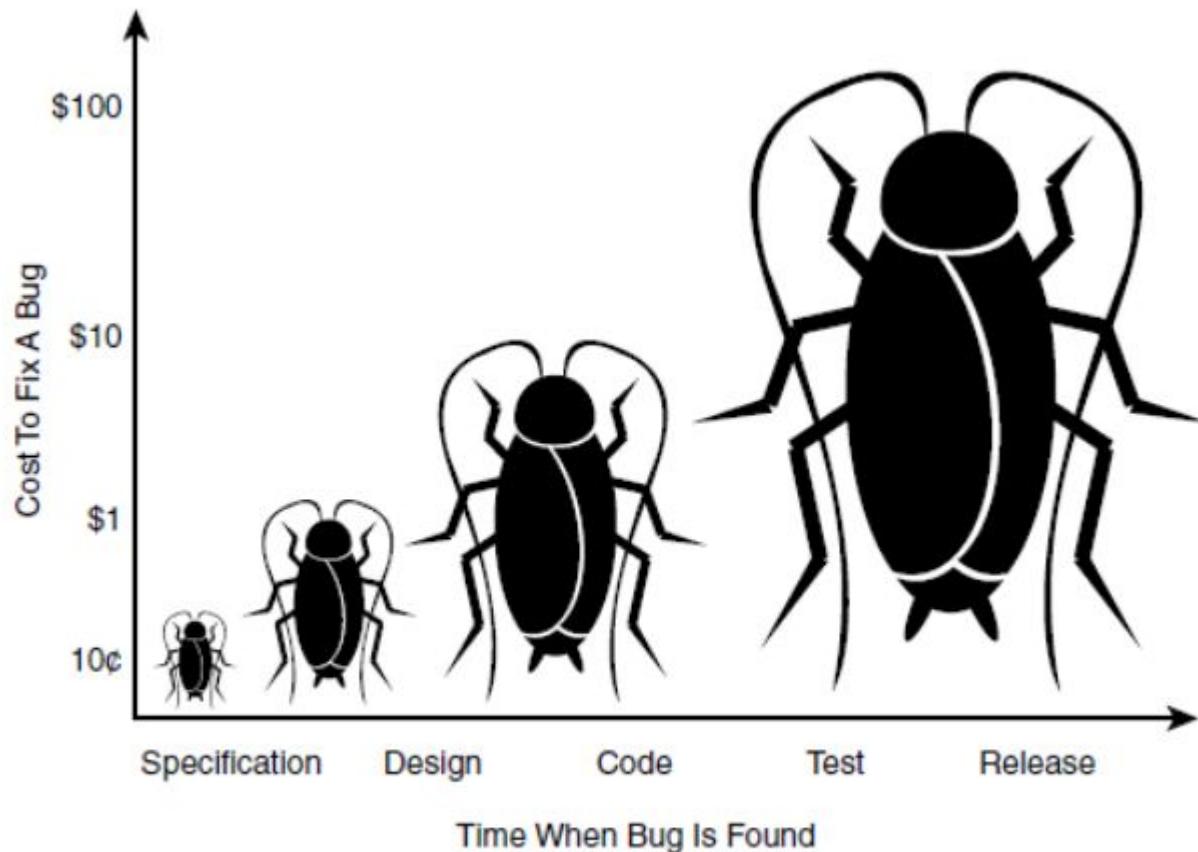
## Statik Analiz



### Statik Analizle Saptanabilecek Hatalar

- ✓ Değer atanmamış değişkenin yanlışlıkla referans gösterildiği durumlar
- ✓ Kullanılmayan değişkenler
- ✓ Tipi uygun olmayan değişkenler
- ✓ Erişilmeyen ölü kod parçaları
- ✓ Standartların göz ardı edilmesi
- ✓ Güvenlik açıkları
- ✓ Syntax hataları

# Static Testin Faydalari



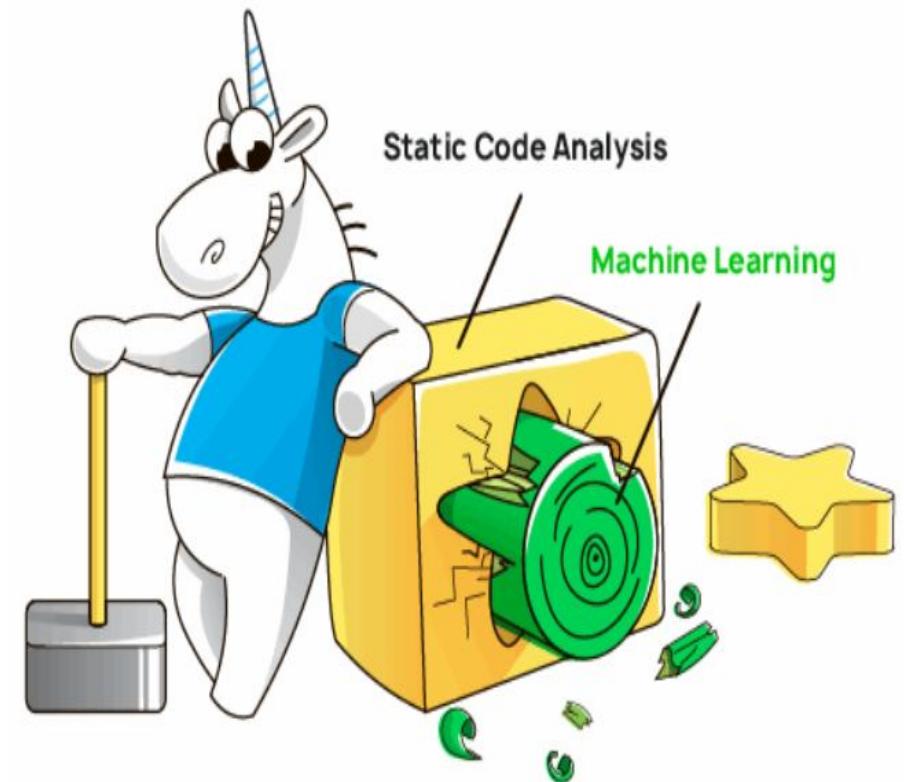
Yazılım hatasının bulunma zamanının maliyetle ilişkisi.

- 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

# Static Testin Faydalari

## Statik Testin Avantajlari

- ✓ 1. Gereksinim kusurları
- ✓ 2. Tasarım kusurları
- ✓ 3. Kodlama hataları
- ✓ 4. Standartlardan sapmalar
- ✓ 5. Yanlış arayüz özellikleri
- ✓ 6. Güvenlik açıkları
- ✓ 7. Test temelindeki boşluklar veya yanlışlıklar
- ✓ 8. Bakım Hataları





**Richmond**  
**College**  
Your Access To Future  
www.richmond.edu.tr

# **ISTQB**

## CERTIFICATION PREPARATION COURSE

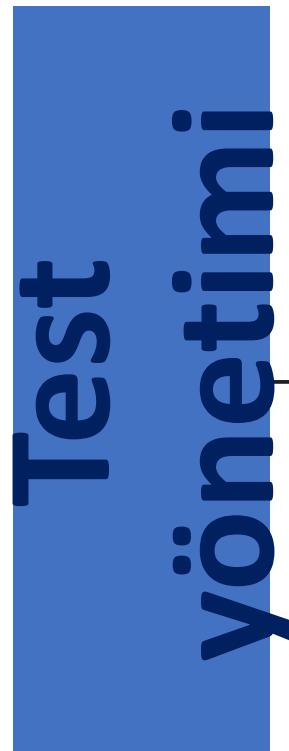
### **Test Yönetimi**

 **Richmond College**  
Advanced Education

Cooperation with  
**TECHPRO**  
EDUCATION

# Test Yönetimi

- Test Yönetimi, Yazılım test süreci ve bu süreç içerisinde yer alan kişi/kİŞİLERİN, ekibin yönetimini kapsamaktadır.



## Bağımsızlık Avantaj ve Dezavantajları



**Bağımsız test uzmanları önyargısızdır.**



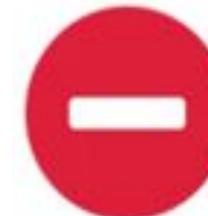
**Bağımsız test uzmanları varsayımları doğrulayabilir.**



**Yazılım geliştirme ekibinden soyutlanma**



**Yazılımcıların kalite sorumluluk hislerinin kaybolması**



**Bağımsız test uzmanlarına karşı önyargılı yaklaşım**



**Richmond**  
**College**  
Your Access To Future  
With Advanced Ed.

# **ISTQB**

## CERTIFICATION PREPARATION COURSE

**ISTQB**  
**04/08/2023**  
**Ders - 5**

## 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 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 DİŞİ  
Test Ekibi

Test Organizasyonu

Bağımsız  
Test  
Uzmanları

## Test Danışmanları



### Artıları

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

### Eksileri

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

## 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

DİŞ  
Kaynak

# 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 Organizasyonu ve Görevler

## 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 liderinin görevleri

- Test stratejisini koordine etme ve planlamak
- Test yaklaşımının diğer proje adımlarını etkilemesini sağlamak
- Testleri planlamak
- Test faaliyetlerini başlatmak, test sonuçlarını gözlemlemek ve çıkış kriterlerini kontrol etmek



# Test Organizasyonu ve Görevler

## Test liderinin görevleri

- 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 Organizasyonu ve Görevler

## 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
- Tüm test seviyelerinde testleri uyardırmak, yürütmemek ve kayıt altına almak, sonuçları değerlendirmek ve beklenen sonuçlardan sapmaları belgelemek
- Testleri otomasyona geçirmek



# Test Organizasyonu ve Görevler

## Test uzmanında bulunması gereken özellikler

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





**Richmond**  
**College**  
Your Access To Future  
www.richmond.edu

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

# Test Plan

 **Richmond College**  
Advanced Education

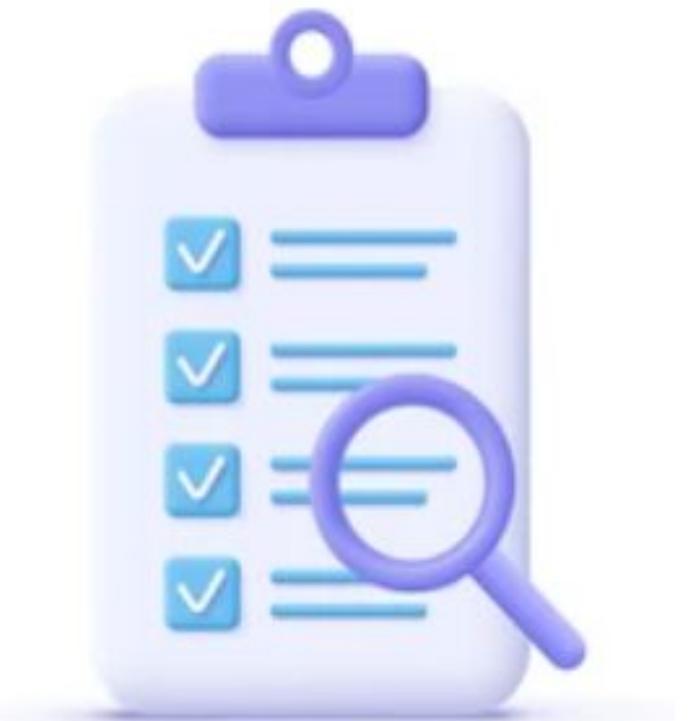
Cooperation with

 **TECHPRO**  
EDUCATION

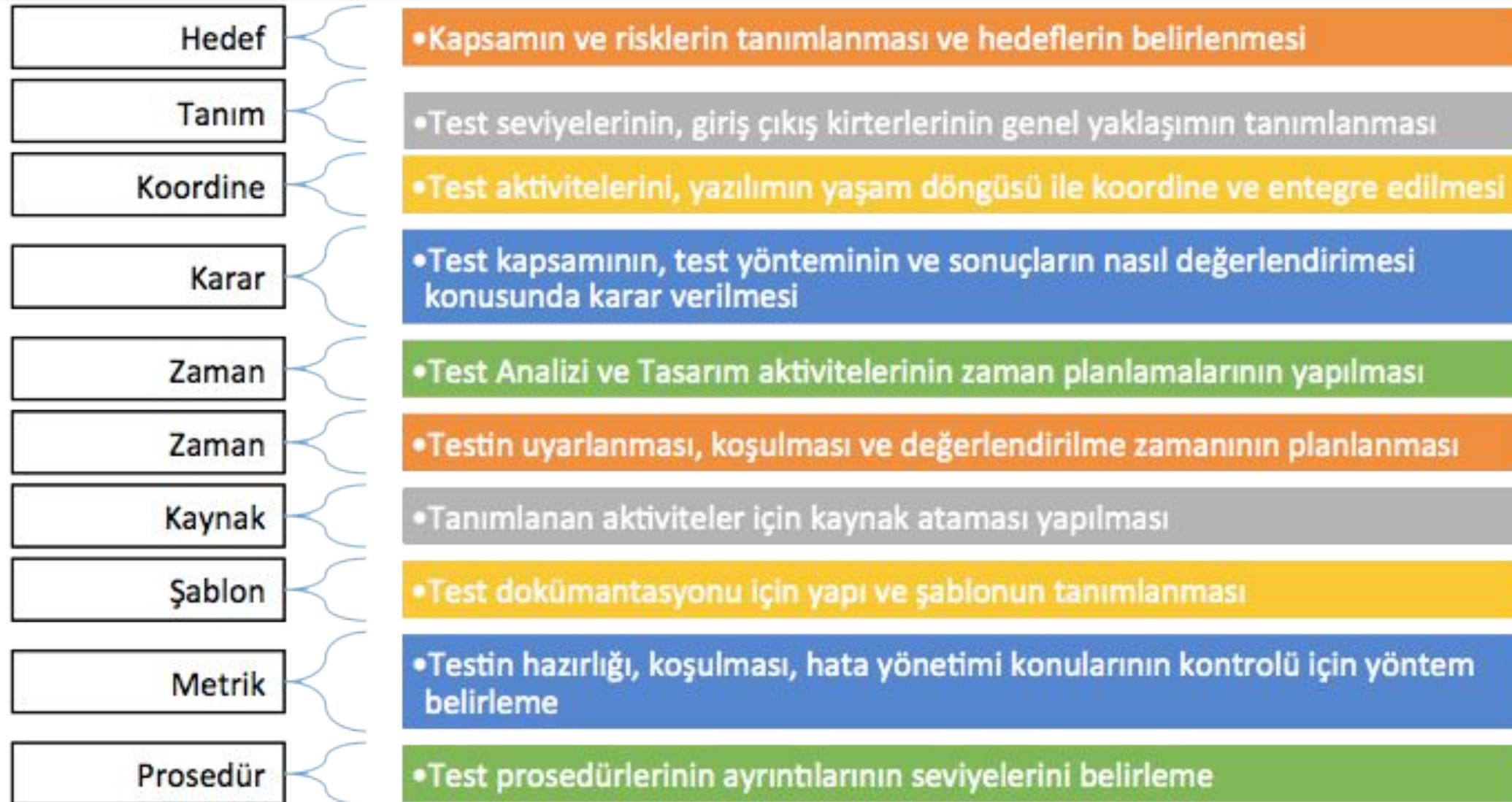
# Test Belgelendirme ve Test Planı

## İyi bir test planı sunları içermelidir;

- Hangi maddelerin test kapsamında olup olmadığı,
- Test amaçlarının ne olduğu,
- Proje ve ürün risklerinin neler olduğu,
- Proje ve ürün için en önemli olguyu,
- Nelerin test edilmeye müsait olduğu,
- Test için gerekli yazılım/donanımı,
- Zaman planı gibi bilgileri içermelidir.
- Kısa ve amaca odaklanmış olmalıdır.

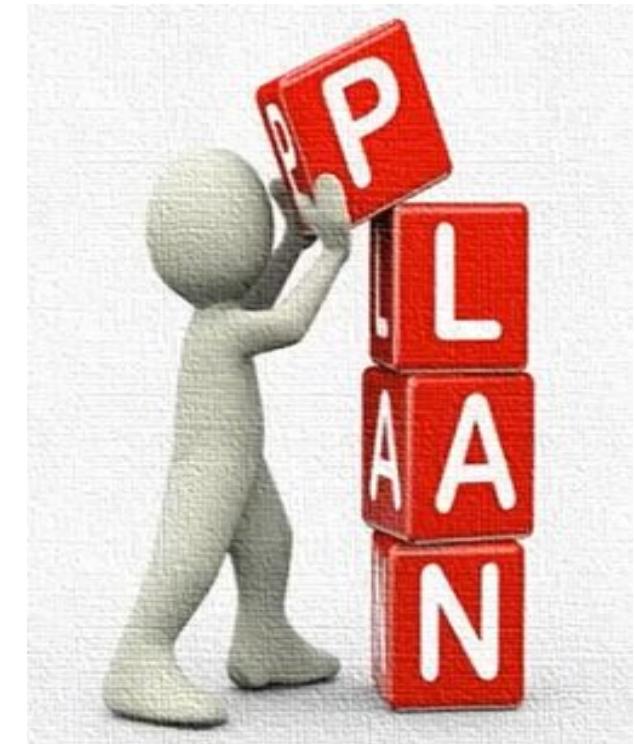


# Test Planlama Adımları



Bir Yazılım Test Planı dokümanı aşağıdaki ana başlıklardan oluşmalıdır.

- Test Stratejisi ve Test Edilecek Bileşenler
- Giriş ve Çıkış Kriterleri
- Hata Yönetimi
- Risk Yönetimi
- Görev ve Sorumluluklar
- Test Ortamı ve Test Araçları
- Test Takvimi



## Test Stratejisi ve Test Edilecek Bileşenler

Test edilecek modüller, kullanılacak test seviyeleri ve test teknikleri bu bölümde tanımlanır.

- Örnek metin aşağıdaki gibidir.
- X projesinde yazılım test faaliyetleri süresince birim, entegrasyon ve sistem testleri gerçekleştirilecektir.
- Aşağıdaki yazılım modülleri bu plan kapsamında belirtilen testlere tabi tutulacaktır.  
(A Modülü, B Modülü, C Modülü)



## Test Planlama Adımları

- **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.



# Test Planlama Adımları

## Hata Yönetimi

Hata önemine göre derecelendirme kriterleri belirlenir. Bulunan hataların hangi proje ve iş yönetimi aracında (TFS, JIRA vb.) kayıt açılacağı belirtilir. Örnek metin aşağıdaki gibidir.

X projesi kapsamında bulunan hatalarda aşağıdaki derecelendirme kriterleri kullanılacaktır:

1. Derece Hatalar: Sistemin genel çalışmasını direkt etkileyen hatalardır.
2. Derece Hatalar: Sistemin genel çalışmasını direkt etkilemeyen fakat işlevsel olarak bazı kısımların çalışmasını engelleyen hatalardır.
3. Derece Hatalar: Sistemin çalışmasını engellemeyen, görsel yönden ortaya çıkan hatalardır.

X projesi kapsamında bulunan hatalar, JIRA altında kayıt altına alınacaktır. Bu hata kayıtları önem ve öncelik derecesine göre işaretlenecektir.

## Risk Yönetimi

Test faaliyetlerinin ilerlemesini engelleyebilecek riskler aşağıdaki gibi olabilir:

- Gereksinimlerin çıkarılması geciktiğinde test adımları ve test senaryolarının yazılması gecikebilir.
- Test ekibi farklı projelerde görev yapması durumunda test faaliyetleri aksayabilir.
- Proje kapsamında geliştirilecek modüllerin zamanında tamamlanmaması sonucunda test faaliyetleri başlamayacaktır.



# Test Planlama Adımları

## Görev ve Sorumluluklar

Proje kapsamında test seviyelerine göre görev dağılımı ve sorumluluklar bu kısımda tanımlanacaktır.

## Test Seviyeleri Görev ve Sorumluluklar

Birim Testleri	Geliştirme Ekibi
Entegrasyon Testleri	Test Ekibi
Sistem Testleri	Test Ekibi
Kabul Testleri	Müşteri ve Test Ekibi

# Test Planlama Adımları

## Test Ortamı

Test Ortamı bölümü aşağıdaki şekilde yazılabilir.

X projesi kapsamında entegrasyon ve sistem test faaliyetleri 192.168.?.??. IP adresine sahip sunucu üzerinde yürütülecektir. Sunucuya ait özellikler aşağıdaki gibidir:

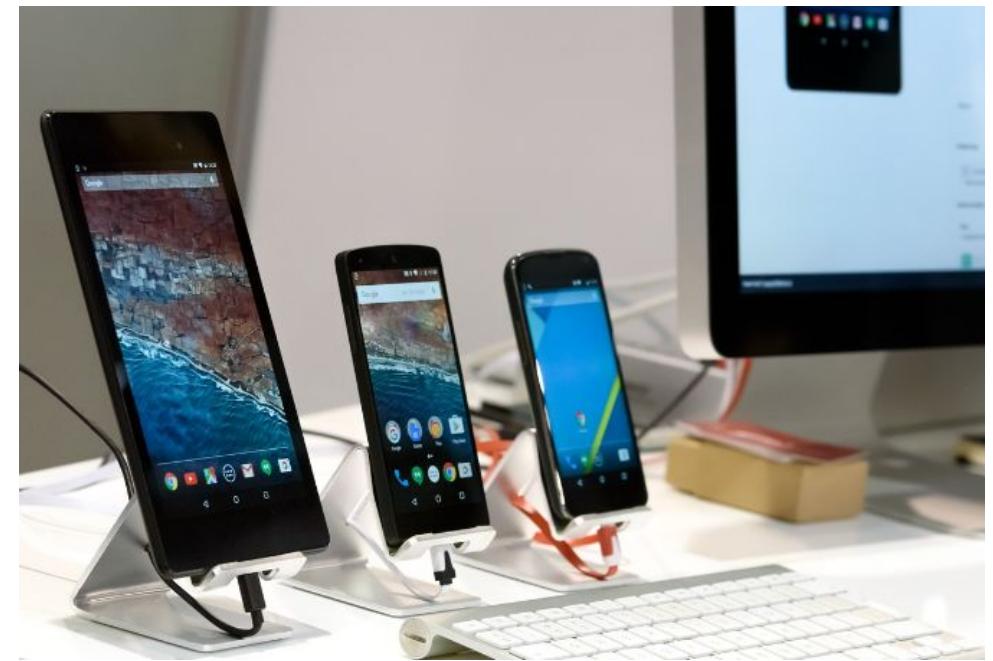
CPU: Intel Xeon 3.3 GHz

RAM: 8 GB DDR4 2133 MHz

HDD: 100 GB

OS: Windows Server 2012 R2

Yüklenecek Programlar: Java 8 Update 211 x86



## Test Araçları Versiyon Kullanım Amacı

- JUnit 4.0** Birim testleri için geliştirme ekibi tarafından JUnit kütüphanesi kullanılacaktır.
- Selenium WebDriver 4.8.3** Sistem testleri için test ekibi tarafından SeleniumWebDriver test otomasyon aracı kullanılacaktır.
- Apache JMeter 4.0** Fonksiyonel olmayan testler (Performans, Yük ve Stres) için Apache JMeter aracı kullanılacaktır.
- JIRA** Test faaliyetleri sonucunda bulunan hatalar JIRA üzerinde ilgili geliştiriciye hata kaydı olarak açılacaktır.



# Test Planlama Adımları

## Test Takvimi

Test seviyelerine göre test faaliyetlerinin başlangıç, bitiş tarihleri ve tahmini süre bilgisi belirtilmelidir. Test takvimine Proje Yönetim Planında da yer verilebilir, bu durumda Proje Yönetim planını referans olarak gösterebilirsiniz. Referans olarak göstermek istemiyorsanız proje yöneticinizle belirlediğiniz test takvimini aşağıdaki tabloda olduğu gibi gösterebilirsiniz.

Test Seviyeleri	Gün	Başlangıç Tarihi	Bitiş Tarihi
Birim Testleri	90	01.01.2019	01.04.2019
Entegrasyon Testleri	90	01.04.2019	01.07.2019
Sistem Testleri	120	01.07.2019	01.11.2019



**Richmond  
College**

Your Access To Future  
İndirimli Eğitim Fırsatları

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

# Bug Life Cycle (Hata Yönetimi)

**Richmond College**  
Advanced Education

Cooperation with

**TECHPRO**  
EDUCATION

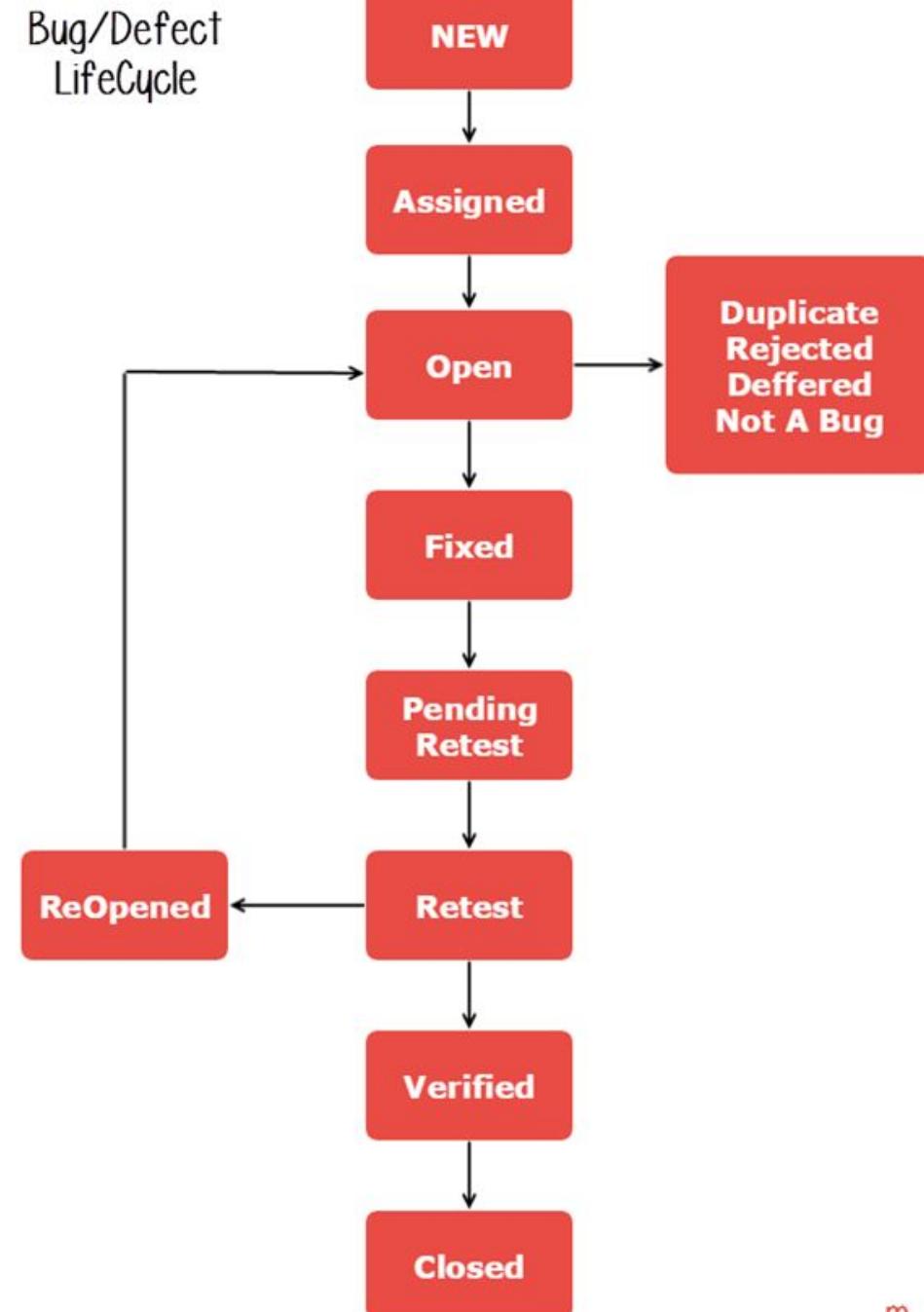
# Hata Yönetimi (Bug Life Cycle )

**New (Yeni)** : Bir hata ilk kez yayınlandığında, durum Yeni olur.  
(Jira, Alm, VersionOne gibi boardlarda..)

**Assigned (Atandi)** : Hata post edilince, test lead veya Proje lead, ilgili Developer'a "atanmış" olarak atayacaktır. (Meşru bir hata olup olmadığı ile alakalı Developerin kesinlikle bir girdisi olacaktır)

**Open (Acildi)** : Developer bu aşamada düzeltmek için hatayı analiz eder ve üzerinde çalışır.

**Fixed (Onarildi)** : Developer gerekli kod güncellemlerini yaptığında ve hatanın düzelttilip düzeltildmediğini doğruladığında, hata durumunu "Fixed" olarak değiştirebilir ve hata test ekibine aktarılır



# Hata Yönetimi (Bug Life Cycle )

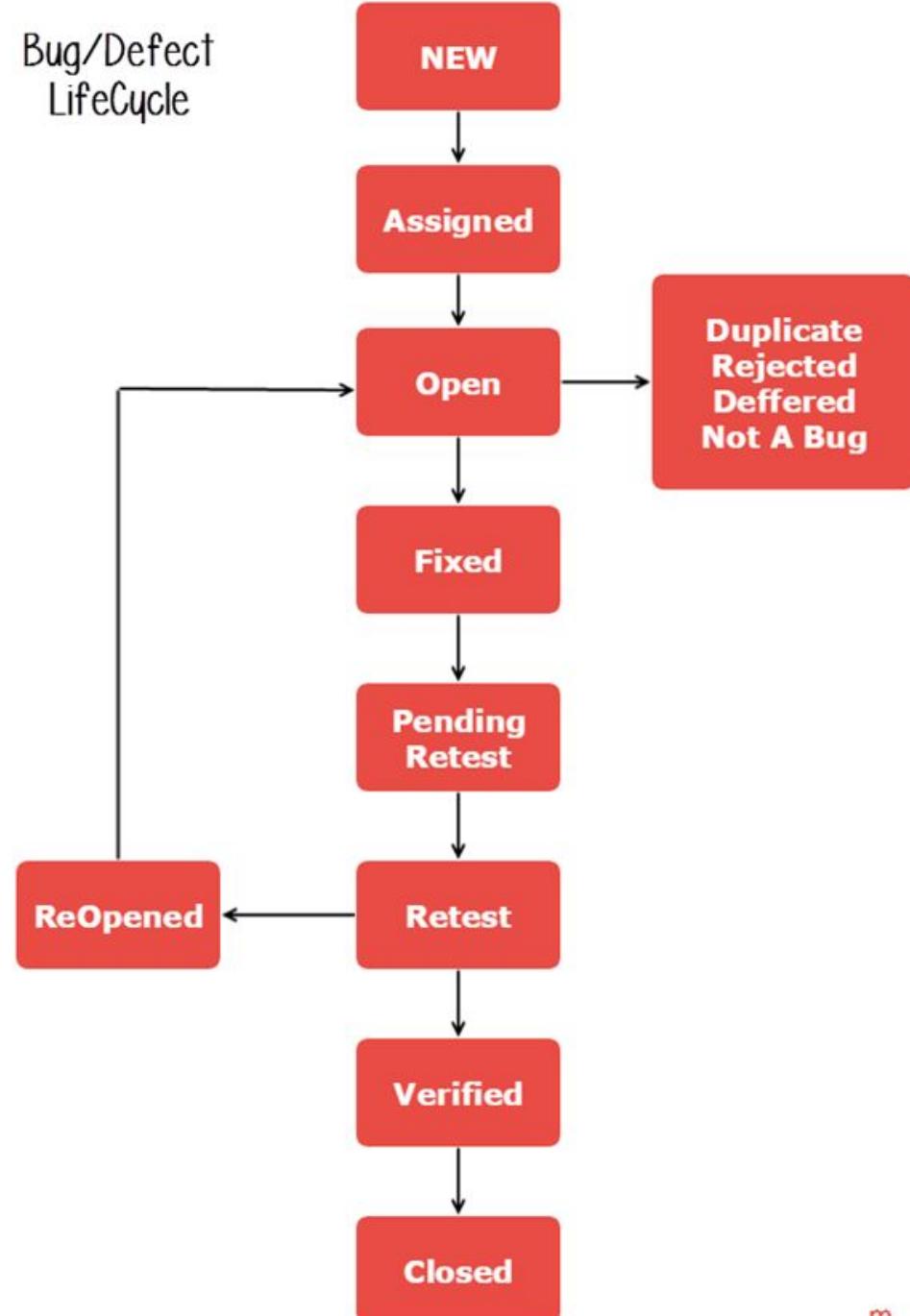
**Pending retest (Tekrar Test İcin Bekliyor)** : Arızayı giderdikten sonra Developer, yeniden test için bu özel functionalityi deploy eder. Burada testler test aşamasında beklemeye olur. Ve durum tekrar test etmeyi gerektirir.

**Retest (Tekrar Test)** : Bu aşamada, tester işlevselligi farklı test verileriyle tekrar test eder ve buldukları defect ilgili her şeyi kapsadığından emin olur.

**Verified (Onaylandı)** : Tester doğrulama yaptıktan ve hata yazılımda görünmediğinde, hatanın düzeltildiğini onaylar ve durumu "Doğrulandı" olarak değiştirir.

**Reopen(Tekrar Acıldı)** : Hata Developer tarafından düzeltildikten sonra bile hata hala mevcutsa, tester durumu "Yeniden Açıldı" olarak yapar. Bu yüzden hata aynı süreçten bir kez daha geçer.

**Closed(Kapandı)** : Hata giderildikten sonra tester tarafından test edilir. Tester yazılımda defect bulunmadığından emin olursa, hatanın durumunu "Kapalı" olarak değiştirir. Bu, hatanın düzeltildiği, test edildiği ve onaylandığı anlamına gelir.



# Hata Yönetimi (Bug Life Cycle )

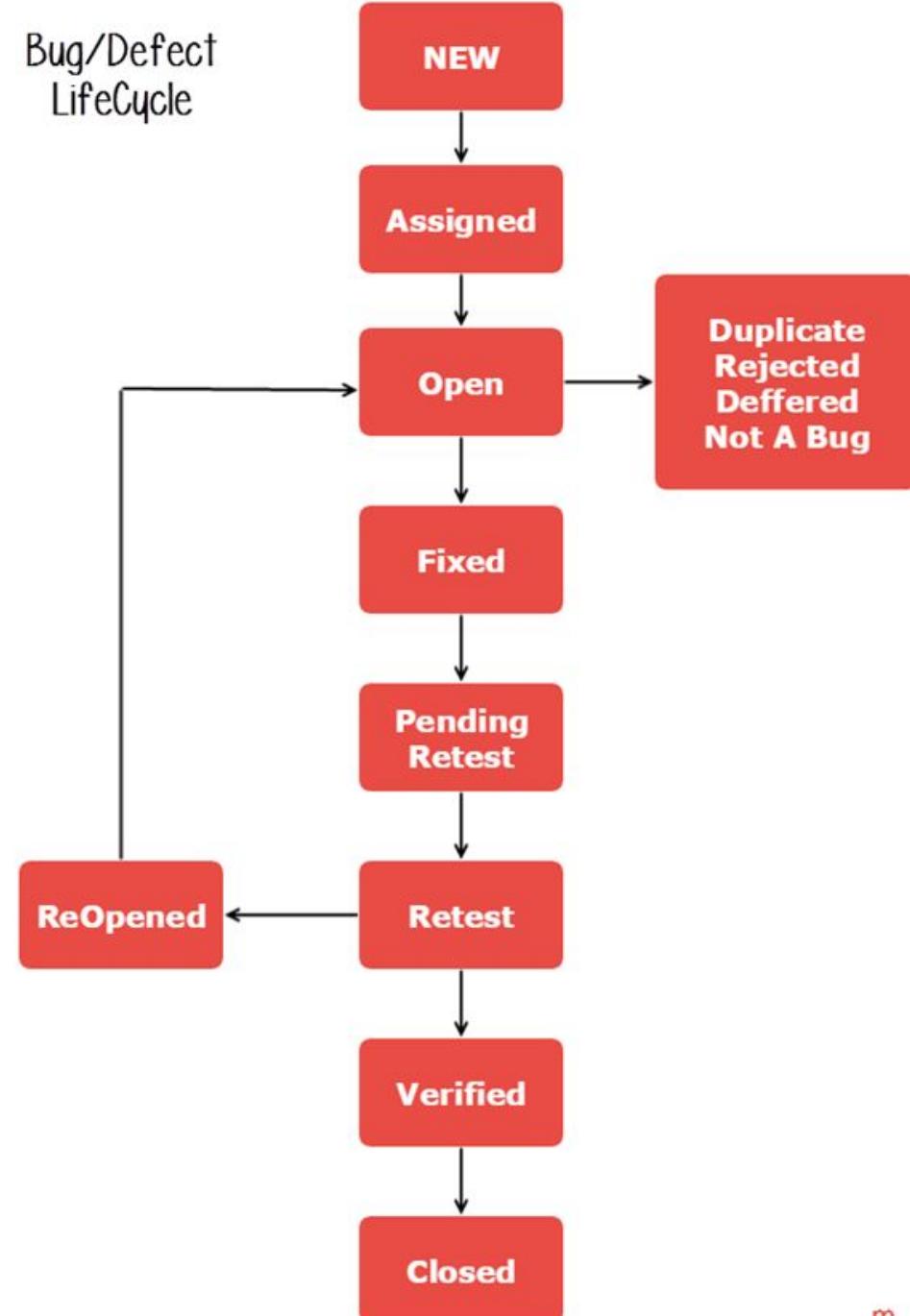
Bug Ticket olusturuldugunda Developer'ların cevapları

**Rejected (Reddedildi)** : Kusur Developer tarafından meşru bir kusur olarak kabul edilmezse Developer tarafından 'Reddedildi' olarak işaretlenir.

**Duplicate (Tekrar)** : Developer hatayı diğer diger defectlerle aynı bulursa, hata durumu Developer tarafından "cift" olarak değiştirilir.

**Deferred (Ertelendi)** : Developer, hatanın çok önemli bir öncelik olmadığını ve sonraki sürümlerde giderileceğini düşünüyorsa, hatanın durumunu 'Ertelenmiş' olarak değiştirebilir.

**Not a Bug (Bug Degil)** : Kusurun uygulamanın işlevselligi üzerinde bir etkisi yoksa, kusurun durumu 'Hata Değil' olarak değiştirilir.



## Bug bulduğumuzda ne yapmalıyız?

- ❑ Bir bug bulduğumuzda öncelikle bunun bug olduğundan emin olmamız gerekiyor.
- ❑ Bunun için testimizi tekrarlamalı, gerekirse başka tester var ise onun fikrini almamızda fayda var. Test Leader bilgi verilebilir.
- ❑ Eğer developerlar ile aynı ortamda çalışılıyorsa, developer durumdan haberdar edilebilir. Developer bunun bug olmadığını, bug olduğunu ya da önemsiz bir durum olduğunu ifade edebilir.
- ❑ Developerin dönüşüne göre hareket edilir.



# Hata Yönetimi (Bug Life Cycle )

## Bug Raporu (Bug Report)

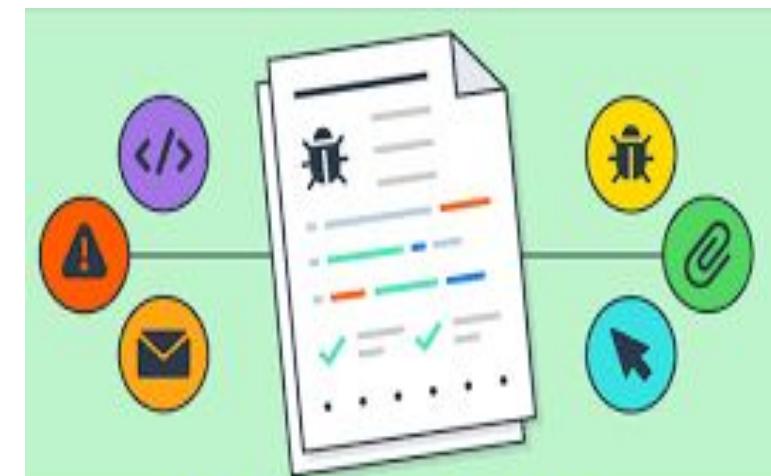
**Başlık (Title) :** Hata başlığı diğer hatalarda ayırt etmek için yazılır. Hata başlığı kisa ve öz olmalıdır. Örnek vermek gerekirse “*Kullanıcı tanımlama ekranında kaydet butonuna basıldığında hata veriyor.*”

**Atama (Assigned To) :** Bulunan hatanın hangi geliştiriciye veya takım liderine atanacağı bu alandan seçilmelidir.

**Öncelik (Priority) :** **Yazılım Test Planı**’da hatalar önceliklerine göre derecelendirilmelidir. Bulunan hataların öncelik değerleri plana göre doğru şekilde girilmelidir.

**Önem Derecesi (Severity) :** Bulunan hatalar önem derecesine göre gruplandırılmalıdır. İlgili geliştirici hataları önem derecesine göre çözmeli dir.

**Hata Kaynağı (Root Cause) :** Bulunan hatanın kaynağı belirtilmelidir. Bu bilgi test faaliyetleri sonucunda ölçüm ve analiz için kullanılabilir.



# Hata Yönetimi (Bug Life Cycle )

## Bug Raporu (Bug Report)

**Adımlar (Repro Steps) :** Bulunan hatanın ilgili geliştirici tarafından tekrar simüle edilebilmesi için hatanın adım adım yazılması gerekmektedir. Bunun için aşağıdaki örneği verebiliriz:

- a. Kullanıcı tanımlama sayfasına gidilir.
  - b. Ekranda yer alan tüm zorunlu alanlar doldurulur.
  - c. Kaydet komutu verilir.
- o **Beklenen Sonuç: Kullanıcı tanımlama işleminin başarıyla tamamlandığına dair bilgilendirme mesajı verilmesi gereklidir.**
  - o **Gerçekleşen Sonuç: NullPointerException hata mesajı alındı.**

**Ekler (Attachments) :** Bulunan hatanın ekran görüntüsünü veya video kaydını alarak bu alana yüklenmesi test ekibinin hatayı geliştirme ekibine daha kolay aktarılmasını sağlayacaktır.

A	B	C
1	<b>Category</b>	<b>Label</b>
2	Bug ID	ID number
3		Name
4		Reporter
5		Submit Date
6	Bug overview	When my cart contains one item, I am unable to add a second item via the add to cart button on a product page
7		Summary
8		URL
9	Environment	Screenshot
10		Platform
11		Operating System
12	Bug details	Browser
13		add one item to cart > go to product abc via the search bar > add new item to cart via "add to cart" button (see screenshot) > go to cart
14		Steps to reproduce
15		Expected result
16	Bug tracking	Actual result
17		Description
18		/
19	Severity	Major
	Assigned to	/
	Priority	High
	Notes	Notes
		/

# Hata Yonetimi (Bug Life Cycle )

Bug Tracker All changes saved.

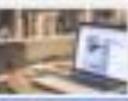
Bugs File Edit View Insert Format Share

A - BUG ID (Autogenerated) B - Type C - Found In D - Feature E - Description F - Priority G - Screenshots & Attachments

**Bug Tracking**

Track bugs and enhancements, with attached screenshots and mockups, status, owners, release assignments, and more.

Use CTRL+V to copy screenshots from your clipboard into Attachment cells.

	A - BUG ID (Autogenerated)	B - Type	C - Found In	D - Feature	E - Description	F - Priority	G - Screenshots & Attachments
1	<b>Bug Tracking</b>						
2	Track bugs and enhancements, with attached screenshots and mockups, status, owners, release assignments, and more.						
3	Use CTRL+V to copy screenshots from your clipboard into Attachment cells.						
4							
5	BUG ID (Autogenerated)	Type	Found In	Feature	Description	Priority	Screenshots & Attachments
6	BUG-000001	Bug	Production	User Profile	When user uploads a photo, placeholder image doesn't get replaced until the user refreshes. This should happen right away as soon as the upload is completed.	High	
7	BUG-000002	Bug	Stage 1	Marketplace	Marketplace listings are not appearing in reverse chronological order by default as they should be	Critical	
8	BUG-000003	Bug	Local	Registration	Pinch gestures not working in search results pages on Android	Medium	
9	BUG-000004	Investigation	Production	Billing	Credit card approval notifications delayed by longer than expected	Critical - High Priority	
10	BUG-000005	Regression	Stage 2	Page Editor	Line breaks lost in embedded code	High	
11	BUG-000006	Bug	Production	User Profile	Settings option shows as highlighted even when mouse is not hovered over it	Medium	
12							



**Richmond**  
**College**  
Your Access To Future  
www.richmond.edu.tr

**ISTQB**  
**CERTIFICATION**  
**PREPARATION**  
**COURSE**

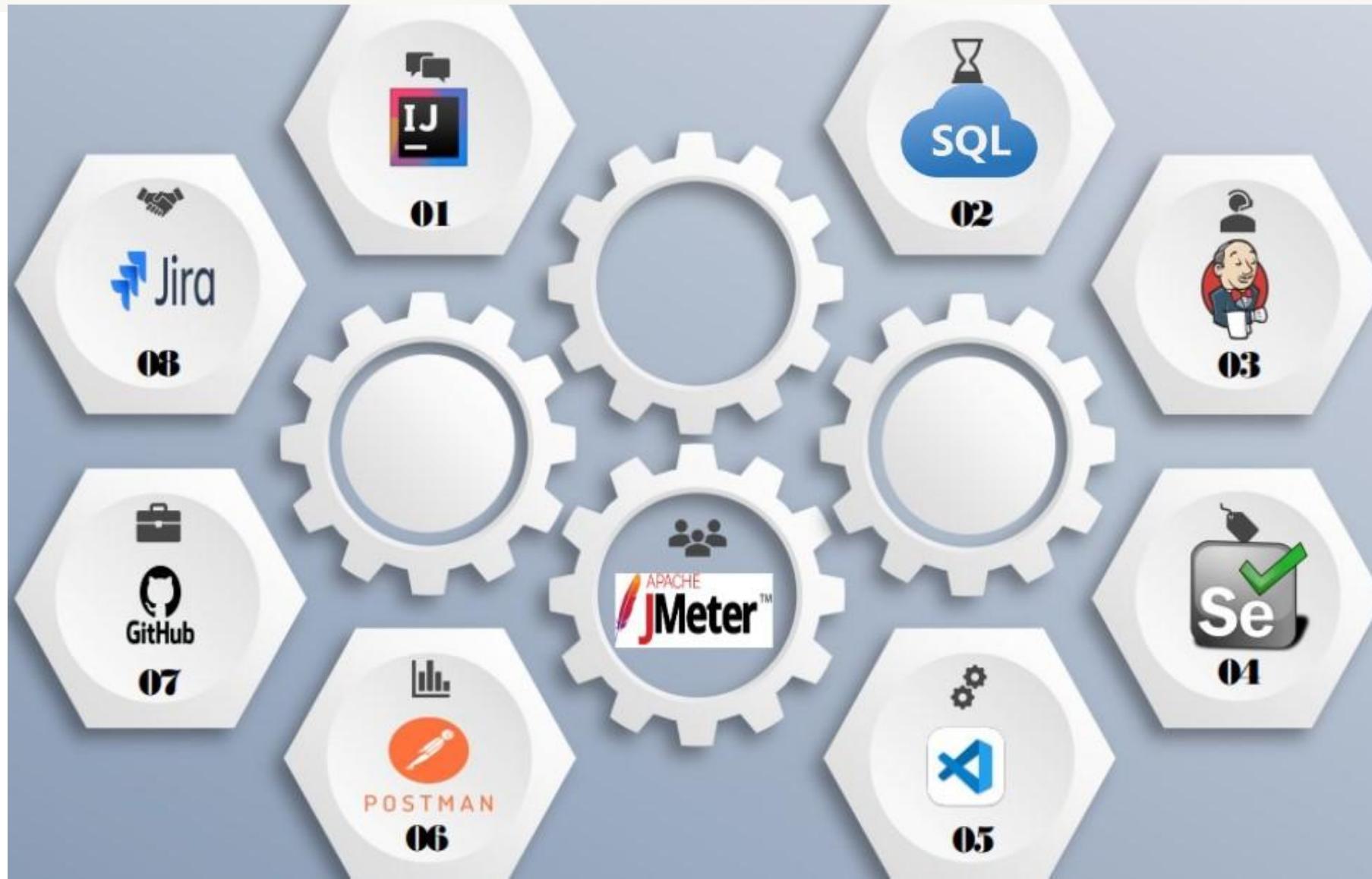
# Test Araçları

 **Richmond College**  
Advanced Education

Cooperation with

 **TECHPRO**  
EDUCATION

# Test Aracları



## Test Aracı Sınıflandırılması

Araçlar çeşitli kriterlere göre sınıflandırılabilir:

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



## Test Yönetimi ve Testler için Araç Desteği

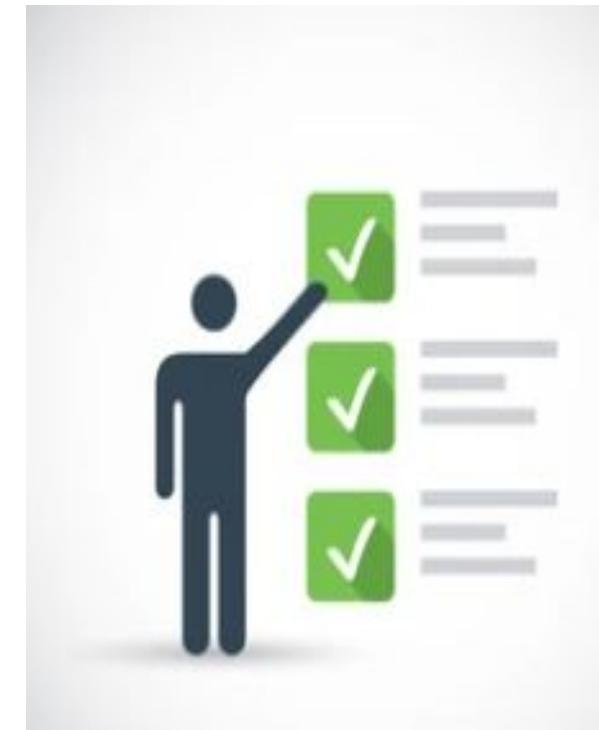
Yönetim araçları tüm yazılım yaşam döngüsü boyunca tüm test aktivitelerine uygulanır

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



## 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.



## 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 Yürütmeye Araçları

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

## Test Kuluçkası / Birim Testi Çerçeve 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 Aracları Avantajları

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

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



# Test Aracları Dezavantajları



- 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

## 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





**Richmond**  
**College**  
Your Access To Future  
www.richmond.edu.tr

# **ISTQB**

## CERTIFICATION PREPARATION COURSE

**Tesekkürler...**

 **Richmond College**  
Advanced Education

Cooperation with  
**TECHPRO**  
EDUCATION