**Sertifikalı Test Uzmanı**

**Temel Seviye Ders Programı(Versiyon 2018)**  
**Ders Notları**

Salih Topal

Önsöz  
  
Hazırlanan ders notları ISTQB® Sertifikalı Test Uzmanı Temel Seviye Ders Programının 2018 Versiyonu çalışmaları sonrasında oluşturulmuştur. Bu notların amacı yoğun bilgi barındıran bir dokümanı daha az sayfada ve dikkat çekici bir şekilde oluşturma gayretidir.   
  
Ders notları hazırlanırken daha önce çıkmış bazı dönem soruları ve örnek sorular incelenerek önemli kısımlar atlanmamaya çalışılmıştır. Yine de ders notlarının geliştirilmesi amacıyla görüş ve düşüncelerinizi mail adresine paylaşarak bu ders notlarının geliştirilmesine yardımcı olabilirsiniz.  
  
Daha ayrıntılı bir şekilde sınava hazırlanmak için ders notları sonrası https://www.turkishtestingboard.org/ adresinde yer alan yayınlardan Sertifikalı Test Uzmanı Temel Seviye Ders Programı ve ISTQB Yazılım Testi Terimler Sözlüğü tercih edilebilir.   
  
Ders notları temel seviye ders programındaki gibi aşağıdaki bölümlerden oluşmaktadır:  
  
Yazılım Testinin Temelleri - Yazılım Geliştirme Yaşam Döngüsü Boyunca Testler - Statik Testler - Test Teknikleri - Test Yönetimi - Yazılım Testleri için Araç Desteği  
  
Hazırlanan bu ders notlarının kaynağı ISTQB'nin bu ders programının kaynağı ISTQB® Sertifikalı Test Uzmanı Temel Seviye Ders Programının 2018 Versiyonudur.

**Antibiyotik direnci**

Aynı testler sürekli tekrar edilirse, en sonunda bu testler artık yeni hatalar bulamamaya başlar. Yeni hataları bulmak için mevcut testler ve test verilerinin değiştirilmesi ve yeni testlerin yazılması gerekebilir (Tıpkı sürekli antibiyotik kullanımında bir süre sonra aynı çeşit antibiyotiğin artık bakterileri öldürmede etkili olmaması, bakterilerin direnç kazanması gibi).

**İnsanların yaptığı hataların nedenleri:**

• Zaman baskısı

• İnsani yanılma payı

• Deneyimsizlik veya yetersiz yetkinlik

• Gereksinimler ve tasarım ile ilgili yanlış iletişim de dâhil olmak üzere proje ekip üyeleri arasındaki yanlış iletişim

• Kodun, tasarımın, mimarinin, çözülecek temel problemin ve/veya kullanılan teknolojilerin karmaşıklığı

• Özellikle sistem içi ve sistemler arası etkileşimlerin sayısının fazla olduğu durumlarda, sistem içi ve sistemler arası arayüzler hakkındaki yanlış anlaşılmalar

• Yeni, henüz tecrübe edilmemiş teknolojiler

**Hataların kök nedenleri,** hataların oluşmasına sebep olan en baştaki eylemler veya koşullardır. Hatalar, kök nedenlerini belirlemek için analiz edilebilir, böylece gelecekte benzer hataların ortaya çıkma olasılığının azaltılması hedeflenir.

**Test Prensipleri**  
1. Testin amacı, yazılımda hataların olduğunu göstermektir; yazılımda hata kalmadığını ispatlamak değildir.  
2. Yazılımı %100 test etmek imkansızdır.  
3. Erken test, zaman ve para tasarrufu sağlar.  
4. Hatalar yazılımın belli alanlarında yoğunlaşır.  
5. Antibiyotik direnci  
6. Yazılım testi, projenin bağlamına, koşullarına göre değişiklik gösterir.  
7. Yeni hata bulamıyoruz, başarılı bir yazılım elde ettik yanılgısı

**1.YAZILIM TESTİNİN TEMELLERİ**

|  |
| --- |
| Yazılım Testi nedir? |
| 1- Yazılım testi, yazılımın kalitesini değerlendirmenin ve kullanım sırasında oluşabilecek |
| yazılım hatası riskini azaltmanın bir yoludur. |
| 2- Yazılım testi birçok farklı faaliyeti içeren bir süreçtir. |
| 3- Test süreci aynı zamanda test planlama, test analizi, test tasarımı, testin uyarlanması, |
| test ilerlemesini ve sonuçlarını raporlama ve bir test nesnesinin kalitesini değerlendirme gibi faaliyetleri içerir |
|  |
| Test Etme ve Hata Ayıklama |
| •Test etme ve hata ayıklama farklı aktivitelerdir. |
| •Testlerin yürütülmesi, yazılımdaki hataların neden olduğu arızaları gösterebilir. |
| •Hata ayıklama ise arızaların arkasında yatan hataları bulan, analiz eden ve |
| düzelten yazılım geliştirme faaliyetidir. |

**Error(İnsan Hataları),** bir insan tarafından gercekleştirilen ve doğru olmayan sonuç üreten bir eylem.

**Defects(Hata-Bug),** bir bileşen ya da sistemin gerekli işlevini gerçekleştirmesini engelleyen kusur. (örn. Doğru olmayan komut veya veri tanımlaması).

**Failure(Arıza),** bileşen veya sistemin, beklenen teslim, servis veya sonuçtan sapması. Defects(Hata-Bug), bileşen ya da sistem çalışırken ortaya cıkarsa failure(arızaya) neden olabilir.

|  |
| --- |
| Test Süreci Ana Aktivite Grupları |
| \*Test planlaması, test hedeflerini yerine getirme yaklaşımıdır. |
| \*Test gözetimi, planlanan ile gerçekleşen ilerlemenin sürekli karşılaştırılmasını içerir. |
| \*Test kontrolü, test planında belirlenen hedeflere ulaşmak için gerekli önlemlerin alınmasını içerir. |
| \*Test analizi, ölçülebilir kapsama kriterleri açısından “neyin test edileceğini” belirler. |
| \*Test tasarımı, “nasıl test edilecek?” sorusuna cevap verir. |
| \*Test uyarlama, “şu anda testleri koşmak için gerekli şeylere sahip miyiz?” sorusuna cevap arar. |
| \*Test koşumu, sırasında, test grupları test koşum çizelgesine uygun olarak çalıştırılır. |
| \*Test tamamlama aktiviteleri, bir yazılımın piyasaya sürülmesi, bir test projesinin |
| tamamlanması, bir çevik proje döngüsünün tamamlanması, |
| bir test seviyesinin tamamlanması veya bir bakım sürümünün tamamlanması gibi proje |
| kilometre taşlarında gerçekleştirilir. |

**İnsan Psikolojisi ve Test**

Hataların ve arızaların bulunması, yazılım ürününün ve yazılım ürününü geliştirenin eleştirilmesi olarak algılanabilir.  
Psikolojide yer alan **“doğrulama sapması”** tezine göre, mevcut görüş ve inançlara uymayan bilgilerin kabul edilmesi kolay değildir.

Doğrulama sapmasının yanında diğer bilişsel eğilimler insanların testlerde elde edilen bilgileri anlamalarını veya kabul etmelerini zorlaştırabilir.   
  
**Test uzmanlarının iyi iletişim kurmasına aşağıdaki örnekler verilebilir:**

**\*** Savaşarak değil iş birliği yaparak başlayın.   
**\*** Herkese daha iyi kalitede yazılımlar geliştirmek olan ortak amacınızı hatırlatın.  
**\*** Testin faydalarını vurgulayın.   
**\*** Hatalı öğeyi oluşturan kişiyi eleştirmeden, test sonuçlarını ve diğer bulguları tarafsız ve gerçekçi bir şekilde iletin.  
**\*** Objektif ve gerçeklere dayalı hata raporları yazın ve bulguları gözden geçirin.  
**\*** Karşınızdaki kişinin nasıl hissettiğini ve verdiğiniz bilgilere olumsuz tepki vermesinin sebeplerini anlamaya çalışın.  
\* Karşınızdaki kişinin söylediğiniz şeyi anladığını ve kendinizin de onun söylediği şeyi anladığınızı doğrulayın.

Şekil- Test Süreci Ana Aktivite Grupları

**Test Esası ve Test Çalışma Ürünleri Arasında İzlenebilirlik**

**İyi bir izlenebilirlik aşağıdakileri de destekler:**  
  
\* Test kapsamının değerlendirilmesine

\* Değişikliklerin etkisinin analiz edilmesi

\* Testlerin denetlenebilir şekilde yapılması

\* BT yönetişim kriterlerinin karşılanması

\* Test esasındaki unsurların durumunu (ör. testleri geçen gereksinimler, testlerde başarısız olan gereksinimler ve bekleyen testleri olan gereksinimler) içerecek şekilde test ilerleme raporlarının ve test özet raporlarının anlaşılabilirliğinin artırılması

\* Testlerin teknik yönlerinin, anlaşılır şekilde paydaşlara aktarılması

\* İş hedeflerine göre ürün kalitesinin, süreç kapasitesinin ve proje ilerlemesinin değerlendirilmesi için bilgi sağlanması

Bir yazılımcının düşünce tarzı, bir test uzmanının düşünce tarzının bazı unsurlarını içerebilir, ancak yazılımcılar, genellikle çözümler tasarlamak ve oluşturmakla ilgilenir, bu çözümlerde neyin yanlış olabileceğini fazla düşünmez. Ek olarak, psikolojideki doğrulama sapması da kendi çalışmalarındaki hataları bulmalarını zorlaştırır.

Bir test uzmanının düşünce tarzı merak, profesyonel karamsarlık, eleştirel bir bakış açısı, ayrıntılara dikkat, iyi ve pozitif iletişim ve ilişkiler için motivasyon içermelidir. Bir test uzmanının düşünce tarzı, test uzmanı deneyim kazandıkça gelişme ve olgunlaşma eğilimindedir.

**2.YAZILIM YAŞAM DÖNGÜSÜ BOYUNCA TEST**

Bir yazılım geliştirme yaşam döngüsü modeli, yazılım geliştirme projesinin her aşamasında gerçekleştirilen aktivite çeşitlerini ve aktivitelerin birbirleriyle mantıksal ve kronolojik olarak nasıl ilişkili olduğunu açıklar.

**2.1.1 Yazılım Geliştirme ve Yazılım Testi**  
  
Bu dokümana göre yaygın yazılım geliştirme yaşam döngüsü modelleri aşağıdaki sınıflara ayrılır:  
   
**• Sıralı yazılım geliştirme modelleri(Doğrusal, sıralı bir faaliyetler akışı)**

**• Döngüsel ve artımlı yazılım geliştirme modelleri**

**A) Şelale modelinde,** yazılım geliştirme faaliyetleri (örneğin; gereksinim analizi, tasarım, kodlama, testler) birbiri ardına tamamlanır. Bu modelde, test aktiviteleri ancak diğer tüm yazılım geliştirme aktiviteleri tamamlandıktan sonra gerçekleştirilir.  
  
**B) V-modeli,** test sürecini yazılım geliştirme sürecinin tamamına entegre ederek erken test prensibini uygular. Ayrıca, V-modeli, karşılık gelen her yazılım geliştirme aşaması ile ilgili test seviyeleri içerir; bu da erken testi desteklemektedir. Bu modelde, her test seviyesiyle ilişkili testlerin yürütülmesi sıralı olarak ilerler, ancak bazı durumlarda kesişmeler gerçekleşir.  
  
**Sıralı yazılım geliştirme modelleri** tüm özellikleri tamamlanmış bir yazılım oluşturulduktan sonra yazılımın paydaşlar ve kullanıcılarla paylaşılmasını hedefler, bu yüzden de yazılımın canlıya alınması için aylar veya yıllar gerekebilir.

**2.Döngüsel yazılım geliştirme modeli,**   
ele alınan yazılım özelliklerinin genellikle sabit bir süreye sahip bir dizi döngüde belirlenmesi, tasarlanması, oluşturulması ve birlikte test edilmesi anlamına gelir.

Son yazılım teslim edilinceye veya yazılım

geliştirme durdurulana kadar her döngü, genel yazılım özellikleri kümesinin büyüyen bir alt kümesi olan çalışan yazılımı sunar.

**Artımlı yazılım geliştirme modelleri ise,**   
parçalar halinde gereksinimleri belirlemeyi, sistemi tasarlamayı, oluşturmayı ve test etmeyi içerir; bu, yazılımın özelliklerinin adım adım artması anlamına gelir. Bu artışlarda ele alınacak yazılım özelliklerinin boyutu değişiklik gösterebilir, bazı modellerde büyük parçalar, bazılarında ise daha küçük parçalar bulunur. Ele alınacak bir yazılım özelliği, yeni bir sorgu seçeneği veya kullanıcı arayüzü ekranında yapılan tek bir değişiklik kadar küçük olabilir.

**1.Sıralı bir yazılım geliştirme modeli,**  
yazılım geliştirme sürecini doğrusal, sıralı bir faaliyetler akışı olarak tanımlar.   
Bu, yazılım geliştirme sürecinde her aşamanın önceki aşama tamamlandığında başlaması gerektiği anlamına gelir.  
Teorik olarak aşamaların kesişimi yoktur, ancak uygulamada takip eden aşamadan erken geri bildirim almak faydalıdır.

**Şekil- Test Seviyeleri**

Bu yöntemlerin kullanıldığı birçok yazılım geliştirme çalışması kendi kendini yöneten ekipler kavramını da içerir; bu da test çalışmasının düzenlenme şeklini ve test uzmanları ve yazılımcılar arasındaki ilişkiyi değiştirebilir.

Bu yöntemler, büyüyen bir sistem oluşturur; bu yöntemlerde canlıya alma yazılım özelliği bazında, döngü bazında veya daha klasik bir ana sürüm bazında olabilir. Yazılım özelliklerinin canlıya alınıp alınmadığına bakılmaksızın, yazılım büyüdükçe regresyon testleri giderek önem kazanır.

Sıralı modellerin aksine, döngüsel ve artımlı modeller haftalar veya hatta günler içinde kullanılabilir yazılımlar sunabilir, ancak gereksinimlerin tamamını içeren tam bir ürün setinin sunulması aylar, hatta yıllar sürebilir.

**• Rational Unified Process**: Her döngü göreceli olarak uzun olma eğilimindedir (örneğin iki ila üç ay) ve döngülerde

ele alınan yazılım özellikleri ilgili yazılım özelliklerini de içine alacak şekilde iki veya üç yazılım özelliği grubunu içerir.

**• Scrum:** Her döngü göreceli olarak kısa olma eğilimindedir (örneğin, birkaç saat, gün veya hafta) ve ele alınan yazılım

özellikleri, birkaç geliştirme ve/veya iki veya üç yeni yazılım özelliği gibi, aynı şekilde küçüktür.

**• Kanban:** Sabit uzunluklu döngüler olmadan uygulanır; tamamlandıktan sonra tek bir geliştirme veya yazılım özelliği

sunabilir veya bir kerede sürüm için gerekli yazılım özelliklerini bir grup halinde sunabilir.

**• Spiral (veya prototipleme):** Görsel veya deneysel yazılım özellikleri oluşturmayı içerir, bunlardan bazıları ilerleyen aşamalarda büyük oranda güncellenebilir veya bunlardan vazgeçilebilir.

**3.Sistem Testi**Sistem testleri, bütün bir sistemin veya ürünün davranış ve yeteneklerine odaklanır ve genellikle sistemin gerçekleştirebileceği uçtan uca işleri (fonksiyonalite) ve bu işleri gerçekleştirirken gösterdiği fonksiyonel olmayan davranışları ele alır.

• Sistemin tamamlandığının ve beklendiği gibi çalışacağının sağlamasının yapılması

• Bir bütün olarak sistemin kalitesine dair güven oluşturulması gibi hedefleri vardır.

Sistem test ortamı ideal olarak canlı ortama karşılık gelmelidir.

Sistem testlerini genellikle bağımsız test uzmanları gerçekleştirir.

**4.Kabul Testi**Kabul testleri de sistem testleri gibi genellikle bütün bir sistemin veya ürünün davranışına ve yeteneklerine odaklanır.

• Bir bütün olarak sistemin kalitesine dair güven oluşturulması

• Sistemin tamamlandığının ve beklendiği gibi çalışacağının sağlamasının yapılması gibi hedefleri vardır.

Kabul testleri genellikle sıralı bir yazılım geliştirme yaşam döngüsündeki en son test seviyesi olarak düşünülür, ancak başka zamanlarda da gerçekleştirilebilir.  
  
Kabul testleri genellikle müşterilerin, kullanıcıların, ürün sahiplerinin veya sistem operatörlerinin sorumluluğundadır ve diğer

paydaşlar da bu sürece katılabilir.

**Test seviyeleri**, birlikte düzenlenen ve yönetilen test aktivite gruplarıdır.

**Test seviyeleri,** yazılım geliştirme yaşam döngüsü içindeki diğer aktivitelerle bağlantılıdır. Bu dokümanda kullanılan test seviyeleri aşağıda verilmiştir:

**2.Entegrasyon Testi**

Entegrasyon testleri, birimler veya sistemler arasındaki etkileşimlere odaklanır.

• Arayüzlerin kalitesine dair güvenin oluşturulması,

• Arayüzlerin kendisinde, birimlerde veya sistemlerde hataların bulunması,

• Hataların daha üst test seviyelerine kaçmasının önlenmesi gibi hedefleri vardır.

Birim testlerinde olduğu gibi bazı durumlarda otomatikleştirilmiş entegrasyon regresyon testleri, değişikliklerin mevcut arayüzleri, birimleri veya sistemleri bozmadığına dair güvence sağlar.  
  
Birim entegrasyon testleri genellikle yazılımcıların sorumluluğundadır. Sistem entegrasyon testleri ise genellikle test uzmanlarının sorumluluğundadır.

**4a- Alfa testleri** yazılım geliştiren kuruluşun tesislerinde, sadece yazılım geliştirme ekibi tarafından değil, potansiyel veya mevcut müşteriler ve/veya operatörler veya bağımsız bir test ekibi tarafından gerçekleştirilir.

**4b-Beta testleri** ise potansiyel veya mevcut müşteriler ve/veya

operatörler tarafından kendi ortamlarında yapılır.

**1.Birim Testi**

Birim testleri (**bileşen veya modül testleri)**, ayrı olarak test edilebilen bağımsız birimlere odaklanır.   
  
• Birimin kalitesine dair güven oluşturulması,

• Birimdeki hataların bulunması,

• Hataların gözden kaçarak daha üst test seviyelere girmesinin önlenmesi gibi hedefleri vardır.  
  
Birim testleri genellikle kodu yazan yazılımcı tarafından gerçekleştirilir, bu da test edilecek koda erişim gerektirir.

**Yaygın kullanılan kabul testi çeşitleri:  
1-Kullanıcı kabul testleri (KKT)**  
Genellikle gerçek veya simüle edilmiş bir operasyonel ortamda sistemin hedeflenen kullanıcıların kullanımına uygunluğunun belirlenmesi için yapılır.   
**2-Operasyonel kabul testleri (OKT)**

Operatörler veya sistem yöneticileri tarafından, genellikle simüle edilmiş canlı ortamda gerçekleştirilir.

**3-Sözleşmeye dayalı ve yasal kabul testleri**

Sözleşmeye dayalı kabul testleri, özel olarak geliştirilmiş bir yazılım üretmek için sözleşmenin kabul kriterlerine göre gerçekleştirilir.

**4-Alfa ve beta testleri**

Hata bulma sürecini basitleştirmek ve hataları erkenden bulmak için entegrasyon “büyük patlama” (tüm birimleri veya sistemleri tek bir seferde birleştirmek) şeklinde değil, artımlı (bir seferinde az sayıda ek birim veya sistem) bir şekilde olmalıdır.

İki farklı entegrasyon testi seviyesi vardır:

**• Birim entegrasyonu testleri,**

**• Sistem entegrasyonu testleri,**

Birim ve sistem entegrasyon testleri entegrasyonun kendisine odaklanmalıdır.

**Test çeşidi,** belirlenmiş test hedeflerine dayanarak bir yazılımın belirli özelliklerini veya bir sistemin bir bölümünü test etmeyi amaçlayan bir test aktiviteleri grubudur.

• Bütünlük, doğruluk ve uygunluk gibi **fonksiyonel** kalite özelliklerinin değerlendirilmesi,

• Güvenilirlik, performans, güvenlik, uyumluluk ve kullanılabilirlik gibi **fonksiyonel olmayan** kalite özelliklerinin

değerlendirilmesi,

• Birim veya sistemin **yapısının veya mimarisinin** doğru, eksiksiz ve gereksinimlerde belirtildiği gibi olup olmadığının

değerlendirilmesi,

• **Değişikliklerin etkilerinin değerlendirilmesi**: hataların giderildiğini onaylama (onaylama testleri) ve düzeltilen

hatanın veya yapılan değişikliğin istenmeyen değişiklikleri tetikleyip tetiklemediğini bulma (regresyon testleri) gibi.

**2.Fonksiyonel Olmayan Testler**

Bir sistemin fonksiyonel olmayan testleri, sistemlerin ve yazılımların, kullanılabilirlik, performans veya güvenlik gibi özelliklerini değerlendirir.

Fonksiyonel olmayan testler sistemin yapılması gerekenleri **"ne kadar iyi"** yaptığını ölçümlemeye çalışır. Tüm test seviyelerinde ve sıkça gerçekleştirilmeli ve mümkün

olduğunca erken yapılmalıdır.

Fonksiyonel olmayan testler için test koşullarını ve test senaryolarını üretmekte kara kutu test teknikleri kullanılabilir.

Fonksiyonel olmayan test tasarımı ve koşumunda, bir tasarım veya teknolojinin yapısal açıdan zayıf yönleri (örneğin, belirli programlama dilleriyle ilgili güvenlik açıkları) veya belirli bir kullanıcı tabanı (örneğin hastane yönetim sistemi kullanıcı grupları) hakkında bilgiler gibi özel beceriler veya bilgiler yer alabilir.

**4.Değişiklikle İlgili Testler**

Bir sistemde, bir hatayı çözmek için veya yeni veya değişen fonksiyonalite nedeniyle değişiklikler yapıldığında, değişikliklerin hatayı çözdüğünü veya fonksiyonaliteyi doğru bir şekilde hayata geçirdiğini ve öngörülemeyen olumsuz sonuçlara neden

olmadığını doğrulamak için testler yapılmalıdır.

**A) Onaylama testleri**: Bir hata çözüldükten sonra, hata nedeniyle başarısız olmuş tüm test senaryoları tekrar test

edilebilir. Onaylama testinin amacı, asıl hatanın başarıyla çözülüp çözülmediğini onaylamaktır.

**B) Regresyon testleri:** Kodun bir bölümünde yapılan bir değişikliğin kazara kodun diğer bölümlerinin davranışını olumsuz bir şekilde etkilemesi olasıdır. Bu istenmeyen yan etkilere **regresyon** denir. Regresyon testleri, bu gibi istenmeyen yan etkileri bulmak için yapılan testleri içerir.

Onaylama testleri ve regresyon testleri tüm test seviyelerinde yapılabilir.

**3.Beyaz Kutu Testleri**

Beyaz kutu testleri, sistemin iç yapısına dayanan testleri oluşturur. İç yapı; kod, mimari, iş akışları ve/veya sistem içindeki veri akışlarını içerebilir.

Beyaz kutu testlerinin bütünlük derecesi yapısal kapsam ile ölçülebilir. Yapısal kapsam, bir yapısal öğenin testlerle ne ölçüde ele alınmış olduğunu belirtir ve kapsanan öğenin yüzdesi olarak ifade edilir.

Beyaz kutu test tasarımı ve koşumu, kodun yazılma şekli (örneğin kod kapsamı araçlarını kullanmak), verilerin nasıl depolandığı (örneğin olası veritabanı sorgularını değerlendirmek), kapsam araçlarının nasıl kullanılacağı ve sonuçlarının doğru şekilde yorumlanması gibi özel bilgi veya becerileri içerebilir.

**1.Fonksiyonel Testler**

Fonksiyonel testleri, sistemin gerçekleştirmesi gereken fonksiyonları değerlendiren testleri içerir.   
  
Fonksiyonel gereksinimler; iş gereksinimleri, kullanıcı gereksinimleri, epikler, kullanıcı hikâyeleri, kullanım senaryoları veya fonksiyonel spesifikasyonlar gibi çalışma ürünlerinde tanımlanmış olabilir.

Fonksiyonlar gereksinimler sistemin **“ne yapması’’** gerektiğini tanımlar.

Fonksiyonel testler tüm test seviyelerinde yapılmalıdır ancak odağı her seviyede farklıdır.

Fonksiyonel testler yazılımın davranışını göz önüne alır; bu nedenle, birim veya sistemin fonksiyonalitesi için test koşullarını ve test senaryolarını oluşturmada kara kutu teknikleri kullanılabilir.

**Bakım Testleri**  
  
Üretim ortamlarına alındıktan sonra yazılım ve sistemlerin bakımının yapılması gerekir.   
Herhangi bir değişiklik yapıldığında, hem değişikliklerin ne kadar başarılı olduğunu değerlendirmek hem de sistemin değişmeyen kısımlarındaki olası yan etkileri kontrol etmek için bakım testleri yapılmalıdır.  
Değişiklik, bir platformdan diğerine taşıma, kullanımdan kaldırma gibi nedenler bakım için tetikleyicidir.

**Bakım için Etki Analizi**

Etki analizi, bir değişikliğin beklenen ve olası yan etkilerini, istenilen sonuçların elde edilip edilemeyeceğini ve değişiklikten

etkilenecek alanları belirlemek amacıyla yapılır.

**Statik Testin Faydaları**

• Testlerin maliyet ve süresinin düşürülmesi

• Yaşam döngüsünün ilerleyen kısımlarında veya canlıya alımdan sonra daha az arıza olması sebebiyle yazılımın kullanım ömrü boyunca gerçekleşecek olan toplam kalite maliyetinin düşürülmesi  
  
• Gereksinimlerdeki tutarsızlıkları, belirsizlikleri, çelişkileri, çıkarmaları, yanlışlıkları ve fazlalıkları ortaya çıkararak tasarımdaki veya kodlamadaki hataların önlenmesi

• Gözden geçirmeler sırasında ekip üyeleri arasındaki iletişimin iyileştirilmesi

**3.Statik Testler**

Yazılımın veya diğer çalışma ürünlerinin manuel incelenmesine (gözden geçirmelere) veya kodun veya diğer çalışma ürünlerinin araç kullanılarak değerlendirilmesine

(statik analizlere) dayanır. Her iki statik test çeşidi de test edilen kod veya çalışma ürününü gerçekten çalıştırmadan değerlendirir.  
  
Her türlü çalışma ürünü statik testler kullanılarak incelenebilir. Örneğin: İş gereksinimleri, fonksiyonel gereksinimler ve güvenlik gereksinimleri dâhil olmak üzere tüm gereksinim çeşitleri, epikler, kullanıcı hikâyeleri ve kabul kriterleri, mimari ve tasarım gereksinimleri, kod…

**Gözden Geçirme Süreci**

Gözden geçirmeler**,** resmi veya gayri resmi olabilir.

Gayri resmi gözden geçirmeler tanımlanmış bir süreci takip etmez ve resmi olarak dokümante edilen çıktıları yoktur.

Resmi gözden geçirmelerin özellikleri arasında ise ekip katılımı, gözden geçirme sonuçlarının dokümante edilmesi ve gözden geçirmenin işletilmesi için dokümante edilmiş prosedürler yer alır.

**Statik ve Dinamik Testler Arasındaki Farklar**  
  
Statik ve dinamik testler, farklı türdeki hataları bularak birbirini tamamlarlar. Aralarındaki farklar:

1. Statik testlerin, doğrudan çalışma ürünlerindeki hataları yazılımı çalıştırmadan bulmasıdır.  
  
2.Statik testlerin çalışma ürünlerinin tutarlılığını ve iç kalitesini iyileştirmek için kullanılması, dinamik testlerin ise genellikle dışarıdan görülebilen davranışlara odaklanmasıdır.

Dinamik testlere göre statik testler ile çalışma ürünlerinde yer alan bulunması ve düzeltilmesi daha kolay ve daha ucuz olan yaygın hatalar şunlardır:

Gereksinim hataları, tasarım hataları, kodlama hataları, standartlardan sapmalar, hatalı arayüz gereksinimleri, güvenlik açıkları, test esası izlenebilirliğinde veya kapsamında boşluklar veya yanlışlıklar.

Ayrıca, çoğu sürdürülebilirlik hatası sadece statik testler (örneğin, hatalı modülerleştirme, birimlerin tekrar kullanılabilirliğinin düşük olması, analiz edilmesi ve değiştirilmesi zor olan kodlar) ile bulunabilir.

**Gözden Geçirmelerin Başarı Faktörleri**

• Her gözden geçirme, gözden geçirme planlaması sırasında tanımlanan ve ölçülebilir çıkış kriterleri olarak kullanılan net hedeflere sahiptir.  
• Hedeflere ulaşmak için uygun olan ve gözden geçirilecek çalışma ürünlerinin ve katılımcıların türüne ve seviyesine uygun gözden geçirme çeşitleri uygulanır.  
• Gözden geçirme tekniklerinin tümü, kontrol listesine dayalı veya role dayalı gözden geçirme gibi, gözden geçirilecek çalışma ürünündeki hataların etkili bir şekilde bulunmasına yardımcı olur.  
• Kullanılan tüm kontrol listeleri ana riskleri ele alır ve günceldir

• Büyük dokümanlar küçük parçalar halinde ele alınır ve gözden geçirilir, böylece dokümanı yazanlara hatalar hakkında erken ve sık geri bildirimde bulunularak kalite kontrolü sağlanır.

• Belirlenen hatalar olumlu bir şekilde karşılanır, takdir edilir ve nesnel olarak değerlendirilir.

• Toplantı iyi yönetilir, böylece katılımcılar bunu zamanlarının faydalı bir şekilde kullanımı olarak görürler.

• Gözden geçirme bir güven ortamında gerçekleştirilir; çıktı, katılımcıların kişisel olarak değerlendirilmesinde kullanılmaz.

• Özellikle teftişler gibi daha resmi gözden geçirme çeşitleri için yeterli eğitim sağlanır.

**Çalışma Ürünü Gözden Geçirme Süreci**

•Gözden geçirme süreci aşağıdaki ana aktivitelerden oluşur:  
  
•Planlama

•Gözden geçirmenin başlatılması

•Bireysel gözden geçirme (bireysel hazırlanma)

•Bulguların iletilmesi ve analizi

•Hataların giderilmesi ve raporlama

**Resmi Gözden Geçirmede Roller ve Sorumluluklar**

**1.Yazar (çalışma ürünü yazarı/oluşturucu)**

• Gözden geçirilen çalışma ürününü oluşturur.

• Gözden geçirilen çalışma ürünündeki hataları düzeltir(gerekirse)

**2.Yönetim**

• Gözden geçirmelerin uygulanmasına karar verir.

• Personel, bütçe ve zaman tahsis eder.

• Sürecin verimli işletilmesiyle ilgili kararları verir. Planlama sorumlusu…

**3.Moderatör**

• Gözden geçirme toplantılarının etkin bir şekilde yürütülmesini sağlar.

**4.Gözden geçirme lideri**

• Gözden geçirmenin genel sorumluluğunu üstlenir.

• Sürece kimlerin dâhil olacağına karar verir ve ne zaman ve nerede gerçekleştirileceğini organize eder.

**5.Gözden geçiriciler**

• Konunun uzmanları, projede çalışan kişiler, çalışma ürünü üzerinde söz sahibi olan paydaşlar ve/veya belirli teknik veya iş tecrübesine sahip kişiler olabilir.

• Farklı bakış açılarını temsil edebilir. (örneğin, test uzmanı, programcı, kullanıcı, operatör, iş analisti, kullanılabilirlik uzmanı, vb.)

**6.Yazıcı (veya kaydedici)**

• Her bir gözden geçirme faaliyeti sırasında bulunan bulguları bir araya getirir ve sıraya koyar.

• Gözden geçirme toplantısında belirlenen yeni potansiyel hataları, açık noktaları ve kararları kaydeder.

**Gözden Geçirme Tekniklerinin Uygulanması**

Hataları bulmak için bireysel gözden geçirme (bireysel hazırlık) faaliyeti sırasında uygulanabilecek birkaç gözden geçirme tekniği vardır.   
**1.Kurgusuz**

Çok az hazırlık gerektiren ve yaygın kullanılan bir tekniktir. Bu teknik büyük ölçüde gözden geçiricilerin becerilerine bağlıdır ve farklı gözden geçiriciler tarafından bildirilen birçok benzer sorunun bulunmasına yol açabilir.

**2.Kontrol listesine dayalı**

Gözden geçiricilerin gözden geçirme başlangıcında (örneğin moderatör tarafından) dağıtılan kontrol listelerine dayanarak sorunları belirlediği sistematik bir tekniktir.  
Gözden geçirme kontrol listesi deneyimle elde edilen ve olası hatalara dayanan bir dizi sorudan oluşur.  
Kontrol listesine dayalı tekniğin en önemli avantajı tipik hata çeşitlerinin sistematik bir şekilde kapsanmasıdır.   
**3.Senaryolar ve provalar**

Senaryoya dayalı yaklaşım, çalşma ürününü nasıl okuyacaklarına dair hazırlanan rehberlerle çalışma ürünü üzerinde “provalar” yapılmasını sağlayarak gözden geçiricilerin işini kolaylaştırır.  
**4.Role dayalı**

Gözden geçiricilerin çalışma ürününden etkilenen her bir paydaşın rolleri gözünden çalışma ürününü değerlendirdiği bir tekniktir. Genellikle roller arasında farklı son kullanıcı grupları (deneyimli, deneyimsiz, kıdemli,çocuk vb.) ve kurumdaki belirli roller (kullanıcı yöneticisi, sistem yöneticisi, performans testi uzmanı vb.) yer alır.  
**5.Bakış açısına dayalı**  
Bu teknik de farklı paydaş bakış açılarını dikkate alır. Tipik paydaş bakış açıları arasında son kullanıcı, pazarlama, tasarımcı, test uzmanı veya kurum yönetimi yer alır.

Bakış açısına dayalı okuma ayrıca gözden geçiricilerin, çalışma ürününü kullanmaya çalışmasını da gerektirir.

Ayrıca, bakış açısına dayalı okumalarda kontrol listelerinin kullanılması beklenir.

Yapılan çalışmalar, bakış açısına dayalı okumanın gereksinimleri ve teknik çalışma ürünlerini ‘’**gözden geçirmek için en etkili teknik’’** olduğunu göstermiştir.

**Gözden Geçirme Çeşitleri**

**1.Gayri resmi gözden geçirme (örneğin; çalışma arkadaşının kontrol etmesi, eşleştirme, eşli gözden geçirme)**

• Resmi (dokümante edilmiş) bir sürece dayanmaz.

• Sonuçlar dokümante edilebilir.

• Kontrol listelerinin kullanımı isteğe bağlıdır.

**2.Üzerinden geçme**

• Gözden geçirme toplantısı genellikle çalışma ürününün yazarı tarafından yönetilir.

• Yazıcının olması zorunludur.

• Kontrol listelerinin kullanımı isteğe bağlıdır.

**3.Teknik gözden geçirme**

• Gözden geçiriciler yazarın teknik olarak dengi olan diğer veya aynı disiplinlerdeki teknik uzmanlardan seçilmelidir.

• Gözden geçirme toplantısından önce bireysel hazırlık gereklidir.

• Gözden geçirme toplantısı isteğe bağlı olup, ideal olarak eğitimli bir moderatör (genellikle çalışma ürününün

geliştirilmesine dâhil olmayan kişiler) tarafından yönlendirilir.

• Yazıcı zorunludur, idealde çalışma ürünü yazarından farklı biri olmalıdır.

• Kontrol listelerinin kullanımı isteğe bağlıdır.

• Potansiyel hata kayıtları ve gözden geçirme raporları genellikle oluşturulur.

**4.Teftiş**

• Kurallara ve kontrol listelerine dayanarak, resmi, dokümante çıktılarla tanımlanmış bir süreci izler.

• Gözden geçirme toplantısından önce bireysel hazırlık gereklidir.

• Gözden geçiriciler çalışma ürünü yazarının dengi veya çalışma ürünü ile ilgili diğer disiplinlerde uzman kişilerdir.

• Belirlenen giriş ve çıkış kriterleri kullanılır.

• Yazıcı olması zorunludur.

• Gözden geçirme toplantısı eğitimli bir moderatör tarafından yönetilir.

• Yazar; gözden geçirme lideri, okuyucu veya yazıcı olarak görev yapamaz.

• Potansiyel hata kayıtları ve gözden geçirme raporu oluşturulur.

**4.Test Tasarım Teknikleri  
Test tekniğinin amacı,** test koşullarını, test senaryolarını ve test verilerini belirlemeye yardımcı olmaktır.   
**Kara kutu test teknikleri**, test nesnesinin iç yapısını dikkate almadan test nesnesinin girdi ve çıktılarına odaklanır.  
**Beyaz kutu test teknikleri** (yapısal veya yapıya dayalı teknikler olarak da bilinir), test nesnesinin mimarisinin, ayrıntılı

tasarımının, iç yapısının veya kodunun analizine dayanır

**Tecrübeye dayalı test teknikleri,** testlerin tasarlanması, uyarlanması ve koşturulması için yazılımcıların, test uzmanlarının ve

kullanıcıların tecrübelerini kullanır.

**5.Kullanım Senaryosu Testleri**

Testler, **farklı profildeki kullanıcıların yazılımla etkileşimleri modellemek için** faydalanılan kullanım senaryolarından elde edilebilir; bu testler, kullanıcı gereksinimlerini ele alır.

Her kullanım senaryosu, bir sistemin bir veya daha fazla aktör ile işbirliği içinde gerçekleştirebileceği bazı davranışları belirtir.

Bir kullanım senaryosu, istisnai davranış ve hata ele alma dâhil olmak üzere, temel yazılım davranışının olası varyasyonlarını,

alternatiflerini içerebilir.

Testler, tanımlanmış davranışları (temel, alternatif, istisnai ve hata ele alma) denemek için

tasarlanır.

Kapsam, test edilen kullanım senaryosu davranışlarının toplam kullanım senaryosu davranışı sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

.

**4.Durum Geçişi Testleri**

Birimler veya sistemler, mevcut veya geçmiş durumuna (örneğin, sistemin başlatılmasından bu yana gerçekleşen olaylar)

bağlı olarak gerçekleşen bir olaya farklı şekilde yanıt verebilir. Bir sistemin geçmiş davranışı, durumlar kavramı kullanılarak özetlenebilir.

**Durum geçişi tablosu, durumlar arasındaki tüm geçerli geçişleri ve potansiyel geçersiz geçişleri, ayrıca olayları, koruma koşullarını ve geçerli geçişler için aksiyonları gösterir.**

Durum geçişi testleri, menü bazlı uygulamalar için kullanılabilir ve gömülü yazılım endüstrisinde yaygın bir şekilde kullanılmaktadır.  
  
Bu teknik, ayrıca belirli durumlara sahip bir iş senaryosunu modellemek veya ekran geçişlerini test etmek

için de uygundur.

Kapsam, genel olarak test edilen durum veya geçişlerin sayısının, test nesnesindeki durumların veya geçişlerin toplam sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

**2.Sınır Değer Analizi**  
Sınır değer analizi, denklik paylarına ayırma tekniğinin genişletilmiş halidir ancak yalnızca sayısal veya sıralı verilerden oluşan paylarda kullanılabilir.

**Bir payın minimum ve maksimum değerleri (veya ilk ve son değerleri) sınır değerleridir**

Sınır değer analizi ve testleri, test verilerini sınır değerin ait olduğu payın dışında farklı bir paydan seçerek yazılımın

davranışlarını zorlamaktadır.

Sınır değer analizi tüm test seviyelerinde uygulanabilir. Bu teknik genellikle bir sayı aralığı (tarihler ve saatler dâhil)gerektiren gereksinimleri test etmek için kullanılır.   
  
Kapsam, test edilen sınır değerlerin sayısının tüm sınır değerlerin sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

**3.Karar Tablosu Testleri**

Karar tablosu testlerinin de yer aldığı kombinasyonlu test teknikleri farklı test koşulları kombinasyonlarının nasıl farklı

sonuçlar verdiğini test etmek için kullanılır.

Karar tabloları, bir yazılımın ele alması beklenen karmaşık iş kurallarını kaydetmenin iyi bir yoludur. Karar tabloları oluştururken, test uzmanı **yazılımın koşullarını** (genellikle girdiler) ve **sonuçta ortaya çıkan aksiyonları** (genellikle çıktılar) tanımlar.

Karar tablosu testleri için genel asgari kapsam standardı, tablodaki her karar kuralı için en az bir test senaryosunun

yazılmasıdır. Bu, genellikle tüm koşul kombinasyonlarının kapsanmasını içerir. Kapsam, test edilmiş karar kurallarının

sayısının, karar kurallarının toplam sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

Karar tablosu testlerinin güçlü yanı, hiçbir koşulu göz ardı etmeden tüm önemli koşul kombinasyonlarını ele almayı

sağlamasıdır. Aynı zamanda bu teknik gereksinimlerdeki boşlukların bulunmasında da yardımcı olur. Her test seviyesinde, yazılım davranışının koşullarının bir kombinasyonuna bağlı olduğu tüm durumlarda uygulanabilir.

**Kara Kutu Test Teknikleri**  
  
**1.Denklik Paylarına Ayırma**  
Denklik paylarına ayırma; verileri, paylara (denklik sınıfları) ayırır.

Hem **geçerli** hem de **geçersiz değerler** için denklik payları vardır.

• Her değer yalnızca bir denklik payına ait olmalıdır, birden fazla payda yer almamalıdır.

• Test senaryolarında geçersiz denklik payları kullanıldığında arızaların maskelenmemesini sağlamak için bu paylar ayrı ayrı test edilmelidir, bir geçersiz denklik payı başka bir geçersiz denklik payıyla birleştirilmemelidir.

Bu teknikle %100 denklik payı kapsamı elde etmek için, test senaryoları, her paydan en az bir değer kullanarak, belirlenmiş tüm payları (geçersiz paylar dâhil) kapsamalıdır.   
  
Kapsam, test edilen denklik paylarının sayısının tüm denklik paylarının sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

**Tecrübeye Dayalı Test Teknikleri**  
  
Tecrübeye dayalı test teknikleri uygulanırken, test senaryoları, test uzmanının beceri ve sezgilerinden ve benzer uygulama

ve teknolojilerdeki tecrübelerinden elde edilir. Bu teknikler, diğer daha sistematik tekniklerle kolayca bulunamayan hataları

bulmada yardımcı olabilir. Test uzmanının yaklaşımına ve tecrübesine bağlı olarak, bu teknikler çok çeşitli kapsam ve etkinlik derecelerine ulaşabilir. Kapsamın değerlendirilmesi zor olabilir ve bu tekniklerle ölçülemeyebilir.

**1.Hata Tahminleme**

Hata tahminleme, test uzmanının bilgilerine dayalı olarak yanlışların, hataların ve arızaların oluşmasını sağlamak için

kullanılan bir tekniktir; bu bilgiler aşağıdaki gibidir:  
Uygulamanın geçmişte nasıl çalıştığı, yazılımcıların yapmaya eğilimli oldukları hata çeşitleri, uygulamalarda oluşan arızalar.  
Hata tahminleme tekniğine metodolojik bir yaklaşım, olası yanlışların, hataların ve arızaların bir listesini oluşturmak ve bu

arızaları ve bunlara neden olan hataları ortaya çıkaracak testler tasarlamaktır.

**2.Keşif Testi**

Keşif testlerinde, test koşumu sırasında gayri resmi (önceden tanımlanmamış) testler tasarlanır, koşturulur, kaydedilir ve

dinamik olarak değerlendirilir. Test sonuçları, birim veya sistem hakkında daha fazla bilgi edinmek ve daha fazla test

gerektirebilecek alanlar için testler oluşturmak için kullanılır.  
Keşif testi sürecini daha sistematik yapmak için oturuma dayalı testlerden faydalanılır. Oturuma dayalı testlerde, keşif

testleri belirli bir zaman dilimi içinde gerçekleştirilir ve test uzmanı, testi yönlendirmek için test hedeflerini içeren bir test

tüzüğünü kullanır. Test uzmanı, izlenen adımları ve yapılan keşifleri dokümante etmek için notlar alabilir.  
Keşif testleri, gereksinimler az veya yetersiz olduğunda veya testler üzerinde önemli bir zaman baskısı olduğunda çok işe

yarar. Keşif testleri, diğer daha resmi test tekniklerini tamamlamak için de kullanılır.  
Keşif testleri, tepkisel test stratejileri ile güçlü bir şekilde ilişkilidir. Keşif testleri, diğer kara kutu, beyazkutu ve tecrübeye dayalı tekniklerle birlikte kullanılabilir.

**3.Kontrol Listesine Dayalı Testler**  
  
Kontrol listesine dayalı testlerde, 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.

Fonksiyonel ve fonksiyonel olmayan testleri desteklemek için kontrol listeleri oluşturulabilir. Ayrıntılı test senaryoları

olmadığında, kontrol listesine dayalı testler rehberlik ve belirli bir yere kadar tutarlılık sağlayabilir. Bunlar üst seviye listeler

olduğu için, gerçek testlerde bazı değişikliklerin ortaya çıkması muhtemeldir; bu da potansiyel olarak daha büyük kapsama ama daha düşük tekrarlanabilirliğe yol açar.

**Beyaz Kutu Test Teknikleri**  
  
Beyaz kutu testleri, test nesnesinin iç yapısına dayanır. Beyaz kutu test teknikleri tüm test seviyelerinde kullanılabilir ancak bu bölümde açıklanan koda bağlı iki teknik en yaygın şekilde birim test seviyesinde kullanılır.

**1.Komut Testi ve Kapsamı**

Komut testi, kod içinde yer alan yürütülebilir komutların üzerinden geçilip bu komutların çalıştırılmasıdır. Kapsam, testler tarafından çalıştırılan komutların sayısının test nesnesindeki çalıştırılabilir komutların toplam sayısına bölünmesi ile ölçülür ve normalde yüzde olarak ifade edilir.

**2.Karar Testi ve Kapsamı**

Karar testi, koddaki kararların üzerinden geçilip bu kararların çalıştırılması ve karar çıktılarına dayanarak kodun test edilmesidir. Bunun için test senaryoları karar noktasındaki kontrol akışlarını takip eder (örneğin, bir IF komutu için, biri doğru çıktı ve biri yanlış çıktı; bir CASE komutu için tüm olası çıktıların test senaryoları gerekli olacaktır).

Kapsam, testler tarafından çalıştırılan karar çıktılarının sayısının test nesnesindeki karar çıktılarının toplam sayısına bölünmesi ile ölçülür ve normalde yüzde olarak ifade edilir.

**Komut ve Karar Testlerinin Önemi**

%100 komut kapsama yüzdesi sağlandığında koddaki tüm çalıştırılabilir komutların en az bir kez test edilmesi sağlanır, ancak tüm kararların tamamen test edilmesi sağlanmaz.

%100 karar kapsamı sağlandığında, tüm karar çıktıları çalıştırılır; buna, doğru çıktının ve yanlış çıktının test edilmesi de dâhildir.   
%100 karar kapsamına ulaşılması %100 komut kapsamına ulaşılmasını garanti eder (ancak bunun tersi geçerli değildir).

**5.Test Yönetimi**

**Test bağımsızlığının potansiyel faydaları aşağıda sıralanmıştır:**  
• Farklı geçmiş deneyimleri, teknik bakış açıları ve eğilimleri nedeniyle bağımsız test uzmanlarının, yazılımcılara kıyasla farklı çeşitteki arızaları bulma olasılıkları yüksektir.

• Bağımsız bir test uzmanı, sistemin spesifikasyonunun belirlenmesi ve uygulanması sırasında paydaşların yaptığı varsayımları doğrulayabilir, test edebilir veya çürütebilir

**Test Yöneticisi ve Test Uzmanının(Tester) Görevleri**

Test yöneticisi, genel olarak test sürecinden ve test faaliyetlerine başarılı bir şekilde liderlik edilmesinden sorumludur. Test yöneticisi rolü; profesyonel bir test yöneticisi, proje yöneticisi, yazılım geliştirme yöneticisi veya kalite güvence yöneticisi tarafından gerçekleştirilebilir.   
  
Daha büyük projelerde veya kurumlarda, bir test yöneticisine, test koçuna veya test koordinatörüne bağlı olarak birçok test ekibi çalışabilir ve her ekip, bir test lideri veya deneyimli test uzmanı tarafından yönetilir.   
  
Özellikle ekip içinde yapılan günlük

testlerle ilgili görevler, genellikle ekip içinde çalışan bir test uzmanı tarafından yerine getirilir.

**Testte Bağımsızlık**

Test görevleri, belirli bir test rolündeki kişiler veya başka bir roldeki kişiler tarafından (ör. müşteriler) yerine getirilebilir. Belli bir ölçüde bağımsızlık, test uzmanının hata bulmasını daha etkili kılar. Ancak bağımsızlık, aşina olmanın

yerine geçemez ve yazılımcılar kendi kodlarındaki birçok hatayı verimli bir şekilde bulabilir.

Testlerdeki bağımsızlık dereceleri **(düşük bağımsızlık seviyesinden yüksek seviyeye)** aşağıda sıralanmıştır:

• Bağımsız test uzmanı olmadan; yazılımcıların kendi kodlarını test etmesi

• Yazılımcıların diğer yazılımcıların kodlarını test etmesi veya yazılım geliştirme ekipleri veya proje ekibi içinde yer

alan test uzmanlarının yazılımcıların kodlarını test etmesi

• Kurum içinde, proje yönetimi veya üst düzey yöneticilere bağlı çalışan bağımsız test ekibi veya grubunun olması

• Kurumdan veya kullanıcılardan seçilen test uzmanlarının testleri yapması veya kullanılabilirlik, güvenlik,

performans, yasal/uyumluluk veya taşınabilirlik gibi belirli test çeşitlerinde uzmanlıklara sahip bağımsız test

uzmanlarının testleri yapması

• Kurumun dışından gelen bağımsız test uzmanlarının ekip içine dahil edilerek veya ekip dışında yer alarak testleri

yapması

**Test bağımsızlığının potansiyel zararları aşağıda sıralanmıştır:**  
• Yazılım geliştirme ekibinden ayrı durma; iş birliği eksikliği, yazılım geliştirme ekibine geri bildirim sağlamada gecikmeler veya yazılım geliştirme ekibiyle düşmanca ilişkilere yol açabilir.

• Yazılımcılar kalite konusunda sorumluluk bilincini kaybedebilir.

• Bağımsız test uzmanları bir dar boğaz olarak görülebilir veya sürümdeki gecikmelerden sorumlu tutulabilir.

• Bağımsız test uzmanları (örneğin test nesnesi hakkında) bazı önemli bilgilere sahip olmayabilir.

**Test Koşum Çizelgesi**

Çeşitli test senaryoları ve test prosedürleri üretildikten ve test gruplarına ayrıldıktan sonra, test grupları, çalıştırılma sırasını belirleyen bir test koşum çizelgesinde düzenlenebilir.

Test koşum çizelgesi; önceliklendirme, bağımlılıklar, onaylama testleri, regresyon testleri ve testleri koşturmak için en verimli sıralama gibi faktörleri dikkate almalıdır.

Bazı durumlarda birçok test sıralaması mümkündür; bu sıralamaların verimlilik seviyeleri de farklılık gösterir. Bu gibi durumlarda, test koşum verimliliği ile önceliklendirme arasında bir denge kurulmalıdır.

**Giriş Kriterleri ve Çıkış Kriterleri (Hazır ve Tamamlandı)**

Test faaliyetinde giriş kriterleri (genellikle çevik yazım geliştirmede “Hazır” tanımı olarak adlandırılır), belirli bir test faaliyetinin gerçekleştirilmesi için gereken ön koşulları tanımlar.

Giriş kriterleri karşılanmazsa, faaliyetin daha zor olması, daha uzun sürmesi, daha maliyetli ve daha riskli olması muhtemeldir.   
  
Çıkış kriterleri (genellikle çevik yazılım geliştirmede “Tamamlandı” tanımı olarak adlandırılır), bir test seviyesi veya bir test grubunu tamamlandı olarak tanımlamak için hangi koşulların sağlanması gerektiğini tanımlar.

Çıkış kriterleri karşılanmasa bile, bütçenin tükenmesi, planlanan süresinin tamamlanması ve/veya ürünün piyasaya sürülmesi için oluşan baskı nedeniyle test faaliyetlerinin kısa kesilmesi de yaygın bir durumdur.

**Test Stratejisi ve Test Yaklaşımı**

Test stratejisi, test sürecinin genellikle ürün veya kurum düzeyinde genel bir tanımını sağlar.  
Test yaklaşımı ise belirli bir proje veya sürüm için test stratejisini düzenler.

Yaygın test stratejisi çeşitleri aşağıda verilmiştir:

• **Analitik:** Risk bazlı testler gibi  
• **Model Bazlı:** İş süreci modelleri, durum modelleri ve güvenilirlik büyüme modelleri bulunur.  
• **Metodik:** Test kümelerinin veya test koşullarının sistematik olarak kullanılması gibi  
• **Süreç uyumluluk(veya standartlara uyumluluk)**

• Yönlendirmeli (veya danışılarak)  
• **Regresyon hassasiyetli**: Yazılımın mevcut çalışan özelliklerinin bozulmasını (regresyon) engelleme amacıyla hayata geçirilmektedir.

• **Tepkisel:** Bu tür test stratejilerinde, testler, test edilen birim veya sisteme ve test koşumu sırasında meydana gelen olaylara verilen tepkilerle şekillenir. Keşif testleri, tepkisel stratejilerde kullanılan yaygın bir tekniktir.

**Test Tahminleme Teknikleri**

Yeterince test etmek için gerekli olan eforu belirlemek amacıyla kullanılan bazı tahminleme teknikleri vardır. En sık kullanılan tekniklerden ikisi aşağıda verilmiştir:

• **Metrik bazlı teknik**: önceki benzer projelerin metriklerine veya tipik değerlere dayanarak test eforunu tahmin etme. Örneğin; hata giderme modelleri, (burndown tabloları (yapılacak işler tablosu))

• **Uzman bazlı teknik:** testi gerçekleştirecek kişilerin tecrübesine veya uzmanlara dayanarak test eforunu tahmin

etme. Örneğin; . Wideband Delphi tahminleme tekniği, poker planlama(ekip üyelerinin kendi deneyimlerine dayanarak yapılacak iş için gereken çalışmayı tahmin ettikleri bir teknik)

**Test Eforunu Etkileyen Faktörler**

**Test eforu tahminlemesi,** belirli bir proje, sürüm veya döngüdeki testlerin hedeflerini yerine getirmek için yapılması gereken testlerle ilgili çalışmanın miktarını öngörmektir. Test çalışmasını etkileyen faktörler arasında ürünün özellikleri, yazılım geliştirme sürecinin özellikleri, testi yapacak kişilerin özellikleri ve test sonuçları sayılabilir.

**HATA YÖNETİMİ**

Testlerin hedeflerinden biri de hataların bulunması olduğu için testlerde bulunan hatalar raporlanmalı ve kaydedilmelidir.

Dinamik testler sırasında oluşturulan bir hata raporu genellikle aşağıdakileri içerir:

• Hatanın seri numarası veya sırası

• Bulunan hatanın başlığı ve kısa bir özeti

• Hata raporunun tarihi, düzenleyen kurum ve yazan kişi

• Test öğesinin (test edilen yapılandırma öğesi) ve ortamının tanımı

• Hata raporunun durumu (ör. açık, ertelenmiş, tekrarlanmış, düzeltilmeyi bekliyor, onaylama testleri bekleniyor,

yeniden açıldı, kapatıldı)

• Sonuçlar, öneriler ve onaylar

**Bazı Tanımlar**  
**Risk,** gelecekte olumsuz sonuçlara yol açacak bir olayın gerçekleşme olasılığını içerir. Risk seviyesi, olayın olasılığı ve etkisi (zararı) ile belirlenir.  
 **Ürün riski,** bir çalışma ürününün (örneğin bir gereksinim, birim, sistem veya test) kullanıcılarının ve/veya paydaşlarının meşru ihtiyaçlarını karşılamaması ihtimalini içerir.   
  
**Kalite riski**, bir ürünün belirli kalite karakteristikleriyle ilişkilendirildiğinde (örneğin, fonksiyonalite, güvenilirlik, performans, kullanılabilirlik, güvenlik, uyumluluk, sürdürülebilirlik ve taşınabilirlik), ürün risklerine kalite riskleri de denir.

**Proje riski,** gerçekleşmesi durumunda, projenin hedeflerine ulaşma imkânı üzerinde olumsuz etkiye sahip olabilecek durumları içerir.

**Risk Bazlı Testler ve Ürün Kalitesi**

**Risk,** test eforunu odaklamak için kullanılır. Testlere nerede ve ne zaman başlanacağını ve daha fazla dikkat gerektiren alanları belirlemek için kullanılır. Testler, olumsuz bir olay gerçekleşme olasılığını azaltmak veya olumsuz bir olayın etkilerini azaltmak için kullanılır.

**Testler,** belirlenen risklerin riskini azaltma ve onlar hakkında geri bildirim sağlama, kalan (ele alınamamış) riskler hakkında ise geri bildirim sağlamak için kullanılır.

**Ürün risk analizi,** ürün risklerinin belirlenmesi, belirlenen her bir risk için gerçekleşme olasılığı ve etkilerinin değerlendirmesi aktivitelerini içerir.

**Risk bazlı testler,** ürün risk analizini gerçekleştirmek için proje paydaşlarının ortak bilgi ve sezgilerine dayanır. Üründe arıza olasılığının en aza indirilmesi için risk yönetimi faaliyetleri disiplinli bir yaklaşım sağlar.

**6.Yazılım Test Araç Desteği**

**Test Aracı Sınıflandırması**

Testlerde araçların kullanılmasının potansiyel faydaları ve sunduğu fırsatların yanında bazı riskleri de vardır. Bu durum, özellikle test koşumu araçları (genellikle test otomasyonu olarak da adlandırılan) için geçerlidir.

Test koşumunu desteklemek için araç kullanmanın potansiyel faydaları aşağıda verilmiştir:

• Tekrarlanan manuel testlerde azalma dolayısıyla zamandan tasarruf

• Daha fazla tutarlılık ve tekrarlanabilirlik

• Daha objektif değerlendirme

• Testlerle ilgili bilgilere daha kolay erişim

**Test Araçlarındaki Önemli Hususlar**  
  
Test araçları, bir veya daha fazla test faaliyetini desteklemek için kullanılabilir. Bu araçlar aşağıda verilmiştir:

**• Doğrudan testlerde kullanılan araçlar:** test koşum araçları ve test verisi hazırlama araçları gibi

**• Gereksinimlerin, test senaryolarının, test prosedürlerinin, otomatikleştirilmiş test betiklerinin, test sonuçlarının, test verilerinin ve hataların yönetilmesine yardımcı olan ve test koşumunun raporlanması ve gözetimi için kullanılan araçlar**

**• Araştırma ve değerlendirme için kullanılan araçlar**

**• Testlerde yardımcı olan herhangi bir araç** (bu anlamda Excel de bir test aracıdır)

**Testleri desteklemek için araç kullanmanın potansiyel riskleri aşağıda verilmiştir:**

• Araç için beklentiler gerçekçi olmayabilir.

• Bir aracın kullanılmaya başlanması için gerekecek zaman, maliyet ve efor olması gerekenden daha az tahmin

edilebilir.  
• Araçtan yüksek miktarda ve sürekli fayda elde etmek için gereken zaman ve efor olması gerekenden daha az olarak

tahmin edilebilir.

• Aracın ürettiği test varlıklarının bakımı ve güncel tutulması için gereken efor olması gerekenden daha az olarak tahmin edilebilir.

• Araca çok fazla bağlı olunabilir.

• Test varlıklarının versiyon kontrolü ihmal edilebilir.

• Gereksinim yönetimi araçları, yapılandırma yönetimi araçları, hata yönetimi araçları gibi kritik önemdeki araçlar ve birden çok tedarikçinin araçları arasındaki ilişkiler ve birlikte çalışabilirlik sorunları gözden kaçırılabilir.

• Aracın tedarikçisi iflas edebilir, aracı desteklemeyi bırakabilir veya aracı farklı bir kuruma satabilir

• Tedarikçi; bakım, destek, yeni versiyonlar ve hata düzeltmeleri için yeterli desteği vermeyebilir

• Açık kaynak kodlu bir yazılımın geliştirilmesi geçici olarak durdurulabilir

• Yeni bir platform veya teknoloji test aracı tarafından desteklenmeyebilir

• Aracın net bir sahibi olmayabilir.

**Test koşumu araçları**

Test koşumu araçları, otomatikleştirilmiş test betikleri kullanarak test nesnelerini koşturur. Bu tür bir araç, genellikle belirgin bir fayda elde etmek için önemli ölçüde çalışma gerektirir.  
  
**Veri güdümlü test yaklaşımı,** test girdilerini ve beklenen sonuçları, genellikle bir elektronik tablo şeklinde ayırır ve giriş verilerini okuyabilen ve aynı test betiğini farklı verilerle çalıştırabilen daha genel bir test betiği kullanır. Betik dilini bilmeyen test uzmanları daha sonra bu önceden tanımlanmış betikler için yeni test verileri yaratabilir.

**Aksiyon kelimesi güdümlü test yaklaşımında**, genel bir betik, gerçekleştirilecek işlemleri tanımlayan anahtar kelimeleri (bunlara aksiyon kelimeleri de denir) işler, ardından ilişkili test verilerini işlemek için anahtar kelime betiklerini çağırır. Test uzmanları (betik diline hakim olmasalar bile), anahtar kelimeleri ve ilgili verileri kullanarak testleri tanımlayabilirler, bunlar da test edilen uygulamaya göre uyarlanabilir.

**Model Bazlı test (MBT) araçları,** bir fonksiyonel gereksinimin, aktivite diyagramı gibi bir model biçiminde kaydedilmesine olanak sağlar. Bu görev genellikle bir sistem tasarımcısı tarafından gerçekleştirilir. MBT aracı, test senaryosu gereksinimlerini oluşturmak için modeli yorumlar; bu, daha sonra bir test yönetim aracına kaydedilebilir ve/veya bir test koşum aracı

tarafından yürütülebilir.