Düzgün çalışmayan yazılımlar; para, zaman veya ticari itibar kaybetme ve hatta yaralanma veya ölüm gibi birçok soruna yol açabilir. Yazılım testi, yazılımın kalitesini değerlendirmenin ve kullanım sırasında oluşabilecek yazılım hatası riskini azaltmanın bir yoludur.

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.

Bazı testler, test edilen birimin veya sistemin çalıştırılmasını içerir; bu testlere dinamik testler denir. Diğer testler, test edilen birimin veya sistemin çalıştırılmasını gerektirmez; bu testlere ise statik testler denir. Dolayısıyla testler; gereksinimler, kullanıcı hikâyeleri ve kaynak kod gibi çalışma ürünlerinin gözden geçirilmesini de içerir

* + 1. **Yazılım Testinin Genel Hedefleri**

Gereksinimler, kullanıcı hikâyeleri, tasarım ve kod gibi çalışma ürünlerini değerlendirmek

• Belirtilen tüm gereksinimlerin yerine getirilip getirilmediğini doğrulamak

• Hataları önlemek

• Arızaları ve hataları tespit etmek

• Yazılımın kalitesiz olma riskini düşürmek (örneğin, canlıda meydana gelebilecek arızaların önceden tespiti)

**Testin hedefleri, proje bağlamına, test seviyesine ve yazılım geliştirme yaşam döngüsü modeline bağlı olarak değişebilir.**

Birim testi sırasındaki hedeflerden biri, olabildiğince çok sayıda arıza tespit etmek ve böylece bunlara neden olan hataların erkenden tespitini ve düzeltilmesini sağlamak olabilir.

Kabul testi sırasındaki hedeflerden biri sistemin beklendiği gibi çalıştığını ve gereksinimleri karşıladığını doğrulamak olabilir.

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.

Bir kişinin yaptığı bir yazılım hatası (yanlışlık), yazılım kodunda veya ilgili başka bir çalışma ürününde bir hatanın (kusur) ortaya çıkmasına neden olabilir.

Çalışma ürününde bir hata oluşmasına neden olan bir yazılım hatası, bir hatayı tetikleyebilir, bu da ilgili bir çalışma ürününde başka bir hatanın ortaya çıkmasına neden olabilir.

Hatalı kod çalıştırıldığında, bir arızaya neden olabilir, ancak bu durum her koşulda gerçekleşmeyebilir

Beklenmedik test sonuçlarının tamamı arıza değildir. Yanlış pozitifler, testlerin uygulanma şeklindeki hatalardan veya test verilerindeki, test ortamındaki veya diğer test yazılımındaki hatalar nedeniyle veya diğer nedenlerden dolayı ortaya çıkabilir. Benzer hataların veya insan hatalarının yanlış negatiflere neden olduğu bunun tersi durumlar da ortaya çıkabilir. Yanlış negatifler, bulunması gereken hataları bulamayan testlerin sonuçlarıdır; yanlış pozitifler hata olarak rapor edilmesine rağmen, gerçekte hata değillerdir.

Yedi Test Prensibi

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

Test Aktiviteleri ve Yapılacak İşler

Bir test süreci aşağıdaki ana aktivite gruplarından oluşur:

• Test planlama

• Test gözetimi ve kontrolü

• Test analizi

• Test tasarımı

• Test uyarlama

• Test koşumu

• Test tamamlama

Test planlaması, test hedeflerini belirleyen aktiviteler ve proje bağlamının dayattığı kısıtlamalar dâhilinde test hedeflerini yerine getirme yaklaşımını içerir (örneğin, uygun test tekniklerini ve işlerini belirlemek ve bunları belirtilen zamanda bitirmek için bir test zaman çizelgesi oluşturmak)

Test gözetimi, test planında tanımlanan test gözetim metriklerini kullanarak planlanan ile gerçekleşen ilerlemenin sürekli karşılaştırılmasını içerir. Test kontrolü, test planında belirlenen hedeflere ulaşmak için gerekli önlemlerin alınmasını içerir. Test gözetimi ve kontrolü, çıkış kriterlerinin değerlendirilmesiyle desteklenir; bu kriterler bazı yaşam döngülerinde “Tamamlandı” tanımı olarak da bahsedilir

Test analizi sırasında, test edilebilir özellikleri belirlemek ve ilgili test koşullarını tanımlamak için test esası analiz edilir. Diğer bir deyişle, test analizi ölçülebilir kapsama kriterleri açısından “neyin test edileceğini” belirler

Test analizi “ne test edilecek?” sorusuna cevap verirken,

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 öğesinin/öğelerinin veya test nesnesinin, test aracının/araçlarının ve test yazılımının kimlik numaralarının ve versiyonlarının kaydedilmesi

• Testlerin manuel olarak veya test koşum araçları kullanılarak koşulması

• Gerçekleşen sonuçlarla beklenen sonuçların karşılaştırılması

• Test koşum sonucunun kaydedilmesi (örneğin, başarılı, başarısız, engellendi)

• Gözlemlenen arızalara dayanılarak hataların bildirilmesi

Test tamamlama

Test tamamlama aktiviteleri; test projesinde elde edilen deneyimi, proje boyunca üretilen test yazılımını ve elde edilen diğer ilgili bilgileri toplar ve bir araya getirir

Yazılım Geliştirme ve Yazılım Testi

• Her yazılım geliştirme aktivitesine karşılık gelen bir test aktivitesi vardır.

• Her test seviyesinin bu seviyeye özgü test hedefleri vardır.

• Belirli bir test seviyesi için test analizi ve tasarımı, ilgili yazılım geliştirme faaliyeti sırasında başlar.

Hangi yazılım geliştirme yaşam döngüsü modeli seçilirse seçilsin, test faaliyetleri yaşam döngüsünün erken aşamalarında başlamalı ve testleri erkenden yapma ilkesine uyulmalıdır.

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.

Şelale modelinden farklı olarak V-modeli test sürecini yazılım geliştirme sürecinin tamamına entegre ederek erken test prensibini uygular.

proje risklerine göre uygun bir yazılım geliştirme yaşam döngüsü modeli seçilmeli ve uyarlanmalıdır

Test Seviyeleri

Her test seviyesi için uygun bir test ortamı gereklidir. Örneğin kabul testinde canlı ortam benzeri bir test ortamı ideal iken, birim testinde yazılımcılar genellikle kendi yazılım geliştirme ortamlarını kullanırlar.

Birim Testi

• Birim testleri (bileşen veya modül testleri olarak da bilinir), ayrı olarak test edilebilen bağımsız birimlere odaklanır.

• Birimin fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması

• Birimdeki hataların bulunması

• Hataların gözden kaçarak daha üst test seviyelerine girmesinin önlenmesi

Birim testleri genellikle yazılım geliştirme yaşam döngüsü modeline ve sisteme bağlı olarak, sahte (mock) nesneler, servis sanallaştırması, kuluçkalar, taklit uygulamalar ve sürücülerin kullanımını gerektirebilmekte ve sistemin geri kalanından izole olarak yapılmaktadır.

Birim testleri fonksiyonaliteyi (ör. hesaplamaların doğruluğu), fonksiyonel olmayan özellikleri (ör. bellek sızıntılarını bulma) ve yapısal özellikleri (ör. karar testleri) kapsayabilir.

Hatalar, genellikle tanımlı bir hata yönetim süreci olmadan, bulundukları anda giderilir.

Birim testleri genellikle kodu yazan yazılımcı tarafından gerçekleştirilir, bu da test edilecek koda erişim gerektirir. Yazılımcılar bu seviyede geliştirme yapma ve hataların bulunup giderilmesini sağlama arasında gidip gelebilirler. Yazılımcılar genellikle bir birimin kodunu yazdıktan sonra testleri yazar ve yürütür.

**Entegrasyon Testi**

Entegrasyon testleri, birimler veya sistemler arasındaki etkileşimlere odaklanır. Entegrasyon testlerinin hedefleri aşağıda verilmiştir.

Arayüzlerin fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması.

Birim testlerinde olduğu gibi bazı durumlarda otomatikleştirilmiş entegrasyon regresyon testleri, değişikliklerin mevcut arayüzleri, birimleri veya sistemleri bozmadığına dair güvence sağlar.

Birim entegrasyonu testleri, entegre birimler arasındaki etkileşimlere ve arayüzlere odaklanır.

Sistem entegrasyonu testleri, sistemler, paketler ve mikroservisler arasındaki etkileşimlere ve arayüzlere odaklanır

Entegrasyon testlerinde ele alınan tipik test nesneleri aşağıda verilmiştir: • Alt-sistemler • Veritabanları Altyapı • Arayüzler • API'ler • Mikroservisler

**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 fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimlerde tanımlandığı gibi olup olmadığının doğrulanması

Sistemin tamamlandığının ve beklendiği gibi çalışacağının sağlamasının yapılması

**Kabul Testi**

Sistem testleri gibi kabul testleri de genellikle bütün bir sistemin veya ürünün davranışına ve yeteneklerine odaklanır.

Sistemin tamamlandığının ve beklendiği gibi çalışacağının sağlamasının yapılması

Kabul testleri, sistemin müşteriye (son kullanıcı) çıkmaya ve kullanıma hazır olduğunu değerlendirmeye yönelik bilgiler elde etmek için yapılırKabul testleri sırasında hatalar bulunabilir, ancak hata bulmak genellikle kabul testinin bir amacı değildir ve kabul testleri sırasında önemli sayıda hata bulunması bazı durumlarda büyük bir proje riski olarak kabul edilebilir.

**Alfa ve Beta Testler**

Alfa ve beta testleri genellikle yazılım ürünü piyasaya sürülmeden önce potansiyel veya mevcut kullanıcılar, müşteriler ve/veya operatörlerden geri bildirim almak isteyen ticari paket yazılımın (COTS) geliştiricileri tarafından kullanılır.

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

Beta testleri ise potansiyel veya mevcut müşteriler ve/veya operatörler tarafından kendi ortamlarında yapılır.

\* Beta testleri alfa testlerinden sonra gelebilir veya öncesinde hiç alfa testi yapılmadan da gerçekleştirilebilir.

**Test Çeşitleri**

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.

**Fonksiyonel Testler**

Bir sistemin fonksiyonel testleri, sistemin gerçekleştirmesi gereken fonksiyonları değerlendiren testleri içerir.

Fonksiyonlar gereksinimler sistemin ne yapması gerektiğini tanımlar.

Fonksiyonel testler tüm test seviyelerinde yapılmalıdır.(örneğin; birim seviyesindeki fonksiyonel testler ilgili birimin gereksinimlerine dayandırılmalıdır), 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

**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.

Yaygın yanlış bilgilerin aksine fonksiyonel olmayan testler tüm test seviyelerinde ve sıkça gerçekleştirilmeli ve mümkün olduğunca erken yapılmalıdır.

Yaygın yanlış bilgilerin aksine fonksiyonel olmayan testler tüm test seviyelerinde ve sıkça gerçekleştirilmeli ve mümkün olduğunca erken yapılmalıdır.

**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.

**Onaylama testleri:**

Bir hata çözüldükten sonra, hata nedeniyle başarısız olmuş tüm test senaryoları tekrar test edilebilir; bu testler yazılımın yeni versiyonunda yeniden koşturulmalıdır.Onaylama testinin amacı, asıl hatanın başarıyla çözülüp çözülmediğini onaylamaktır.

**Regresyon testleri:**

Kodun bir bölümünde yapılan bir değişikliğin (bir düzeltme veya başka bir değişiklik çeşidi olabilir) kazara kodun diğer bölümlerinin (aynı birim içinde, aynı sistemin diğer birimlerinde ve hatta diğer sistemlerde) davranışını olumsuz bir şekilde etkilemesi olasıdır. Değişiklikler, işletim sisteminin veya veritabanı yönetim sisteminin yeni bir versiyonu gibi ortamda yapılan değişiklikler de olabilir. 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.

**-Statik Testin Temelleri-**

Test edilen yazılımın çalıştırılmasını gerektiren dinamik testlerin aksine statik testler, yazılımın veya diğer çalışma ürünlerinin manuel incelenmesine (gözden geçirmelere) veya kodun veya diğer çalışma ürünlerinin araç kullanılarak değerlendirilmesine (statik analizlere) dayanır.

Yazılım geliştirme yaşam döngüsünün erken dönemlerinde uygulandığında statik testler, dinamik testler yapılmadan önce hataların erkenden bulunmasına olanak sağlar.

Gözden geçirmeler, katılımcıların okuyup anlayabileceği her çalışma ürününe uygulanabilir.

**Statik ve Dinamik Testler Arasındaki Farklar**

Önemli farklarından birisi de statik testlerin, yazılım çalışırken hataların neden olduğu arızaları bulmak yerine doğrudan çalışma ürünlerindeki hataları çalıştırmadan bulmasıdır.

Statik testler hatayı çok daha az eforla bulabilir.

Diğer bir ayrım ise, statik testlerin çalışma ürünlerinin tutarlılığını ve iç kalitesini iyileştirmek için kullanılması, dinamik testlerin ise genellikle dışarıdan görülebilen davranışlara odaklanmasıdır.

**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.

**Çalışma Ürünü Gözden Geçirme Süreci**

Planlama

Gözden geçirmenin başlatılması

Bireysel gözden geçirme (bireysel hazırlanma)

Bulguların iletilmesi ve analizi

Hataların giderilmesi ve raporlama

**Resmi Gözden Geçirmede Roller ve Sorumluluklar**

Yazar (çalışma ürünü yazarı/oluşturucu)=Gözden geçirilen çalışma ürününü oluşturur Gözden geçirilen çalışma ürünündeki hataları düzeltir(gerekirse)

Yönetim

Moderatör

Gözden geçirme lideri

Gözden geçiriciler

Yazıcı (veya kaydedici)

**Gözden Geçirme Çeşitleri**

* Gayri resmi gözden geçirme (örneğin; çalışma arkadaşının kontrol etmesi, eşleştirme, eşli gözden geçirme)
* Üzerinden geçme
* Teknik gözden geçirme
* Teftiş

**Kara Kutu Test Teknikleri**

Kara kutu test teknikleri (davranışsal veya davranışa dayalı teknikler olarak da bilinir) uygun bir test esasının (örneğin gereksinim dokümanları, spesifikasyonlar, kullanım senaryoları, kullanıcı hikâyeleri veya iş süreçleri) analizine dayanır. Bu teknikler hem fonksiyonel hem de fonksiyonel olmayan testlere uygulanabilir. Kara kutu test teknikleri, test nesnesinin iç yapısını dikkate almadan test nesnesinin girdi ve çıktılarına odaklanır.

**Beyaz Kutu Test Teknikleri**

Beyaz kutu test teknikleri (yapısal veya yapıya dayalı teknikler olarak da bilinir), test nesnesinin mimarisinin, ayrıntılı tasarımının, iç yapısının veya kodunun analizine dayanır. Kara kutu test tekniklerinden farklı olarak, beyaz kutu test teknikleri test nesnesinin içindeki yapı ve işlemlere odaklanır.

Tecrübeye dayalı test teknikleri, testlerin tasarlanması, uyarlanması ve koşturulması için yazılımcıların, test uzmanlarının ve kullanıcıların tecrübelerini kullanır. Bu teknikler genellikle kara kutu ve beyaz kutu test teknikleriyle birleştirilir.

**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 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.

**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

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

**Risk Tanımı**

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. Ürün riskleri, bir ürünün belirli kalite karakteristikleriyle ilişkilendirildiğinde (örneğin, fonksiyonalite, güvenilirlik, performans, kullanılabilirlik, güvenlik, uyumluluk, sürdürülebilirlik ve taşınabilirlik), ürün risklerine kalite riskleri de denir.

Yazılım; kullanıcı, müşteri ve/veya paydaş ihtiyacına göre amaçlanan fonksiyonları yerine getiremeyebilir. • Sistem mimarisi, fonksiyonel olmayan bazı gereksinimleri uygun şekilde desteklemeyebilir • Bazı durumlarda belirli bir hesaplama yanlış yapılabilir. • Bir döngü kontrol yapısı yanlış kodlanmış olabilir

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

Proje sorunları: o Teslimatta, görevi tamamlamada veya çıkış kriterlerinin veya tamamlandı tanımının karşılanmasında gecikmeler olabilir o Yanlış tahminler, maddi kaynakların daha yüksek öncelikli projelere yeniden tahsis edilmesi veya kurum genelinde genel harcama kısıntıları, parasal kaynak yetersizliğine neden olabilir o Geç yapılan değişiklikler önemli ölçüde yeniden yapılması gereken işlere neden olabilir.

****Sahte pozitif / False Positive**** durumunda, test uzmanı gerçekte hata olmayan bir hata bulur. Test uzmanının yazılımı test etmesi ve internet bağlantısı kesildiği için web sitesinin yüklenmemesini hata olarak rapor etmesi gibi bir durum buna örnek olarak verilebilir.

****Sahte negatif / False Negative**** durumunda ise yazılımda bir hata vardır ancak test uzmanı bunu bulamamıştır. Örneğin, test uzmanının tüm test senaryolarının mobil uygulama için portre modunda (telefon dik durumda iken) yürüttüğünü düşünelim fakat uygulamayı yatay modunda kullanmadıkça ortaya çıkmayan keşfedilmemiş hatalar da vardır. Bu durumda test uzmanı bu hataları raporlayamaz.

Kabul testinin amacı, sisteme, sistemin parçalarına veya sistemin fonksiyonel olmayan gereksinimlerine karşı güven oluşturmaktır. Kabul testinde ana odak hataları bulmak değildir, sistemin canlıya hazır olduğunu göstermektir.

Regresyon, yapılan değişiklikler sonucunda oluşan yeni hataları keşfetmek amacıyla zaten test edilmiş olan bir programı yeniden test etme işlemidir. Yeni oluşan hatalar, test edilmekte olan yazılımda veya doğrudan veya dolaylı bir başka yazılım bileşenin de olabilir. Yazılım veya yazılımın ortamı değiştirildiğinde gerçekleştirilir. Regresyonun kapsamı, daha önceden çalışan yazılımdaki hataları bulamama riskine dayanır.

Gözden geçirme, yazılımı (kod dahil) test etmenin bir yoludur ve dinamik testler yapılmadan önce gerçekleştirilebilir. Yazılım geliştirme yaşam döngüsünün başlarında gözden geçirmeler sırasında tespit edilen hataları ortadan kaldırmak (örn. gereksinimlerde bulunan hatalar) genellikle yürütülen kod üzerinde çalıştırılan testlerle tespit edilen hataları ortadan kaldırmaktan daha ucuzdur.

Gözden geçirmenin avantajları arasında, erken hata tespiti ve düzeltmesi, geliştirme sürecinde üretkenlik iyileştirmeleri, daha hızlı yazılım geliştirme, azalan test maliyeti ve süresi, yaşam boyu maliyet azalmaları, daha az hata ve gelişmiş iletişim sayılabilir. Gözden geçirmeler, dinamik testte bulunamayan gereksinim eksikliklerini bulabilir.

****Proje Riskleri****

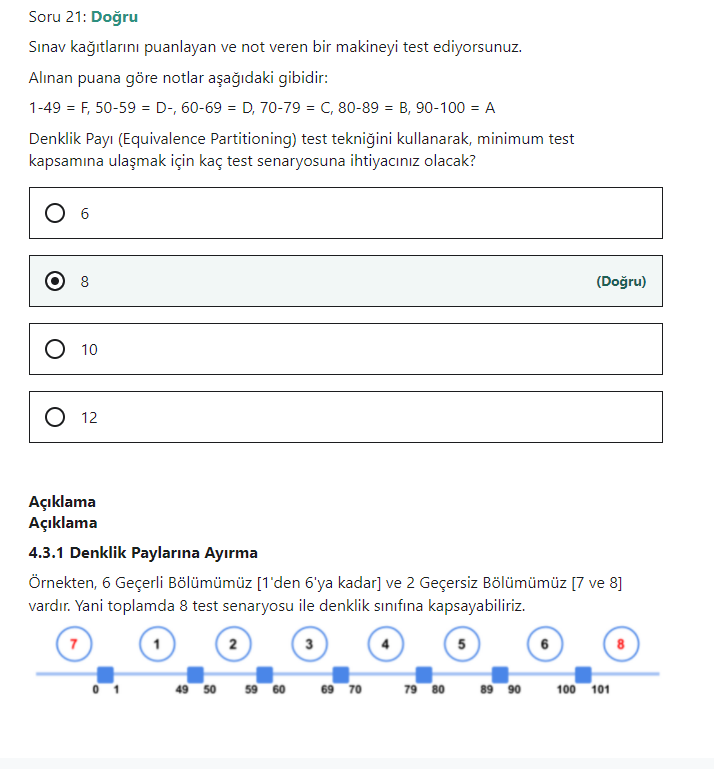
Proje riskleri, projenin hedeflerine ulaşmasını engelleyebilecek risklerdir, örneğin;

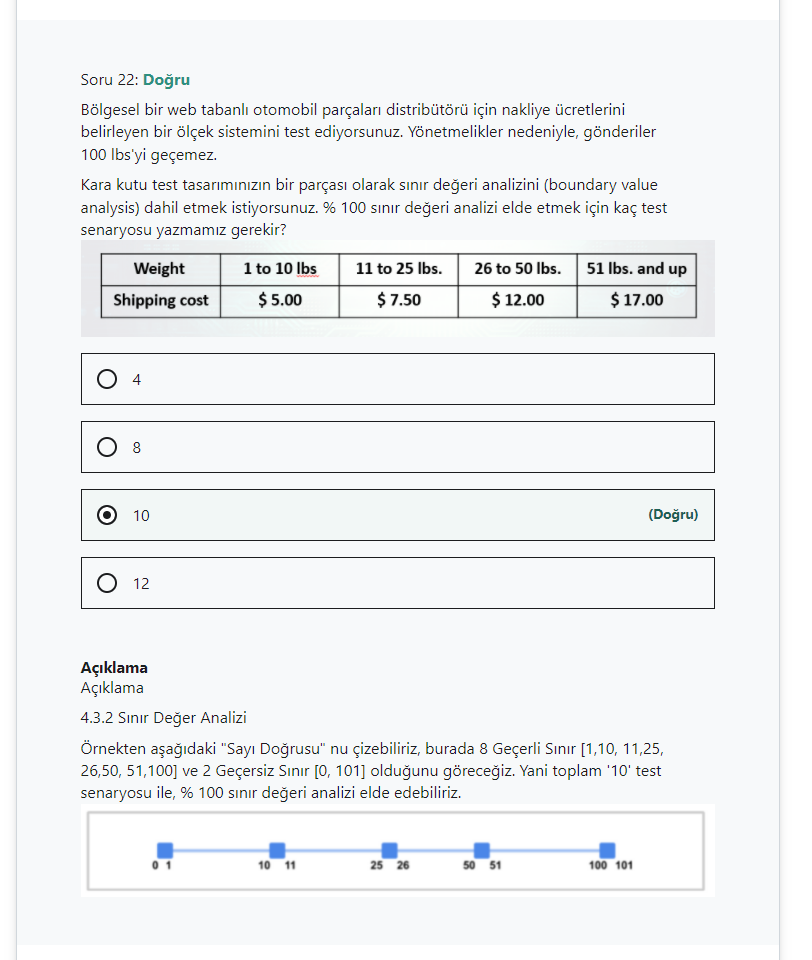
Bir formüldeki hata nedeniyle yanlış hesaplamalar yapan bir modül - Bu bir ürün riskidir

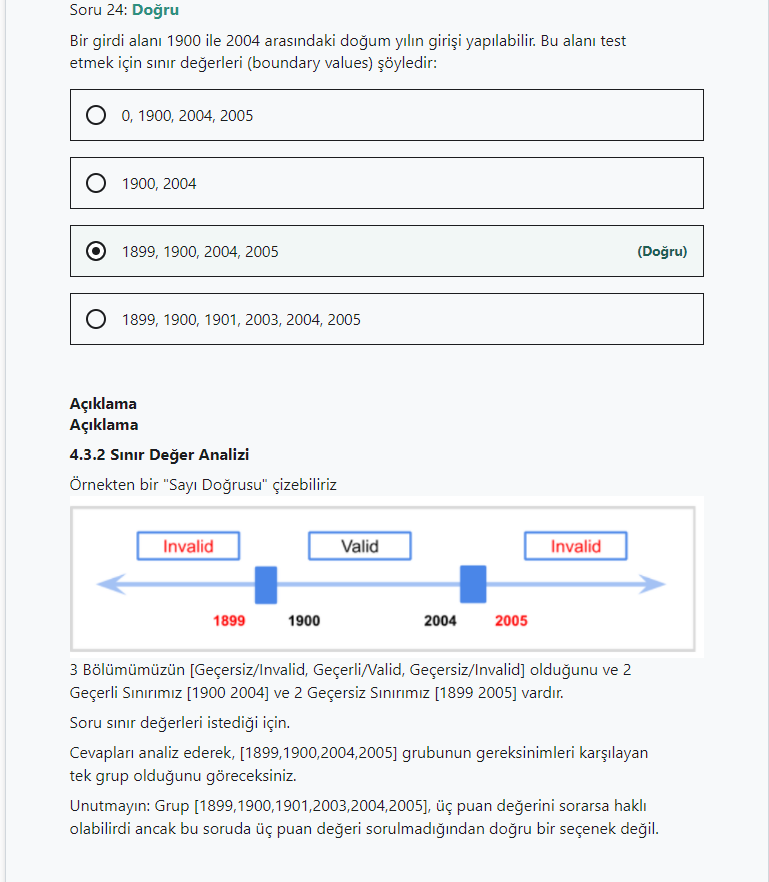
Amaçlanan fonksiyonlarını gerçekleştirmeyen yazılım - Bu bir ürün riskidir

Başarısız olan performance testi - Bu bir ürün riskidir

Geliştirme/Yazılım yöneticisiyle ilgili olan anlaşmazlık sonucu tüm hata raporlarını reddetmesine - Proje riskidir.







****İnsan Hataları, Hatalar ve Arızalar (Errors, Defects, and Failures)****

Bir kişi bir ****hata (yanlış)**** yapabilir. Bu da yazılım kodunda veya diğer bazı ilgili çalışma ürünlerinde bir ****hatanın (kusur veya bug)**** ortaya çıkmasına neden olabilir.

Kodda bir ****hata**** yürütülürse, bu bir ****arıza****ya neden olabilir, ancak bunun her zaman olması gerekmez. Örneğin, bazı hatalar (defects), nadiren veya hiçbir zaman meydana gelmeyen bir arızayı tetiklemek için oldukça kendine özgü girdiler veya ön koşullar gerektirir.

****İnsan hatası/hata = Yanlış [Error = Mistake]****

****Hata = Kusur = Bug [Defect = Fault = Bug]****

Açıklamalı sözlüğe göre:

" ****Hata [Defect]**** ": Bir iş ürününde, o ürünün gereksinimlerini veya özelliklerini karşılamayan bir kusur veya eksiklik olarak tanımlanır.

" ****İnsan Hatası/Hata [Error]**** ": Doğru olmayan bir sonuç üreten bir insan eylemi olarak tanımlanır.

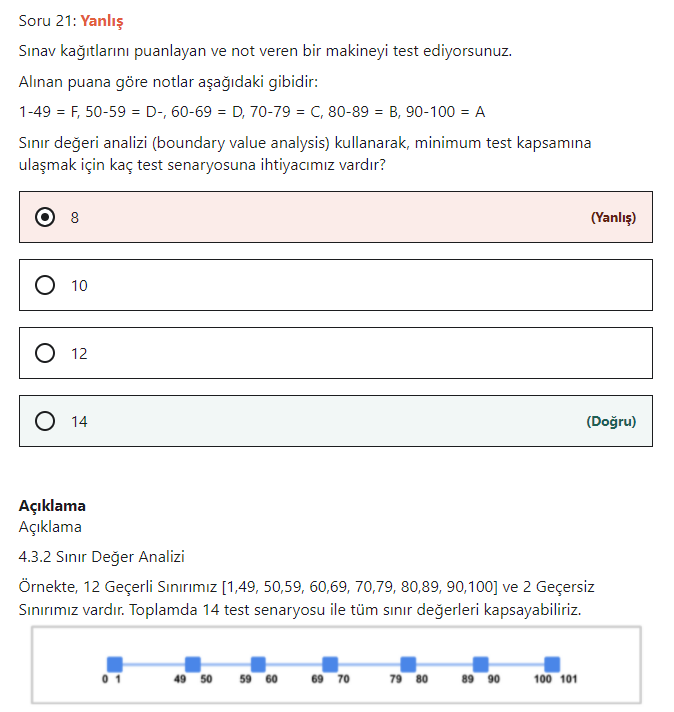
" ****Arıza [Failure]**** ": Bir bileşenin veya sistemin belirli sınırlar dahilinde gerekli bir işlevi yerine getirmediği olay olarak tanımlanır.

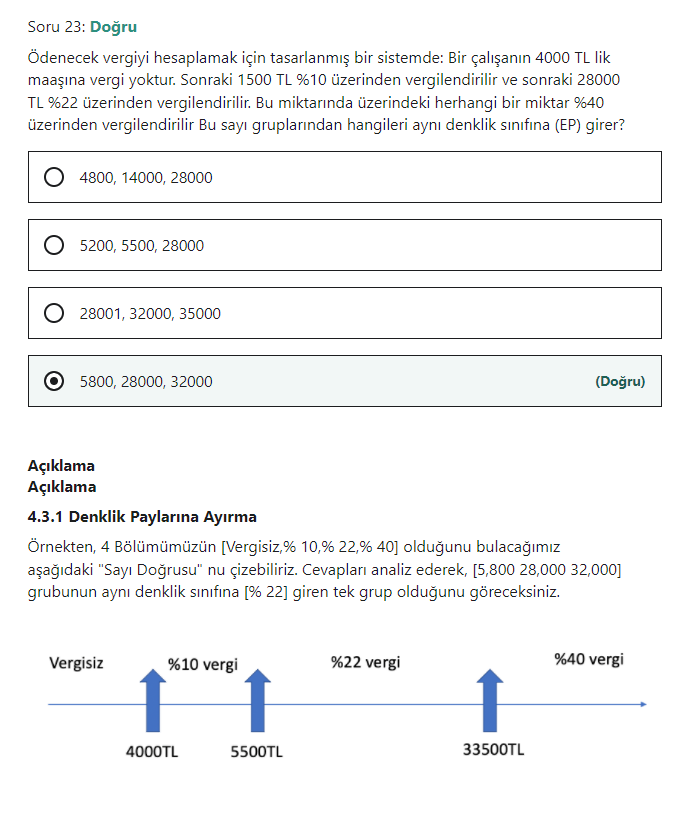
****Sıralı modellerde**** (Şelale ve V-modeli), kabul testi yalnızca ****bir kez**** yapılır.

****Şelale modelinde**** kabul testi tüm geliştirme yapıldıktan sonra gerçekleşir.

****V-modelinde****, planlama kabul testleri projenin başlangıcında, uygulanması ise sonunda gerçekleşir.

****Çevik, yinelemeli ve spiral modellerde****, tüm test türleri yaşam döngüsü boyunca birçok kez gerçekleşir.





****Fonksiyonu Test Etme (Fonksiyonel Test)****

Bir sistemin, alt sistemin veya bileşenin gerçekleştirmesi gereken fonksiyonlar, gereksinimler, kullanım senaryoları veya bir fonksiyonel spesifikasyon gibi kriterler için tanımlanabilir. Fonksiyonlar, sistemin "yaptıklarıdır". Fonksiyonel testler, fonksiyonlara ve özelliklere (dokümanlarda tanımlanan veya test uzmanları tarafından görüşmeler sonucunda elde edilmiş) ve bunların belirli sistemlerle birlikte çalışabilirliğine dayanır. Tüm test seviyelerinde gerçekleştirilebilir.

****2.3.2 Fonksiyonel Olmayan Gereksinimleri Test Etme (Fonksiyonel Olmayan Test)****

Fonksiyonel olmayan test, performans, yükleme, stres, kullanılabilirlik, sürdürülebilirlik, güvenilirlik ve taşınabilirlik gibi testleri içerir fakat bunlarla sınırlı değildir. Bu, yazılımın "nasıl" çalıştığını gösteren bir testtir.

****3.2.2 Roller ve Sorumluluklar****

****Yönetici****: Önerilen değerlendirmelerin uygulanıp uygulanmayacağına karar verir

****Moderatör****: Doküman veya doküman grubunun gözden geçirilme işlemini, gözden geçirmenin planlanmasını, toplantının gerçekleştirilmesini ve toplantı sonrasında takiplerin yapılmasını yöneten kişidir.

****Yazar****: Gözden geçirilecek dokümanların sahibi veya bunlardan sorumlu olan kişidir.

****Gözden geçiriciler:**** Gerekli hazırlıklardan sonra gözden geçirilen dokümandaki bulguları (örn. hatalar)

****Katip (veya kaydedici):**** Toplantı sırasında tanımlanan tüm sorunları, problemleri ve açık noktaları kayıt altına alır.