ISTQB Chapter 2

Testing Throughout the Software Development Lifecycle

- **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.
  • Test uzmanları, gereksinimleri ve tasarımı belirlemek ve iyileştirmek için tartışmalara katılır ve taslaklar hazırlanır hazırlanmaz çalışma ürünlerinin (örneğin; gereksinimler, tasarım, kullanıcı hikâyeleri vb.) gözden geçirilmesinde yer alır

1.Sirali yazilim gelistirme modelleri - Sequential development models

2.Döngüsel ve artımlı yazılım geliştirme modelleri - Iterative and incremental development models

Bu, yazılım geliştirme sürecinde her aşamanın önceki aşama tamamlandığında başlaması gerektiği anlamına gelir—> sirali.

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.

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.

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.

Artımlı yazılım geliştirme modelleri ise, parçalar halinde gereksinimleri belirlemeyi, sistemi tasarlamayı, oluşturmayı ve test etmeyi içerir; bu, yazılımın özelliklerinin adım adım artması 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.

Bazı örnekleri şunlardır:
• 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. iterative***
• 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. iterative***
- V-Model ve Waterfall modeli, sequential modellerdir.***(ardisik)

Ek olarak, yazılım geliştirme yaşam döngüsü modellerinin kendileri de birleştirilebilir. Örneğin, front-end kullanıcı arayüzünü (UI) ve fonksiyonalitesini geliştirmek ve test etmek için çevik yazılım geliştirme modeli kullanılabilirken, backend sistemlerinin geliştirilmesi ve test edilmesi ve entegrasyonu için V-modeli kullanılabilir.

Agile—>Iterative and Incremental (Yinemeli ve artimli)

Test Seviyeleri

1.Birim Testi—Component Testing

hedefler;

- Riskin azaltılması
  • Birimin fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması
  • Birimin kalitesine dair güven oluşturulması
  • Birimdeki hataların bulunması
  • Hataların gözden kaçarak daha üst test seviyelerine girmesinin önlenmesi

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.

“Component testing is often done in isolation from the rest of the system, depending on the software development lifecycle model and the system, which may require mock objects, service virtualization, harnesses, ***stubs****, and* ***drivers****.”****

stubs: Task

2.Entegrasyon Testi

hedefler;

- Riskin azaltılması
  • Arayüzlerin **fonksiyonel ve fonksiyonel olmayan*** ** davranışlarının tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması
  • Arayüzlerin kalitesine dair güvenin oluşturulması
  • Hataların bulunması (arayüzlerin kendisinde, birimlerde veya sistemlerde olabilir)
  • Hataların daha üst test seviyelerine kaçmasının önlenmesi

Birim entegrasyonu testleri, entegre birimler arasındaki etkileşimlere ve arayüzlere odaklanır. Birim entegrasyonu testleri, birim testlerinden sonra gerçekleştirilir ve genellikle otomatikleştirilmiştir.

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

Birim ve sistem entegrasyon testleri entegrasyonun kendisine odaklanmalıdır. Örneğin A modülü ile B modülü entegre ediliyorsa, testler modüllerin ayrı olarak fonksiyonalitesine değil, modüller arasındaki iletişime odaklanmalıdır; çünkü modüllerin fonksiyonalite testi, birim testleri sırasında ele alınmış olmalıdır.

Birim entegrasyon testleri genellikle yazılımcıların sorumluluğundadır. Sistem entegrasyon testleri ise genellikle test uzmanlarının sorumluluğundadır.

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.

Which ADDITIONAL test level could be introduced into a standard V-model after system testing?—→System Integration Testing

A top-down development strategy affects which level of testing most?—>Integrasyon testi

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.

hedefler;

- Riskin azaltılması
  • Sistemin fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimlerde tanımlandığı gibi olup olmadığının doğrulanması
  • Sistemin tamamlandığının ve beklendiği gibi çalışacağının sağlamasının yapılması
  • Bir bütün olarak sistemin kalitesine dair güven oluşturulması
  • Hataların bulunması
  • Hataların daha üst test seviyelerine veya canlıya kaçmasının önlenmesi

Sistem testleri genellikle paydaşlar tarafından sürüm kararları almak için kullanılan bilgileri üretir. Sistem testleri ayrıca yasal veya sözleşmeye dayalı gereksinimleri veya standartları da karşılayabilir.

Sistem testlerini genellikle bağımsız test uzmanları gerçekleştirir.

Bileşen testi için test senaryoları genellikle bileşen spesifikasyonlarından, tasarım spesifikasyonlarından veya veri modellerinden türetilirken, sistem testi için test senaryoları genellikle gereksinim spesifikasyonlarından, fonksiyonel spesifikasyonlardan veya kullanım senaryolarından türetilir.

4.Kabul Testi

hedefleri;

- Bir bütün olarak sistemin kalitesine dair güven oluşturulması
  • Sistemin tamamlandığının ve beklendiği gibi çalışacağının sağlamasının yapılması

• Sistemin fonksiyonel ve fonksiyonel olmayan davranışlarının gereksinimlerde belirtildiği gibi olup olmadığının doğrulanması.

Kabul testleri, sistemin müşteriye (son kullanıcı) çıkmaya ve kullanıma hazır olduğunu değerlendirmeye yönelik bilgiler elde etmek için yapılır.

KULLANICI KABUL TESTLERI

Sistemin kabul testlerinin kullanıcılar tarafından yapılmasıdır.

Temel amaç, kullanıcıların ihtiyaçlarını karşılamak, gereksinimleri yerine getirmek ve istenilen iş süreçlerini gerçekleştirmek için sistemi asgari zorluk, maliyet ve riskle kullanabileceklerine dair güven oluşturmaktır.

OPERASYONEL KABUL TESTLERI

Operasyonel kabul testleri operatörler veya sistem yöneticileri tarafından, genellikle simüle edilmiş canlı ortamda gerçekleştirilir. Testler operasyonel konulara odaklanır ve aşağıdakileri içerebilir:
• Yedekleme ve geri yükleme testleri
• Kurma, kaldırma ve yükseltme
• Felaket kurtarma
• Kullanıcı yönetimi
• Bakım işleri
• Veri yükleme ve taşıma işleri
• Güvenlik açıkları için kontroller
• Performans testleri

sistemi istisnai veya zorlu koşullar altında bile düzgün bir şekilde çalışmaya devam ettirebileceği konusunda güven oluşturmak.

SÖZLESMEYE DAYALI VE YASAL KABUL TESTLERI

Sözleşmeye dayalı kabul testleri, özel olarak geliştirilmiş bir yazılım üretmek için sözleşmenin kabul kriterlerine göre gerçekleştirilir. Sözleşme kabul testleri genellikle kullanıcılar veya bağımsız test uzmanları tarafından gerçekleştirilir.

ALFA VE BETA TESTLERI

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.

System integration testing is executed by the testers. Acceptance testing is done by the customer***

Test Çeşitleri

1. Fonksiyonel Testler - Functional Testing

Fonksiyonel testler tüm test seviyelerinde yapılmalıdır.

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

2.Fonksiyonel olmayan Testler -Non-functional Testing

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

Yaygın yanlış bilgilerin aksine fonksiyonel olmayan testler tüm test seviyelerinde ve sıkça gerçekleştirilmeli ve mümkün olduğunca erken yapılmalıdır. Fonksiyonel olmayan hataların geç fark edilmesi bir projenin başarısı için son derece tehlikeli olabilir.

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

***Performans ve kullanılabilirlik dahil olmak üzere sistemin kalite özelliklerinin test edilmesi

***Testing system attributes, such as usability, reliability or maintainability.

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 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.

4 Değişiklikle İlgili Testler - Change-related Testing

- Onaylama testleri-Confirmation testing: Bir hata çözüldükten sonra, hata nedeniyle başarısız olmuş tüm test senaryoları tekrar test edilebilir; bu testler yazılımın yeni versiyonunda yeniden koşturulmalıdır. Hatanın fonksiyonalite eksikliğinden kaynaklanması durumunda, yazılımın test edilmesi için yeni testler de yazılabilir. 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. Bu istenmeyen yan etkilere regresyon denir. Regresyon testleri, bu gibi istenmeyen yan etkileri bulmak için yapılan testleri içerir.

Özellikle döngüsel ve artımlı yazılım geliştirme yaşam döngülerinde (ör. Çevik), yeni özellikler, mevcut özelliklerde yapılan değişiklikler ve kod yeniden düzenlemesi (refactoring), kodda sık sık değişiklik yapılmasıyla sonuçlanır; bu da değişiklikle ilgili testler gerektirir.

***Re-testing is running a test again; regression testing looks for unexpected side effects

Bakim Testleri

Üretim ortamlarına alındıktan sonra yazılım ve sistemlerin bakımının yapılması gerekir.

Bakımın bir parçası olarak herhangi bir değişiklik yapıldığında, hem değişikliklerin ne kadar başarılı olduğunu değerlendirmek hem de sistemin değişmeyen kısımlarındaki (genellikle sistemin büyük kısmıdır) olası yan etkileri (ör. regresyonlar) kontrol etmek için bakım testleri yapılmalıdır.