

**Hazırlayanın Notu :** Bu Syllabus ISTQB'ye Türkçe hazırlananlar için yapılmış bir çalışmadır. İngilizceden Türkçe'ye çeviri esnasında Teknik tabirler mümkün olduğunca orjinal haliyle bırakılmıştır. Örneğin Syllabus kelimesi yerine ders özeti veya müfredat kelimeleri kullanmaktansa orjinalini korumayı tercih ettim. Aynı şekilde tam karşılığının tek kelime olarak türkçede olmadığını düşündüğüm Testing kelimesini de İngilizce aslı ile kullandım. Yine error, failure, defect gibi Türkçe karşılıkları benzer olan kelimelerde Türkçe çevirinin yanına İngilizce orjinal halini de eklemeyi tercih ettim. Yani bu çeviri profesyonel bir İngilizce-Türkçe çeviri olmaktan ziyade kendi derslerimde de yaptığım üzere teknik terimlerin İngilizce orijinal halleri kullanılarak yapılmış bir tercümedir. Ticari hiç bir fayda gözetmeksizin, kendime göre yapılmış bir tercümedir. Yanlıslarım varsa bana yazmanız durumunda seve seve düzeltceğimi de bilmenizi isterim.

**İngilizce syllabus** üzerinden vakit buldukça bu çalışmayı yapmaya devam edip sizlerle paylaşacağım.

## 0 Introduction

### 0.1 Syllabus'un amacı

Bu syllabus, Temel Düzeyde Uluslararası Yazılım Test Kalifikasyonunun standartlarını belirler. ISTQB® bu syllabus'u aşağıdaki amaçları gerçekleştirmek için oluşturmıştır.

1. Üye kurulların, yerel dillerine tercüme yapması ve eğitim sağlayıcıları akredite etmesi. Üye kurullar, müfredatı kendi dil ihtiyaçlarına göre uyarlayabilir ve yerel yayınlarına uyarlamak için referanslar ekleyebilir.
2. Belgelendirme kuruluşlarının, bu müfredatın öğrenme hedeflerine uyarlanmış kendi yerel dillerinde sınav soruları türetmesi.
3. Eğitim sağlayıcıların, eğitim materyalleri üretmesi ve uygun öğretim yöntemlerini belirlemesi
4. Sertifika adaylarının, sertifika sınavına hazırlanması (bir eğitim kursunun parçası veya bağımsız olarak)
5. Uluslararası yazılım ve sistem mühendisliği topluluğunun, yazılım ve sistem testi mesleğini ilerletmesi ve yazılacak kitap ve makalelere bir temel oluşturmaları

ISTQB®, önceden yazılı izin almak koşuluyla, diğer kuruluşların bu müfredatı başka amaçlar için kullanmasına izin verebilir.

### 0.2 Software Testing için Foundation Level Sertifikası

Foundation Level yeterliliği, yazılım testinde yer alan herkese yöneliktir. Bu kalifikasyon tester'larla birlikte test analyst, test engineer, test consultant, test manager, user acceptance tester ve software developerları da içerir. Ayrıca bu kalifikasyon software test'leri konusunda temel bir bilgi edinmek isteyen product owner, project manager, quality manager, software development manager, business analyst, IT director veya yönetim danışmanları için de uygundur. Temel seviyedeki bu sertifikaya sahip olanlar üst düzey yazılım sertifikaları için de müracaat edebilirler.

Bu Temel Müfredat 2018 V3.1 aşağıdaki bilgileri içerir:

- Müfredat için ticari sonuçlar
- Ticari sonuçlar ve öğrenme hedefleri arasındaki izlenebilirliği gösteren matris
- Bu syllabus'un özeti

### 0.3 Ölçümlendirilebilir Öğrenme Hedefleri ve Bilişsel Bilgi Düzeyleri

Öğrenme hedefleri, ticari sonuçları destekler ve Sertifikalı Test Uzmanı Temel Düzey sınavlarını oluşturmak için kullanılır. Bu syllabus içeriği (giris ve ekler hariç) K1 seviyesinde test için gereklidir.

Yapılan sınavda adaylardan syllabus icerisindeki alti bolumde anlatılan konulardan herhangi bir keyword'u hatirlaması veya tanımlaması istenebilir. Ogrenme hedef seviyeleri her bolumun basında belirtilmekle birlikte asagidaki uc seviye olarak siniflandırılmıştır.

- K1: hatırlama
- K2: anlama
- K3: uygulama

Öğrenme hedeflerine ilişkin daha fazla ayrıntı ve örnekler Ek B'de verilmiştir.

Oğrenme hedeflerinde acik olarak yazmasa bile K1 seviyesi için bölüm başlıklarının hemen altında anahtar kelime olarak listelenen tüm terimlerin tanımları hatırlanmalıdır .

## 0. 4 Foundation Level Sertifika Sınavı

Foundation Level sertifika sınavı bu syllabus iceriginden yapılacaktır. Sınav sorularının cevapları, bu syllabus'un birden fazla bölümüne dayalı materyal kullanımını gerektirebilir. Sınav'da giriş ve ekler haric tüm bolumlerden soru gelebilir. Standartlar, kitaplar ve diğer ISTQB® müfredatları referans olarak dahil edilmiş olsa da bu syllabus'da özet olarak verilen kisimler haric o kaynaklardan soru sorulmayacaktır.

Sınav, çoktan seçmeli 40 sorudan oluşmaktadır. Sınavı geçmek için soruların en az %65'inin (yani 26 sorunun) doğru cevaplanması gerekir.

Sınava girmek için bir kursa dahil olma mecburiyeti yoktur.

## 0. 5 Akreditasyon

Bir ISTQB® Kurulu, ders materyali bu müfredatı takip eden eğitim sağlayıcılarını akredite edebilir. Eğitim sağlayıcılar, akreditasyonu gerçekleştiren Üye Kurul veya organdan akreditasyon yönergeleri almalıdır. Akredite edilmiş bir kurs, bu müfredata uygun olarak kabul edilir ve kursun bir parçası olarak bir ISTQB® sınavına girmesine izin verilir.

## 0. 6 Ayrıntı Seviyesi

Bu syllabus'daki ayrıntı düzeyi, uluslararası düzeyde kabul edilmiş kurslar ve sınavlar gozonunde bulundurulurak belirlenmiştir. Bu amaca ulaşmak için syllabus şunlardan oluşur:

- Foundation Level'in amacını açıklayan genel öğretim hedefleri
- Öğrencilerin hatırlaması gereken terimlerin bir listesi
- Ulaşılabilecek bilişsel öğrenme sonucunu tanımlayan, her bilgi alanı için öğrenme hedefleri
- Kabul edilen literatür ve standartlar gibi kaynaklara yapılan referanslar dahil olmak üzere temel kavramların tanımı

Bu syllabus, yazılım testi ile ilgili tüm konuların açıklanmasını değil, Foundation Level eğitim kurslarında kapsanacak konuları ve ayrıntı düzeyini içermektedir. Agile metodolojisine uygun yürütülen projeler de dahil olmak üzere tüm yazılım projelerine uygulanabilecek test kavram ve tekniklerine odaklanır. Bu syllabus, belirli bir yazılım geliştirme yaşam döngüsü veya yöntemiyle ilgili herhangi bir özel öğrenme hedefi içermez, ancak bu kavramların Agile projelerde veya diğer yinelemeli(iterative) ve artımlı(incremental) yaşam döngülerinde ve sıralı(sequential) yaşam döngülerinde nasıl uygulanacağını tartışır.

## 0.7 Bu Syllabus Hangi Bolumlerden Olusur?

Syllabus sınavda soru çıkacak içeriğe sahip altı bölüme ayrılmıştır. Her bölümün üst düzey başlığı, bölümün zamanını belirtir. Akredite eğitim kursları için, müfredat, aşağıdaki gibi altı bölüme dağıtılmış en az 16.75 saatlik eğitim gerektirir:

- Bölüm 1: 175 dakika Testing Temel Bilgileri
- Bölüm 2: 100 dakika Yazılım Geliştirme Yaşam Döngüsü ve Testing
- Bölüm 3: 135 dakika Statik Testing
- Bölüm 4: 330 dakika Test Teknikleri
- Bölüm 5: 225 dakika Test Yönetimi
- Bölüm 6: 40 dakika Testing için Tool Desteği

### 1 Testing'in Temelleri

175 minutes

coverage, debugging, defect, error, failure, quality, quality assurance, root cause, test analysis, test basis, test case, test completion, test condition, test control, test data, test design, test execution, test implementation, test monitoring, test object, test objective, test oracle, test planning, test procedure, test process, test suite, testing, testware, traceability, validation, verification

Testing'in Temelleri bolumu ogrenme hedefleri:

#### 1.1 Testing nedir?

FL-1.1.1 (K1) Testing'in hedeflerini tanımlama

FL-1.1.2 (K2) Testing ve debugging karsilastirilmesi

#### 1.2 Testing Nicin Gereklidir?

FL-1.2.1 (K2) Testing'in nicin gerekli olduguna dair ornekler

FL-1.2.2 (K2) Testing and quality assurance(Kalite Guvence) arasindaki iliski ve Testing'in daha iyi bir kaliteye ulasmadaki rolunu gosteren ornekler

FL-1.2.3 (K2) Error, defect ve failure karsilastirilmesi

FL-1.2.4 (K2) Bir defect'i olusturan temel neden ve etkilerini tanımlama

#### 1.3 Testing'in Yedi Prensibi

FL-1.3.1 (K2) Testing'in Yedi Prensibinin aciklanmasi

#### 1.4 Test Asamalari

FL-1.4.1 (K2) Context'in test surecine etkileri

FL-1.4.2 (K2) Test asamalari boyunca yapilan test faaliyetlerini ve ilgili gorevleri tanımlayın

FL-1.4.3 (K2) Test sureclerinde kullanılan work product (calisma urunleri)'nin tanımlanmasi

FL-1.4.4 (K2) Testin temel hedefi ile kullanılan work product arasinda izlenebilirliğin onemi

#### 1.5 Testing Psikolojisi

FL-1.5.1 (K1) Testin başarısını etkileyen psikolojik faktörleri hatırlama

FL-1.5.2 (K2) Bir test uzmanı ile yazılımcının bakış açılarını karşılaştırma

## 1.1 Testing Nedir ?

Yazılım sistemleri, ticari uygulamalardan (örn. bankacılık) tüketici ürünlerine kadar (örn. otomobiller), yaşamın ayrılmaz bir parçasıdır. Çoğu insan, beklediği gibi çalışmayan bir yazılımla karşılaşmıştır. Düzgün çalışmayan yazılımlar para, zaman veya iş itibarı kaybı, hatta yaralanma veya ölüm dahil olmak üzere birçok soruna yol açabilir. Software testing, yazılımın kalitesini değerlendirmenin ve çalışma sırasında yazılım hatası(failure) riskini azaltmanın bir yoludur.

Testing için yaygın ama yanlış bir algı, testing'in yalnızca test amaçlı kodların çalıştırılmasından ve sonuçların kontrol edilmesinden ibaret olduğudur. Bölüm 1.4'te açıklandığı gibi, yazılım testi birçok farklı aktiviteyi içeren bir süreçtir. Testleri çalıştırma(execution) (sonuçların kontrolü de dahil) bu faaliyetlerden sadece biridir. Test süreci(test process) testleri execute etme dışında, test planlama, analiz etme, test tasarlama(designing) ve uygulama(implementing tests), test ilerlemesini ve sonuçlarını raporlama ve yapılan testin kalitesini değerlendirme gibi faaliyetleri de içerir.

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

Testing için var olan başka bir yanlış algı ise, Testing'in tamamen gereksinimlerin(requirements), kullanıcı hikayelerinin(user story) veya diğer spesifikasyonların doğrulanmasına odaklandığıdır. Testing, sistemin belirtilen gereksinimleri karşılayıp karşılamadığını kontrol etmenin yanında, sistemin çalışma ortamında kullanıcı (user) ve diğer paydaşların(stakeholder) ihtiyaçlarını karşılayıp karşılamayacağını kontrol etmeyi de içerir.

Test faaliyetleri, farklı yaşam döngülerinde(life cycle) farklı şekilde düzenlenir ve yürütülür (Bölüm 2.1).

### 1.1.1 Testing'in Temel Amaçları

Herhangi bir proje için testin amaçları şunları içerebilir:

- Gereksinimler, user story, tasarım ve kodlar gibi çalışma ürünlerini(work product) değerlendirerek kusurları önlemek
- Belirtilen tüm gereksinimlerin karşılanıp karşılanmadığını doğrulamak
- Testin tamamlanıp tamamlanmadığını kontrol etmek ve kullanıcılar ve stakeholder'ların beklediği gibi çalışıp çalışmadığını doğrulamak
- Testimizin kalite düzeyine güven oluşturmak
- Kusurları(defects) ve arızaları(failures) bulmak, böylece yazılımın "yetersiz yazılım kalitesine (inadequate software quality)" düşme riskini azaltmak
- Paydaşlara(stakeholder), bilinçli kararlar vermelerini sağlamak için yeterli bilgi vermek, özellikle de testimizin kalite düzeyi ile ilgili bilgiler sağlamak
- Sözleşmeye dayalı, yasal veya düzenleyici gereksinimlere veya standartlara uymak ve/veya testimizin bu tür gereksinimler veya standartlara uygunluğunu doğrulamak

Testin amaçları, test edilen bileşen(component) veya sistemin bağlamına(context), test düzeyine ve yazılım geliştirme yaşam döngüsü(lifecycle) modeline bağlı olarak değişebilir. Bu farklılıklar örneğin şunları içerebilir:

- Bileşen(component) testi sırasında, bir amaç mümkün olduğunca çok sayıda hata bulmak olabilir, böylece altta yatan kusurlar erken tespit edilip düzeltilir. Diğer bir amaç, bileşen testlerinin kodlarının üzerinden geçmek ve kod kalitesini artırmak olabilir.

- Kabul testi(acceptance testing) sırasında, sistemin beklendiği gibi çalıştığını ve gereksinimleri karşıladığını doğrulamak bir amaç olabilir. Bu testin bir başka amacı da , paydaşlara(stakeholder), sistemi belirlenen zamanda yayınlama(release) durumunda oluşabilecek riskler konusunda bilgi vermek olabilir.

### 1.1.2 Testing ve Hata Ayıklama(Debugging)

Testing ve debugging birbirinden farklıdır.

Testlerin yürütülmesi(Execution), yazılımdaki kusurların(defect) neden olduğu arızaları(failure) gösterebilir. Hata ayıklama(debugging) ise, bu tür kusurları(defect) bulan, analiz eden ve düzelten geliştirme(development) etkinliğidir. Sonraki onay testi, düzeltmelerin(fixes) kusurları(defect) çözüp çözmediğini kontrol eder. Bazı durumlarda, geliştiriciler(developers) hata ayıklamayı(debugging), ilişkili bileşen(associated component) ve bileşen entegrasyonu testini(component integration testing) yaparken, test uzmanları(testers) ilk testten(initial test) ve son onay testinden(final confirmation test) sorumludur. Ancak, Agile geliştirmede ve diğer bazı yazılım geliştirme yaşam döngülerinde, tester'lar hata ayıklama(debugging) ve bileşen testine(component testing) dahil olabilir.

Yazılım testi kavramları hakkında daha fazla bilgiye ISO standardı (ISO/IEC/IEEE 29119-1 standardından) ulaşılabilir.

## 1.2 Testing Nicin Gereklidir ?

Bileşenlerin(component), sistemlerin ve bunlarla ilişkili belgelerin titiz bir şekilde test edilmesi, çalışma sırasında meydana gelen arıza(failure) riskinin azaltılmasına yardımcı olabilir. Kusurların(defects) tespit edilip giderilmesi, bileşenlerin veya sistemlerin kalitesine katkıda bulunur. Ayrıca, sözleşmeye bağlı veya yasal gereklilikleri ve sektöre özel standartları karşılamak için de yazılım testi gerekebilir.

### 1.2.1 Testing'in Basariya Katkiları

Bilgi işlem tarihi boyunca, yazılım ve sistemler işleme alındıktan sonra kusurların varlığı(defect) nedeniyle, arızalara(failure) neden olması veya bir şekilde paydaşların(stakeholders) ihtiyaçlarını karşılamaması oldukça yaygındır. Ancak uygun test teknikleri, gerekli test uzmanlığı düzeyinde, uygun test seviyelerinde ve yazılım geliştirme yaşam döngüsünün uygun noktalarında uygulandığında bu tür sorunların sıklığını azaltabilir. Örnekler şunları içerir:

- Gereksinimlerin(requirements) gözden geçirilmesine veya user story'nin iyileştirilmesine tester'ların dahil edilmesi bu work product'daki kusurları tespit edilmesine katkı sağlar. Gereksinim kusurlarının tanımlanması ve kaldırılması, yanlış veya gereksiz özelliklerin geliştirilmesi riskini azaltır.
- Sistem tasarlanırken tester'ların sistem tasarımcıları ile yakın bir şekilde çalışması, her iki tarafın birbirinin yaptığı işlemleri daha iyi anlamasını sağlayabilir. Bu artan anlayış, temel tasarım kusurları riskini azaltabilir ve testlerin daha erken bir aşamada tanımlanmasını sağlayabilir.
- Kod geliştirme aşamasındayken tester'ların developer'lar ile yakın bir şekilde çalışması, tarafların kodu ve kodun nasıl test edileceğini daha iyi anlamasını sağlayabilir. Bu artan anlayış, kod ve testler içindeki kusur(defect) riskini azaltabilir.
- Test edenlerin yazılımı piyasaya sürmeden önce doğrulamasını(verify) ve onaylamasını(validate) sağlamak, bunlar yapılmadığında gözden kaçabilecek hataları tespit edebilir ve hatalara(failure) neden olan kusurların(defects) kaldırılması sürecini destekleyebilir(Ornegin debugging). Bu da, yazılımın paydaş(stakeholders) ihtiyaçlarını karşılama ve gereksinimleri karşılama olasılığını artırır.

Bu örneklerle ek olarak, tanımlanmış test hedeflerine (bkz. bölüm 1.1.1) ulaşılması, genel yazılım geliştirme(development) ve bakım(maintenance) başarısına katkıda bulunur.

### 1.2.2 Kalite Guvencesi(Quality Assurance) ve Testing

İnsanlar testing'e atıfta bulunmak için genellikle kalite güvencesi (veya sadece KG / QA) ifadesini kullansa da, kalite güvencesi ve test aynı değildir, ancak ilişkilidir. Daha kapsamlı bir kavram olan **kalite yönetimi** onları birbirine bağlar. Kalite yönetimi, bir kuruluşu kalite açısından yönlendiren ve kontrol

eden tüm faaliyetleri içerir. Diğer faaliyetlerinin yanı sıra kalite yönetimi, hem kalite güvencesini hem de kalite kontrolünü içerir. Kalite güvencesi, uygun kalite seviyelerine ulaşılacağına dair güven sağlamak için tipik olarak uygun süreçlere bağlı kalmaya odaklanır. Prosesler düzgün bir şekilde yürütüldüğünde, bu prosesler tarafından oluşturulan iş ürünleri(work products) genellikle daha yüksek kalitede olur ve bu da kusurların önlenmesine katkıda bulunur. Ek olarak, kusurların nedenlerini tespit etmek ve ortadan kaldırmak için kök neden(root cause) analizinin kullanılması ve süreçleri iyileştirmek için geriye dönük toplantılarındaki(retrospective meetings) tespitlerin doğru uygulanması, etkin kalite güvencesi için önemlidir.

Kalite kontrol, ürünün uygun kalite seviyelerine ulaşmasını destekleyen test faaliyetleri de dahil olmak üzere çeşitli faaliyetleri içerir. Test faaliyetleri, genel yazılım geliştirme(development) veya bakım(maintenance) sürecinin bir parçasıdır. Kalite güvencesi, tüm sürecin uygun şekilde yürütülmesiyle ilgili olduğundan, kalite güvencesi, uygun testler yapılmasını destekler. 1.1.1 ve 1.2.1 bölümlerinde açıklandığı gibi, test etme, kalitenin elde edilmesine çeşitli şekillerde katkıda bulunur.

### 1.2.3 Hata(Error), Kusur(Defect) ve Arıza(Failure)

Bir kişi bir hata (error / mistake) yapabilir ve bu da yazılım kodunda veya diğer ilgili iş ürününde bir kusurun(defect) (arıza/fault veya hata/bag) ortaya çıkmasına neden olabilir. Bir iş ürününde bir kusurun(defect) ortaya çıkmasına neden olan bir hata(error), ilgili başka bir iş ürününde bir kusurun ortaya çıkmasına yol açan başka bir hatayı tetikleyebilir. Örneğin, bir gereksinim belirleme hatası gereksinim hatasına yol açabilir ve bu da kodda bir hataya(defect) yol açan bir programlama hatasıyla(error) sonuçlanır.

Koddaki bir kusur calistirilirs(execute), her zaman olmasa da bir arizaya(failure) neden olabilir. Örneğin, bazı kusurlar(defect), nadiren veya hiç meydana gelmeyebilecek bir arızayı(failure) tetiklemek için çok özel girdiler(input) veya ön koşullar(precondition) gerektirir.

Hatalar(Error), aşağıdakiler gibi birçok nedenden dolayı oluşabilir:

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

Koddaki kusurlardan(defects) kaynaklanan arızalara(failure) ek olarak, arızalar çevresel koşullardan da kaynaklanabilir. Örneğin radyasyon, elektromanyetik alanlar ve kirlilik, aygıt yazılımında kusurlara neden olabilir veya donanım koşullarını değiştirerek yazılımın çalışmasını(execute) etkileyebilir.

Beklenmeyen(unexpected) test sonuçlarının tümü başarısızlık(failure) değildir. Testlerimizin görevi hatayı bulmak olsa da bazen testler hatalı bir urundeki hatayı tespit edemeyip, hatasız ürün sonucu verebilir(False positives). 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 hatasiz olan bir urun hatali olarak tanimlanabilir(false negatives).

#### 1.2.4 Kusurlar(Defects), Temel Nedenler(Root Causes) ve Etkiler(Effects)

Kusurların(defect) temel nedenleri(root causes), kusurların oluşmasına katkıda bulunan ilk eylem veya koşullardır. Kusurlar, gelecekte benzer kusurların oluşmasını azaltmak için temel nedenlerini tespit etmek amacıyla analiz edilebilir. Kök neden analizi, en önemli kök nedenlere odaklanarak, gelecekte önemli sayıda kusurun ortaya çıkmasını önleyen süreç iyileştirmelerine imkan tanır.

Örneğin, tek bir yanlış kod satırı nedeniyle yanlış faiz ödemelerinin müşteri şikayetleriyle sonuçlandığını varsayalım. Arızalı(defective) kod, product owner'ın faizin nasıl hesaplanacağını yanlış anlaması nedeniyle, tam anlaşılamayan bir user story neticesinde yazılmıştır. Faiz hesaplamalarında büyük oranda kusur varsa ve bu kusurların temel nedeni benzer yanlış anlamalara dayanıyorsa, product owner'lar gelecekte bu tür kusurları azaltmak için faiz hesaplamaları konusunda eğitilebilir.

Bu örnekte, müşteri şikayetleri etkilerdir(effect). Yanlış faiz ödemeleri başarısızlıktır(failure). Koddaki yanlış hesaplama bir kusurdur(defect) ve koddaki yanlış hesaplama, orijinal kusurdan yani user story'deki belirsizlikten kaynaklanmaktadır. Orijinal kusurun temel nedeni ise, product owner'ın user story yazarken bir hata yapmasına neden olan bilgi eksikliğiydi. Kök neden analizi süreci hakkında tüm fikirler ISTQB-CTEL-TM ve ISTQB-CTEL-ITP'de karşılaştırılmıştır.

### 1.3 Testing'in Yedi Prensipleri



Soru ile ilgili olduğu için aşağıdaki bölümler eklendi. Normal çalışma yukarıdaki kısımdan devam edecek

#### 2.2.4 Acceptance Testing (Kabul Testi)

Kabul testinin amaçları

Kabul testi(acceptance testing), sistem testi gibi, tipik olarak tüm sistemin veya ürünün(product) davranışına(behaviour) ve yeteneklerine(capability) odaklanır.

Kabul testinin yapılış amaçları aşağıdakileri maddeleri içerir;

- Bir bütün olarak sistemin kalitesine güven oluşturmak
- Sistemin tamamlandığının ve beklendiği gibi çalışacağının doğrulanması
- Sistemin işlevsel(functional) ve işlevsel olmayan (non-functional) davranışlarının belirtildiği gibi olduğunun doğrulanması

Kabul testi, sistemin dağıtım(deployment) ve müşteri (son kullanıcı) tarafından kullanıma hazır olup olmadığını değerlendirmek için bilgi üretebilir. Kabul testi sırasında kusurlar bulunabilir, ancak kusurları bulmak genellikle bir amaç değildir(a) ve kabul testi sırasında önemli sayıda kusur bulmak bazı durumlarda büyük bir proje riski olarak kabul edilebilir.

Kabul testleri, yasal veya düzenleyici gereklilikleri veya standartları da karşılayabilir.

Kabul testinin yaygın biçimleri aşağıdakileri içerir:

- User acceptance testing : Kullanıcı Kabul Testi
- Operational acceptance testing : Operasyonel Kabul Testi
- Contractual and regulatory acceptance testing : Sözleşmeye dayalı ve düzenleyici kabul testleri
- Alpha and beta testing: Alfa ve beta testi.

Her biri aşağıdaki dört alt bölümde açıklanmıştır.

User acceptance testing (UAT) : Kullanıcı Kabul Testi

Sistemin kullanıcı kabul testi, tipik olarak, gerçek veya simüle edilmiş bir işletim ortamında hedef kullanıcılar tarafından sistemin kullanımına uygunluğun doğrulanmasına odaklanır. Temel amaç, kullanıcıların ihtiyaçlarını karşılamak, gereksinimleri karşılamak ve iş süreçlerini minimum zorluk, maliyet ve riskle gerçekleştirmek için sistemi kullanabilecekleri konusunda güven oluşturmaktır.

Spesifik yaklaşımlar ve sorumluluklar

Kabul testi genellikle müşterilerin, iş kullanıcılarının, ürün sahiplerinin veya bir sistemin operatörlerinin sorumluluğundadır ve diğer paydaşlar da dahil olabilir.

Kabul testi genellikle sıralı geliştirme (sequential development ) yaşam döngüsündeki son test düzeyi olarak düşünülür, ancak başka zamanlarda da gerçekleştirilebilir.

Yinelemeli geliştirmede(iterative development), proje ekipleri, kabul kriterlerine göre yeni bir özelliği doğrulamaya odaklananlar ve yeni bir özelliğin kullanıcıların ihtiyaçlarını karşıladığını doğrulamaya odaklananlar gibi, her yineleme sırasında ve sonunda çeşitli kabul testi biçimleri kullanabilir. Ayrıca, alfa testleri ve beta testleri, her yinelemenin sonunda, her yinelemenin tamamlanmasından sonra veya bir dizi yinelemeden sonra gerçekleştirilebilir. Kullanıcı kabul testleri, operasyonel kabul testleri, yasal kabul testleri ve sözleşmeye dayalı kabul testleri de, her yinelemenin sonunda, her yinelemenin tamamlanmasından sonra veya bir dizi yinelemeden sonra gerçekleştirilebilir.