ISTQB Sertifikalı Test Uzmanı Temel Seviye v4.0

International Software Testing Qualifications Board Özet

1.1. Yazılım Testi Nedir?

- Yazılımların doğru çalışmaması ciddi mali kayıplara ve hatta güvenlik risklerine yol açabilir.
- Yazılım testi, hataları bulmak ve yazılımın kalitesini artırmak için yapılan bir süreçtir.
- Test süreci sadece yazılımın çalışıp çalışmadığını değil, aynı zamanda kullanıcı ihtiyaçlarını da kontrol eder.
- Yazılım testleri dinamik (yazılım çalıştırılarak) veya statik (analiz ve gözden geçirme) olabilir.
- Test uzmanları sadece araçları kullanmakla kalmaz, aynı zamanda düşünsel ve analitik becerileri de kullanır.

1.1.1. Test Hedefleri

- Gereksinimleri, tasarımları ve kodu değerlendirme.
- Arızaları bulma ve hataları düzeltme.
- Test kapsamını sağlama.
- Kalite kriterlerini karşılamama riskini azaltma.
- Gereksinimlerin doğruluğunu ve uyumluluğunu kontrol etme.
- Paydaşlara doğru bilgi sunma ve güven oluşturma.
- Ürünün eksiksiz ve beklentilere uygun çalışmasını sağlama.

Test hedefleri, test edilen ürün, test seviyesi, riskler ve izlenen yazılım geliştirme süreci gibi faktörlere bağlı olarak değişebilir. Ayrıca, kurumsal yapı, rekabetçi faktörler ve pazara sürüm süresi gibi ticari faktörler de test hedeflerini etkileyebilir.

1.1.2. Yazılım Testi ve Hata Ayıklama

- Test etme ve hata ayıklama farklı aktivitelerdir.
- Dinamik test, yazılımdaki hataların neden olduğu arızaları tetikleyebilir veya doğrudan hataları bulabilir.
- Hata ayıklama süreci, arızaların nedenlerini bulma, analiz etme ve giderme işlemlerini içerir.
- Hata ayıklama süreci genellikle arızanın yeniden oluşturulması, teşhis ve nedenin düzeltilmesi adımlarını içerir.
- Düzeltmelerin problemi giderip gidermediğini kontrol etmek için onaylama testleri yapılır.
- Onaylama testi, tercihen başlangıçtaki testi gerçekleştiren aynı kişi tarafından yapılır.

- Statik test, bir hata bulduğunda hata ayıklama işlemiyle ilgilenir ve hataları doğrudan bulur. Yeniden oluşturma veya teşhis gerektirmez.

1.2. Yazılım Testi Neden Gereklidir?

- Test etme süreci, belirlenen kapsam, zaman, kalite ve bütçe kısıtları içinde önceden kararlaştırılmış hedeflere ulaşmayı sağlar.

- Bu başarı yalnızca test ekibiyle sınırlı değildir; her paydaş kendi test becerilerini kullanarak projeyi destekleyebilir. Test bileşenleri ve ilgili dokümantasyon, yazılımdaki hataları belirlemeye yardımcı olur.

1.2.1. Yazılım Testinin Başarıya Katkısı

- Test süreci, hataların maliyet etkin bir şekilde bulunmasını sağlar.
- Bulunan hatalar, daha yüksek kaliteli test nesnelerine katkıda bulunmak için giderilebilir.
- Test etme süreci, yazılımın çeşitli aşamalarında kalitesini doğrudan değerlendirme imkanı sunar.
- Test, proje yönetimi kapsamında karar alma süreçlerine katkıda bulunur.

- Test, kullanıcıların dolaylı olarak temsil edilmesini sağlar.
- Yazılım testleri, sözleşmelere bağlı veya yasal gereksinimleri karşılamak için de gereklidir.

1.2.2. Test Etme ve Kalite Güvence (KG)

- Test etme ve kalite güvence (KG) farklı kavramlardır.
- Test etme, bir çeşit kalite kontrolüdür (KK) ve ürün odaklı, düzeltici bir yaklaşımı temsil eder.
- Kalite kontrol, uygun kalite seviyelerine ulaşmayı destekleyen aktivitelere odaklanır.
- Test, kalite kontrolünün önemli bir bileşenidir ve resmi yöntemleri, simülasyonu ve prototiplemeyi içerebilir.
- Kalite güvence (KG), süreç odaklı ve önleyici bir yaklaşımdır.
- KG, süreçlerin uygulanması ve iyileştirilmesine odaklanır ve sonuçta iyi bir ürün ortaya çıkmasına odaklanır.
- Hem yazılım geliştirme hem de test süreci için geçerlidir ve projedeki herkesin sorumluluğundadır.
- Test sonuçları, kalite kontrolü ve kalite güvencesi süreçlerinde kullanılır.
- Kalite kontrolü sürecinde test sonuçları hataların düzeltilmesinde kullanılırken, kalite güvencesi sürecinde yazılım geliştirme ve test süreçlerinin performansı hakkında geri bildirim sağlar.

1.2.3. İnsan Hataları, Hatalar, Arızalar ve Kök Nedenler

- İnsanlar hata yapabilir, bu da yazılım hatalarına neden olabilir ve sonunda başarısızlığa yol açabilir.
- Hatalar, zaman baskısı, iş karmaşıklığı, süreçler, altyapı veya iletişim gibi birçok sebepten kaynaklanabilir.
- Hatalar dokümantasyonda, kaynak kodda veya derleme dosyasında bulunabilir.
- Ürünlerin erken aşamalarında bulunan hatalar, ilerleyen aşamalarda hatalı ürünlerin ortaya çıkmasına neden olabilir.
- Bir hata, sistemde beklenmeyen davranışlara neden olabilir.
- Arızaların tek nedeni insan ve yazılım hataları değildir; çevresel koşullar da etkileyebilir.
- Kök neden, bir problemin ortaya çıkmasının temel sebebidir ve kök neden analizi ile belirlenir.
- Kök nedenin ortadan kaldırılması, benzer arızaların veya hataların önlenebilmesini veya sıklıklarının azaltılmasını sağlar.

1.3. Test Prensipleri

- **Testin Amacı:** Testlerin amacı, yazılımda hataların varlığını göstermektir. Ancak, hiçbir test, yazılımdaki tüm hataları bulamaz veya hata olmadığını kanıtlayamaz. Testler, keşfedilmemiş hataların kalma olasılığını azaltır, ancak tam bir hata olmama garantisi vermez.
- **%100 Test Mümkün Değildir:** Tüm yazılım unsurlarını test etmek genellikle mümkün değildir. Test teknikleri, test senaryosu önceliklendirmesi ve risk bazlı test yaklaşımı gibi stratejiler kullanılarak testten daha fazla verim alınabilir.
- **Erken Test:** Erken aşamalarda hataları bulmak, zaman ve para tasarrufu sağlar. Erken bulunan hatalar, yazılımın sonraki aşamalarında daha az arıza olasılığına sahip olmasını sağlar.
- **Hataların Yoğunlaştığı Alanlar:** Bulunan hatalar genellikle belirli sistem bileşenlerinde yoğunlaşır. Bu durum, risk-bazlı test için önemli bir kriterdir.
- **Antibiyotik Direnci:** Aynı testler tekrarlandıkça, yeni hataları bulma etkinliği azalır. Bu durumu önlemek için testlerin ve test verilerinin düzenlenmesi gerekebilir.
- **Testin Bağlama Göre Değişmesi:** Test, projenin bağlamına ve koşullarına göre değişir. Evrensel olarak uygulanan tek bir test yaklaşımı yoktur.
- **Yanılgılar:** Testin tüm gereksinimleri titizlikle test etmek ve bulunan tüm hataları çözmek, başarılı bir yazılım elde edildiği anlamına gelmez. Test, sadece doğrulamanın yanı sıra sağlamayı da içermelidir.

1.4. Test Aktiviteleri, Test Yazılımı ve Test Rolleri

- Test süreci, belirli bir duruma bağlı olarak özelleştirilebilir ve test planlaması kapsamında hangi test aktivitelerinin dahil edileceği, nasıl uygulanacağı ve ne zaman gerçekleştirileceği belirlenir.
- Test sürecini oluşturan genel test aktivitesi grupları, test hedeflerine ulaşma olasılığını artırır.
- Test süreci, çeşitli faktörlere dayalı olarak belirlenir ve ISO/IEC/IEEE 29119-2 standardı gibi kaynaklar daha fazla bilgi sağlar.

- Test aktiviteleri ve görevler, bağlamın etkisi, test yazılımı, test esası ile test yazılımı arasındaki izlenebilirlik ve test rolleri gibi genel yönler açıklanır.

1.4.1. Test Aktiviteleri ve Görevleri

Test süreci genellikle aşağıdaki ana aktivite gruplarından oluşur ve bu aktiviteler tekrarlanabilir veya paralel olarak uygulanabilir:

1. **Test Planlama**: Test hedeflerini belirleyip, en iyi ulaşım yöntemini seçmek.
2. **Test Gözetimi ve Kontrolü**: Tüm test aktivitelerini izlemek ve planlanan ile gerçekleşenin karşılaştırılması.
3. **Test Analizi**: Test edilebilir özellikleri tanımlamak, test koşullarını belirlemek ve önceliklendirmek.
4. **Test Tasarımı**: Test koşullarını test senaryoları ve diğer test yazılımları halinde detaylandırmak.
5. **Test Uyarlama**: Test koşumu için gerekli test yazılımını oluşturmak veya edinmek.
6. **Test Koşumu**: Testleri test koşum çizelgesine göre çalıştırmak, manuel veya otomatik olarak.
7. **Test Tamamlama**: Çözülmemiş hataların ele alınması, test ortamının kapatılması ve test tamamlama raporunun oluşturulması.

Bu aktiviteler, test sürecinin belirli bir duruma göre özelleştirilebilmesini ve test planlaması kapsamında karar verilebilmesini sağlar.

1.4.2. Proje Bağlamında Test Süreci

Testler, organizasyon içinde gerçekleştirilen yazılım geliştirme süreçlerinin ayrılmaz bir parçasıdır ve proje paydaşları tarafından finanse edilir.

Testlerin yapılma şekli birçok bağlamsal faktöre bağlıdır, bunlar:

1. **Paydaşlar**: İhtiyaçlar, beklentiler, gereksinimler, iş birliği isteği vb.
2. **Ekip Üyeleri**: Beceri, bilgi, deneyim seviyesi, elverişlilik, eğitim ihtiyaçları vb.
3. **Kurumun Faaliyet Alanı**: Test nesnesinin kritiklik düzeyi, tanımlanan riskler, pazar ihtiyaçları, yasal düzenlemeler vb.
4. **Teknik Faktörler**: Yazılım türü, ürün mimarisi, kullanılan teknoloji vb.
5. **Proje Kısıtları**: Kapsam, zaman, bütçe, kaynaklar vb.
6. **Organizasyonel Faktörler**: Organizasyonel yapı, mevcut politikalar, kullanılan pratikler vb.
7. **Yazılım Geliştirme Yaşam Döngüsü**: Mühendislik uygulamaları, geliştirme yöntemleri vb.
8. **Araçlar**: Elverişlilik, kullanılabilirlik, uyumluluk vb.

Bu faktörler, test stratejisi, kullanılan test teknikleri, test otomasyonu derecesi, gerekli kapsam seviyesi, test dokümantasyonu ayrıntı düzeyi, raporlama gibi testle ilgili konulara etki eder

1.4.3. Test Yazılımı

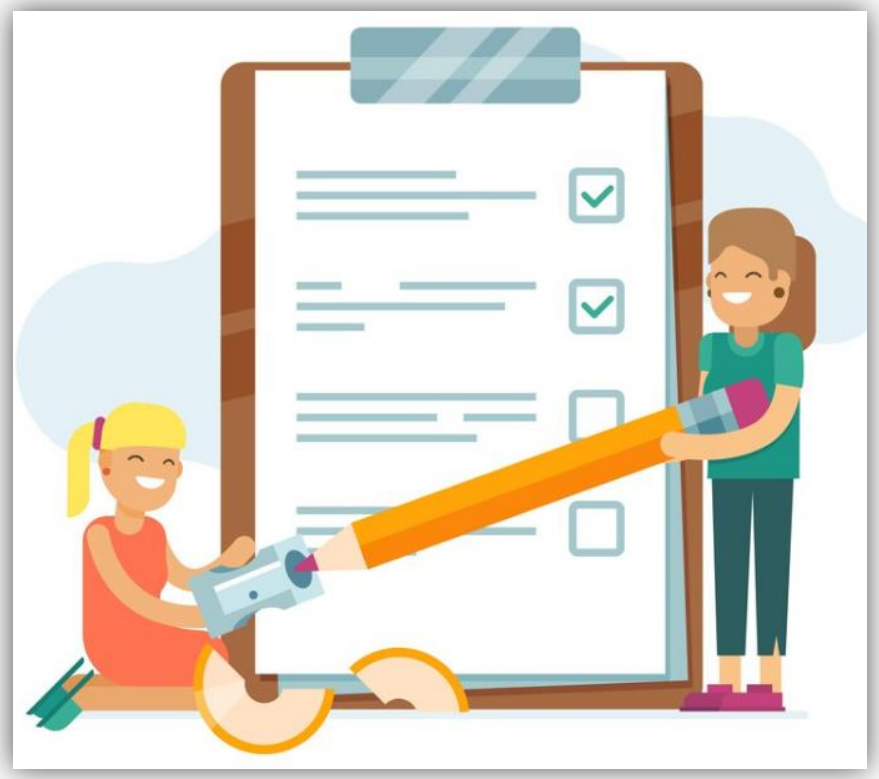
- Test yazılımı, belirli test aktivitelerinden elde edilen çalışma ürünlerinin bir parçası olarak oluşturulur.
- Farklı organizasyonlar, çalışma ürünlerini üretme, şekillendirme, adlandırma, düzenleme ve yönetme konusunda farklı yaklaşımlar kullanabilir.
- Uygun yapılandırma yönetimi, çalışma ürünleri arasında tutarlılık ve bütünlük sağlar.
- Çalışma ürünleri şunları içerebilir:
  1. **Test planlama**: Test planı, test zaman çizelgesi, risk kaydı, giriş ve çıkış kriterleri.
  2. **Test gözetimi ve kontrolü**: Test ilerleme raporları, kontrol direktifleri dokümantasyonu, risk bilgisi.
  3. **Test analizi**: Test koşulları, hata raporu.
  4. **Test tasarımı:** Test senaryoları, test başlatma belgeleri, kapsam öğeleri, test verisi gereksinimleri, test ortamı gereksinimleri.
  5. **Test uyarlama:** Test prosedürleri, otomatik test betikleri, test grupları, test verisi, test koşumu çizelgesi, test ortamı öğeleri.
  6. **Test koşumu:** Test kayıtları, hata raporları.
  7. **Test tamamlama:** Test tamamlama raporu, aksiyon öğeleri, tecrübeler, değişiklik talepleri.

1.4.4. Test Esası ve Test Yazılımı Arasında İzlenebilirlik

- İzlenebilirlik, test gözetimi ve kontrolünü etkili hale getirmek için önemlidir.
- Test esası unsurları, test yazılımı, test sonuçları ve tespit edilen hatalar arasındaki ilişkiyi sağlar.
- Kapsam değerlendirmesini destekler ve ölçülebilir kapsama kriterlerinin takibini kolaylaştırır.
- Örneğin, test senaryolarının gereksinimlere göre izlenebilirliği, testlerin gereksinimleri ne kadar karşıladığını doğrulamaya yardımcı olabilir.
- İyi bir izlenebilirlik, kapsam değerlendirmesi yanında etki analizi, test denetimleri ve BT yönetişimi gereksinimlerini karşılamada da yardımcı olur.
- Ayrıca, test ilerleme durumu ve tamamlanma raporlarının anlaşılmasını kolaylaştırır ve paydaşlara testin teknik yönlerini anlaşılır şekilde iletmeye yardımcı olur.
- İzlenebilirlik, iş hedeflerine göre ürün kalitesinin, süreç kapasitesinin ve proje ilerlemesinin değerlendirilmesi için önemli bilgiler sağlar.

1.4.5. Test Etme Sürecindeki Roller

1. **Test Ders Programında İki Temel Rol:**
   - Test Yönetimi Rolü
   - Test Etme Rolü
2. **Test Yönetimi Rolü:**
   - Test sürecini yönetmek ve denetlemekle sorumludur.
   - Özellikle test planlama, gözetimi ve kontrolü, ve test tamamlama aktivitelerine odaklanır.
   - Yerine getirme şekli, projenin bağlamına, ekibin becerilerine ve kuruma göre değişir.
3. **Test Etme Rolü:**
   - Testin teknik yönünden sorumludur.
   - Temel olarak test analizi, tasarımı, uyarlama ve koşumu aktivitelerine odaklanır.
4. **Sorumlulukların Değişimi:**
   - Rollerin sorumluluğunu farklı kişiler üstlenebilir.
   - Aynı kişi hem test etme hem de test yönetimi rollerini üstlenebilir.



1.5. Test Etme Sürecinde Gerekli Beceriler ve İyi Uygulamalar
1.5.1. Test Etme Sürecinde Gerekli Genel Beceriler

1. **Önemli Beceriler:**
   - Test bilgisi (test tekniklerini kullanarak etkinlik)
   - Titizlik, dikkat, merak, detaycılık (zor hataları bulmak için)
   - İyi iletişim, aktif dinleme, ekip çalışması (paydaşlarla etkili iletişim)
   - Analitik ve eleştirel düşünme, yaratıcılık (etkin test için)
   - Teknik bilgi ve alan bilgisi (test etmenin etkinliği için)
2. **İletişim Becerisinin Önemi:**
   - Test sonuçlarının ürüne yönelik eleştiri olarak algılanabileceği unutulmamalı.
   - İletişim becerisi, test sonuçlarının kabul edilmesini kolaylaştırır ve doğrulama yanlılığını önler.
   - Hatalar ve arızalar yapıcı bir şekilde iletilmeli, projeye ve ürüne katkı sağladığı vurgulanmalı.

1.5.2. Tüm Ekip Yaklaşımı

- **Ekip Çalışması Becerisi:**
  - Test uzmanları için temel becerilerden biri, ekip bağlamında etkin çalışma ve ekip hedeflerine olumlu katkı sağlama yeteneğidir.
  - Ekstrem Programlama metodolojisinden alınan tüm ekip yaklaşımı, bu beceriyi vurgular.
- **Tüm Ekip Yaklaşımı:**
  - Bu yaklaşımda, tüm ekip üyeleri gerekli bilgi ve beceriye sahiptir ve kaliteden herkes sorumludur.
  - Ortak çalışma alanı iletişimi kolaylaştırır, ekip içi iletişimi artırır ve farklı beceri setlerinin sinerji oluşturmasını sağlar.

- **Test Uzmanlarının Rolü:**
  - Test uzmanları, diğer ekip üyeleriyle yakın iş birliği içinde çalışarak istenen kalite seviyelerine ulaşmayı hedefler.
  - İş birimleriyle kabul testleri oluşturmak, yazılımcılarla test stratejisi ve otomasyon metodolojilerini belirlemek gibi görevlerde bulunurlar.
  - Test uzmanları, bilgilerini diğer ekip üyelerine aktararak ve ürün gelişimine katkı sağlayarak önemli bir rol oynarlar.
- **Bağlama Göre Uyarlanabilirlik:**
  - Tüm ekip yaklaşımı, her zaman uygun olmayabilir; özellikle bazı emniyet hassasiyetli durumlarda, yüksek düzeyde test bağımsızlığı gerekebilir.

1.5.3. Testin Bağımsızlığı

- Bağımsızlık, test uzmanlarının hataları bulmada etkin olmalarını sağlar, ancak aşina olmanın yerine geçemez.
- Çalışma ürünleri, farklı bağımsızlık seviyelerine sahip kişiler tarafından test edilebilir.
- Bağımsız test uzmanları, farklı deneyimlerinden ve bakış açılarından dolayı çeşitli hataları daha iyi bulabilirler.
- Ancak, farklı davranışları ve iletişim problemleri gibi sakıncalar da olabilir.

2. Yazılım Geliştirme Yaşam Döngüsü Boyunca Test
2.1. Yazılım Geliştirme Yaşam Döngüsü Bağlamında Test

- Yazılım geliştirme yaşam döngüsü (YGYD), yazılım geliştirme sürecinin genel bir tasviri olarak kabul edilir.
- YGYD modelleri, farklı geliştirme aşamaları ve faaliyet türlerinin mantıksal ve kronolojik ilişkilerini gösterir.
- Örnek YGYD modelleri şunlardır: şelale modeli, V modeli, spiral model, prototipleme, Birleşik Süreç.
- Yazılım geliştirme sürecindeki aktiviteler, daha detaylı yazılım geliştirme yöntemleri ve Çevik uygulamalarla tanımlanabilir.
- Örnekler: kabul testi güdümlü yazılım geliştirme (ATDD), davranış güdümlü yazılım geliştirme (BDD), alan güdümlü tasarım (DDD), ekstrem programlama (XP), özellik güdümlü geliştirme (FDD), Kanban, Yalın BT, Scrum ve test güdümlü geliştirme (TDD).

2.1.1. Yazılım Geliştirme Yaşam Döngüsünün Test Üzerindeki Etkisi

- Testlerin başarılı olması için YGYD'ne uyarlanması gerekir. YGYD seçimi şunları etkiler:
  - Test aktivitelerinin kapsamı ve zamanlaması
  - Test dokümantasyonunun ayrıntı düzeyi
  - Test tekniklerinin ve yaklaşımının seçimi
  - Test otomasyonunun kapsamı
  - Test uzmanının rol ve sorumlulukları

- Sıralı yazılım geliştirme modellerinde, test uzmanları genellikle ilk aşamalarda gereksinim gözden geçirmelerine, test analizine ve test tasarımına katılır. Dinamik testler genellikle sonraki aşamalarda gerçekleştirilir.
- Bazı döngüsel ve artımlı geliştirme modellerinde, her döngü bir çalışma prototipi veya ürün özelliği sağlar. Bu durum her döngüde statik ve dinamik testlerin gerçekleştirilebileceği anlamına gelir.
- Çevik yazılım geliştirmede, iş ürünü belgelerinin hafifletilmesi ve kapsamlı test otomasyonunun kullanılması tercih edilir. Manuel testler, tecrübeye dayalı test teknikleri kullanılarak yapılır ve önceden kapsamlı test analizi ve tasarım gerektirmez

2.1.2. Yazılım Geliştirme Yaşam Döngüsü ve İyi Test Etme Uygulamaları

- Her yazılım geliştirme aktivitesine bir test aktivitesi eşlik eder, böylece tüm geliştirme süreci kalite kontrole tabi tutulur.
- Farklı test seviyeleri belirli ve farklı test hedeflerine sahiptir, bu da kapsamlı test yapılırken gereksiz tekrarlardan kaçınılır.
- Test analizi ve tasarımı belirli bir test seviyesi için YGYD'nin ilgili geliştirme aşamasında başlar, böylece erken test prensibine uygun şekilde test gerçekleştirilir.
- Test uzmanları, dokümantasyonun taslakları hazır olur olmaz çalışma ürünlerini gözden geçirme sürecine dahil olur, böylece erken test ve hata tespiti "shift-left" stratejisini destekler.

2.1.3. Yazılım Geliştirme Faktoru Olarak Test

- **TDD (Test Güdümlü Yazılım Geliştirme):**
  - Kodlamayı test senaryoları yönlendirir, kod önce değil testler önce yazılır.
  - Testler yazılır, ardından kod yazılır ve son olarak testler ve kod düzenlenir.
- **ATDD (Kabul Testi Güdümlü Yazılım Geliştirme):**
  - Kabul kriterlerinden testler türetilir, sistem tasarımının bir parçası olarak yapılır.
  - Testler önce yazılır, sonra kodlama yapılır.
- **BDD (Davranış Güdümlü Yazılım Geliştirme):**
  - Uygulamanın istenen davranışı, basit bir doğal dilde test senaryoları ile ifade edilir.
  - Test senaryoları otomatik olarak yürütülebilir testlere dönüştürülür.

  Tüm bu yaklaşımlarda, testler gelecekteki uyarlama ve yeniden düzenleme süreçlerinde kod kalitesini sağlamak için otomatikleştirilmiş testler olarak kalır.

2.1.4. DevOps ve Test Etme

- DevOps, yazılım geliştirme ve operasyon ekiplerinin birlikte çalışmasını hedefler.
- Organizasyonel bir değişim gerektirir ve ekip özerkliği, hızlı geri bildirim gibi kültürel unsurları destekler.
- Teknik uygulamalar arasında sürekli entegrasyon (CI) ve sürekli teslimat (CD) bulunur.

Test perspektifinden faydalar:

- Hızlı geri bildirim ve kod kalitesi üzerinde olumlu etki.
- Shift-left yaklaşımının teşviki ve stabil test ortamlarının oluşturulması.
- Otomatik süreçlerin desteklenmesi ve fonksiyonel olmayan gereksinimlerin artan görüş alanı.
- Otomatik regresyon testleriyle regresyon riskinin azaltılması.

Zorluklar:

- DevOps teslimat hattının tanımlanması ve kurulması gerekliliği.
- CI / CD araçlarının tanıtılması ve bakımının yapılması.
- Test otomasyonunun ek kaynaklar gerektirmesi ve kurulumunun zor olması.

Yüksek düzeyde test otomasyonuna rağmen, kullanıcı perspektifinden bakıldığında manuel testlere hala ihtiyaç duyulabilir

2.1.5. Shift-Left Yaklaşımı

- **"Erken test" prensibi**, YGYD'nün başlarında gerçekleştirilen testleri vurgular ve bazen **"shift-left"** olarak adlandırılır.
- Shift-left, testlerin daha erken yapılmasını teşvik eder, ancak ilerleyen aşamalarda testlerin ihmal edilmesi anlamına gelmez.

Shift-left'in sağlanması için birtakım iyi uygulamalar vardır:

1. Analizin test bakış açısıyla gözden geçirilmesi.
2. Kod yazılmadan önce test senaryolarının yazılması ve kodun bir test kuluçkasında çalıştırılması.
3. CI ve hatta CD'nin kullanılması.
4. Kaynak kodun statik analizinin tamamlanması.
5. Bileşen testi seviyesinden başlayarak fonksiyonel olmayan testlerin gerçekleştirilmesi.

- Shift-left, süreç başında ek eğitim, efor ve maliyete neden olabilir, ancak sonraki aşamalarda tasarruf sağlayabilir.
- Paydaşların shift-left yaklaşımına ikna edilmesi ve bu konsepte inanmaları önemlidir

2.1.6. Geçmişe Dönük Öğeler ve Süreç İyileştirmesi

- **Geçmişe Dönük Öğeler:**
  - Proje veya döngü sonunda, sürüm kilometre taşında veya gerektiğinde ele alınır.
  - Zamanlaması ve organizasyonu, izlenen YGYD modeline bağlıdır.
- **Toplantı İçeriği:**
  - Başarılar ve başarısızlıklar tartışılır.
  - Nasıl sürdürülebilir ve iyileştirme yapılabilir soruları ele alınır.
- **Sonuçların Kaydedilmesi:**
  - Genellikle test tamamlama raporunun bir parçasıdır.
- **Faydaları:**
  - Daha yüksek test etkinliği ve verimliliği.
  - Daha yüksek test yazılımı kalitesi.
  - Ekip kaynaşması ve öğrenmesi.
  - Daha yüksek test esası kalitesi.
  - Geliştirme ve test etme süreci arasında daha iyi iş birliği.

2.2. Test Seviyeleri ve Test Çeşitleri

- **Test Seviyeleri:**
  - Yazılım geliştirme aşamalarındaki test aktiviteleri gruplarıdır.
  - Bağımsız bileşenlerden komple sistemlere kadar gerçekleştirilen test süreçlerinin örnekleridir.
- **Bağlantıları:**
  - YGYD içindeki diğer aktivitelerle ilişkilidir.
  - Sıralı YGYD modellerinde, çıkış ve giriş kriterleri tanımlanır.
- **Gelişim Aktiviteleri:**
  - Birden fazla test seviyesini kapsayabilir.
  - Zaman içinde çakışabilir.
- **Test Çeşitleri:**
  - Gereksinimlerle ilgili test aktiviteleri gruplarıdır.
  - Her test seviyesinde yapılabilirler.

2.2.1. Test Seviyeleri

- **Bileşen Testi (Birim Testi):**
  - Bileşenleri ayrı olarak test etmeye odaklanır.
  - Genellikle geliştiriciler tarafından kendi ortamlarında yapılır.
- **Bileşen Entegrasyon Testi (Birim Entegrasyon Testi):**
  - Arayüzlerin ve bileşenler arasındaki etkileşimleri test eder.
  - Entegrasyon stratejisine bağlı olarak aşağıdan yukarıya, yukarıdan aşağıya veya big-bang gibi yaklaşımlar kullanılabilir.
- **Sistem Testi:**
  - Sistemin genel davranışına ve yeteneklerine odaklanır.
  - Uçtan uca görevlerin fonksiyonel ve fonksiyonel olmayan testlerini içerir.
- **Sistem Entegrasyon Testi:**
  - Test edilen sistemin diğer sistemler ve harici hizmetlerle arayüzlerini test eder.
  - Operasyonel ortama benzer uygun test ortamları gerektirir.
- **Kabul Testi:**
  - Sistemin kullanıcıya sunulmaya hazır olduğunu göstermeye odaklanır.
  - Kullanıcılar tarafından gerçekleştirilir.
- **Test Seviyelerini Ayırmak İçin Kullanılan Liste:**
  - Test nesnesi
  - Test hedefleri
  - Test esası
  - Hatalar ve arızalar
  - Yaklaşım ve sorumluluklar

2.2.2. Test Çeşitleri

- **Fonksiyonel Testler:**
  - Bileşen veya sistemin yerine getirmesi gereken fonksiyonları değerlendirir.
  - Ana hedef, fonksiyonel bütünlüğü, doğruluğu ve uygunluğu kontrol etmektir.
- **Fonksiyonel Olmayan Testler:**
  - Fonksiyonel gereksinimler dışındaki özellikleri değerlendirir.
  - Ana hedef, performans, uyumluluk, kullanılabilirlik, güvenilirlik, güvenlik, sürdürülebilirlik, taşınabilirlik gibi fonksiyonel olmayan gereksinimleri kontrol etmektir.
- **Kara Kutu Testi:**
  - Gereksinim bazlıdır ve dışarıdan bakarak testleri elde eder.
  - Ana hedef, sistemin gereksinimlere karşı davranışını kontrol etmektir.
- **Beyaz Kutu Testi:**
  - Yapı bazlıdır ve uygulama veya iç yapıdan testleri elde eder.
  - Ana hedef, altta yatan yapıyı kabul edilebilir seviyeye kadar kapsamasını sağlamaktır.
- Her bir test çeşidi farklı odak noktalarına sahip olsa da, tüm test seviyelerinde uygulanabilir.
- Her bir test çeşidi için çeşitli test teknikleri uygulanabilir.

2.2.3. Onaylama Testleri ve Regresyon Testleri

- **Onaylama Testi:**
  - Asıl hatanın başarıyla giderildiğini onaylamak için yapılır.
  - Yazılımın düzeltilmiş versiyonu çeşitli şekillerde test edilir: önceki hatalı senaryoların çalıştırılması veya yeni testlerin eklenmesi.
  - Zaman veya maliyet kısıtları nedeniyle sınırlı olabilir, hata tekrarını kontrol etmekle sınırlı olabilir.
- **Regresyon Testi:**
  - Bir değişikliğin herhangi bir olumsuz etkiye neden olmadığını teyit etmek için yapılır.
  - Değişiklik, test nesnesini, diğer bileşenleri ve hatta diğer bağlı sistemleri etkileyebilir.
  - Test otomasyonu için güçlü bir adaydır ve projenin erken aşamalarında başlaması tavsiye edilir.
- **Test Seviyelerinde Onaylama ve Regresyon Testi:**
  - Hataların düzeltilmesi veya değişiklikler yapıldığında tüm test seviyelerinde yapılmalıdır.
  - Tüm test seviyelerinde onaylama ve/veya regresyon testi gereklidir.

2.3. Bakım Testleri

- **Bakım Kategorileri:**
  - Bakım, düzeltici, uyarlanabilir veya performans/sürdürülebilirlik artırıcı olabilir.
  - Planlı sürümleri/dağıtımları ve planlanmamış sürümleri/dağıtımları içerebilir.
- **Etki Analizi:**
  - Değişiklik yapmadan önce, olası sonuçları değerlendirmek için etki analizi yapılabilir.
  - Canlı ortamdaki sistemde yapılan değişikliklerin test edilmesi ve regresyonların kontrol edilmesi gereklidir.
- **Bakım Testi Kapsamı:**
  - Değişikliğin risk derecesi, mevcut sistemin boyutu ve değişikliğin boyutu bakım testinin kapsamını belirler.
- **Bakım Tetikleyicileri:**
  - Değişiklikler, operasyonel ortam yükseltmeleri/taşımaları ve kullanımdan kaldırma gibi durumlar bakım ve bakım testlerini tetikleyebilir.
  - Operasyonel ortam yükseltmeleri, taşımaları ve kullanımdan kaldırma durumları, özel test gereksinimleri ortaya çıkarabilir.

3.Statik Testler

3.1. Statik Testin Temelleri

- **Statik Test:**
  - Test edilen yazılımın çalıştırılmasına gerek yoktur.
  - Kod, süreç, sistem mimarisi veya diğer çalışma ürünleri, manuel inceleme veya araçlarla değerlendirilir.
  - Test hedefleri arasında kaliteyi artırmak, hataları tespit etmek ve özellikleri değerlendirmek bulunur.
  - Doğrulama ve sağlama için kullanılabilir.
- **İş Birliği ve Gözden Geçirme:**
  - Test uzmanları, iş birimleri ve geliştiriciler, kullanıcı hikayeleri ve ilgili çalışma ürünlerinin kriterlere uygunluğunu sağlamak için birlikte çalışır.
  - Kullanıcı hikayelerinin tam ve anlaşılır olmasını ve test edilebilir kabul kriterlerini içermesini sağlamak için gözden geçirme teknikleri kullanılır.
- **Statik Analiz:**
  - Problemleri dinamik testten önce belirleyebilir.
  - Genellikle daha az efor gerektirir ve CI sürecine dahil edilir.
  - Sürdürülebilirlik ve güvenlik de değerlendirilebilir.
  - Yazım denetleyici ve okunabilirlik araçları gibi araçlar kullanılabilir.

3.1.1. Statik Testlerle İncelenebilen Çalışma Ürünleri

- **Statik Test İncelenebilecek Örnekler:**
  - Gereksinimler
  - Kaynak kod
  - Test planları
  - Test senaryoları
  - Ürün iş listesi öğeleri
  - Test başlatma belgeleri
  - Proje dokümantasyonu
  - Sözleşmeler
  - Modeller
- **Gözden Geçirme için Uygun Çalışma Ürünleri:**
  - Okunup anlaşılabilir olanlar
  - Kontrol mekanizması niteliğinde yapıya sahip olanlar
- **Uygun Olmayan Çalışma Ürünleri:**
  - İnsanlar tarafından zor yorumlanabilenler
  - Araçlarla analiz edilemeyenler

3.1.2. Statik Testin Önemi

- **Statik Testin Erken Aşamalardaki Rolü:**
  - Erken hataları tespit ederek erken test prensibini yerine getirebilir.
  - Dinamik testin tespit edemeyeceği hataları bulabilir.
- **Çalışma Ürünlerinin Kalitesini Artırma:**
  - Kaliteyi değerlendirir ve güven duygusunu artırır.
  - Belgelenmiş gereksinimlerin doğruluğunu sağlar.
- **İletişimi Güçlendirme:**
  - İlgili paydaşlar arasında ortak bir anlayış oluşturur.
  - İletişimi güçlendirir.
- **Gözden Geçirmelerin Maliyet ve Faydası:**
  - Gözden geçirmelerin maliyeti olabilir, ancak geri dönüş oranı yüksektir.
  - Projenin ilerleyen aşamalarında daha fazla zaman ve efor tasarrufu sağlar.
- **Statik Analiz ile Kod Hatalarının Tespiti:**
  - Kod hataları dinamik test yerine statik analiz kullanılarak daha etkin bir şekilde tespit edilebilir.
  - Bu genellikle daha az kod hatası ve daha düşük toplam geliştirme eforuyla sonuçlanır.

3.1.3. Statik Test ve Dinamik Test Arasındaki Farklar

- **Statik ve Dinamik Testler Arasındaki Farklar:**
  - Her iki test türü de hataları tespit eder, ancak bazı hata türleri sadece biriyle bulunabilir.
  - Dinamik testler, hataları belirlerken, statik testler hataları doğrudan bulur.
  - Statik testler, çalıştırılamayan kod kısımlarındaki hataları daha kolay tespit edebilir.
  - Statik testler, kodun çalıştırılmasına bağlı olmayan kalite ölçümünde kullanılabilirken, dinamik testler kodun çalıştırılmasına bağlıdır.
- **Statik Testler ile Bulunan Yaygın Hatalar:**
  - Gereksinimlerdeki hatalar (tutarsızlıklar, belirsizlikler, çelişkiler)
  - Tasarım hataları (verimsiz yapılar, zayıf modülerleştirme)
  - Bazı kod hataları (tanımsız değişkenler, ulaşılamaz kod blokları)
  - Standartlardan sapmalar (adlandırma kurallarına uymama)
  - Arayüz gereksinimlerindeki hatalar (parametre uyumsuzlukları)
  - Güvenlik açıkları (arabellek aşımları)
  - Test kapsamındaki eksiklikler veya hatalar

3.2. Geri Bildirim ve Gözden Geçirme Süreci
3.2.1. Erken ve Sık Paydaş Geri Bildiriminin Faydaları

- Erken ve sık geri bildirimler, potansiyel kalite problemlerinin erken tespitini sağlar.
- Az paydaş katılımı, ürünün asıl vizyonu karşılamamasına ve projenin başarısız olmasına neden olabilir.
- Sık paydaş geri bildirimleri, gereksinimlerdeki yanlış anlaşılmaları önler ve değişikliklerin erken uygulanmasını sağlar.
- Geliştirme ekibine ne yaptıklarını daha iyi anlama fırsatı verir.
- Paydaşlara en değerli ve riskleri en iyi yönetecek özelliklere odaklanmalarını sağlar.

3.2.2. Gözden Geçirme Süreci Faaliyetleri

- **Planlama:** Gözden geçirme kapsamı, kalite karakteristikleri, odaklanılacak alanlar ve çıkış kriterleri gibi bilgileri içerir.
- **Gözden geçirme başlangıcı**: Herkesin hazır olması sağlanır, erişim ve roller belirlenir.
- **Bireysel gözden geçirme**: Gözden geçiriciler, çalışma ürününü değerlendirir ve anomalileri belirler.
- **İletişim ve analiz:** Anomalilerin analizi yapılır, kararlar alınır ve takip aksiyonları belirlenir.
- **Düzeltme ve raporlama**: Hata raporları oluşturulur, düzeltici aksiyonlar izlenir ve sonuçlar raporlanır.

3.2.3. Gözden Geçirmede Roller ve Sorumluluklar

- **Yönetici**: Gözden geçirme kaynaklarını sağlar ve neyin gözden geçirileceğine karar verir.
- **Yazar**: Çalışma ürünlerini oluşturur ve düzeltir.
- **Moderatör**: Gözden geçirme toplantılarını yönetir, güvenli bir ortam sağlar ve etkili iletişimi kolaylaştırır.
- **Katip**: Anomalileri kaydeder ve gözden geçirme toplantılarında alınan kararları belgeleme görevini üstlenir.
- **Gözden geçirici**: Çalışma ürünlerini inceleyen ve değerlendiren kişi veya ekip.
- **Gözden geçirme lideri**: Gözden geçirmenin genel sorumluluğunu üstlenir ve organizasyonu yönetir.

3.2.4. Gözden Geçirme Çeşitleri

- **Gayri resmi gözden geçirme**: Süreç tanımlı değil, resmi çıktı gerektirmez, anomalileri tespit eder.
- **Üzerinden geçme**: Yazar liderliğinde, kaliteyi artırmak ve güven oluşturmak için yapılır.
- **Teknik Gözden Geçirme**: Nitelikli kişiler tarafından yönetilir, teknik problemleri ele alır, kaliteyi değerlendirir.
- **Teftiş**: En resmi, eksiksiz izlenen bir süreç, maksimum sayıda anomaliyi bulmayı hedefler.

3.2.5. Gözden Geçirmelerin Başarı Faktörleri

- Net hedefler ve ölçülebilir çıkış kriterlerinin belirlenmesi önemlidir.
- Uygun gözden geçirme türünün seçilmesi, çalışma ürününün türüne, katılımcılara ve proje ihtiyaçlarına uygun olmalıdır.
- Gözden geçirme sürecinin parçalara bölünmesi, katılımcıların dikkatini dağıtmamak için önemlidir.
- Paydaşlara ve yazarlara geri bildirim verilerek ürün ve aktivitelerin iyileştirilmesi sağlanmalıdır.
- Katılımcıların hazırlanması için yeterli süre verilmelidir.
- Yönetimden destek alınması, gözden geçirme sürecini güçlendirir.
- Şirket kültürünün bir parçası olarak gözden geçirmelerin benimsenmesi ve sürekli iyileştirilmesi önemlidir.
- Tüm katılımcılara rollerini yerine getirebilmeleri için eğitim verilmelidir.
- Toplantıların kolaylaştırılması, verimli bir gözden geçirme süreci için gereklidir.

4. Test Analizi ve Tasarımı
4.1. Test Tekniklerine Genel Bakış

- **Kara kutu test teknikleri**: test nesnesinin davranışını analiz eder ve yazılımın nasıl yazıldığından bağımsızdır.
- **Beyaz kutu test teknikleri:** test nesnesinin iç yapısını ve işlemlerini analiz eder ve yazılımın nasıl tasarlandığına bağlıdır.
- **Tecrübeye dayalı test teknikleri:** test senaryolarının tasarımı ve uyarlanması için test uzmanlarının bilgi ve deneyimlerini kullanır.
- Tecrübeye dayalı test teknikleri, kara kutu ve beyaz kutu test tekniklerini tamamlayıcı niteliktedir.

4.2. Kara Kutu Test Teknikleri

Yaygın olarak kullanılan kara kutu test teknikleri aşağıdaki bolumlerde ele alınmaktadır:

- Denklik Paylarına Ayırma
- Sınır Değer Analizi
- Karar Tablosu Testleri
- Durum Gecişi Testleri

4.2.1. Denklik Paylarına Ayırma

- Denklik Paylarına Ayırma (DPA), bir test nesnesinin tüm unsurlarının aynı şekilde işleneceği beklentisine dayanarak verileri paylara ayırır.
- DPA, her pay için bir test yapılmasını önerir ve paylar sürekli veya ayrık, sıralı veya sırasız, sonlu veya sonsuz olabilir.
- Paylar birbiriyle çakışmamalı ve boş küme halinde olmamalıdır.
- DPA'nın kullanımı, test nesnesinin davranışını anlamak için titizlik gerektirir.
- Geçerli değerler içeren paylara "geçerli pay" denirken, geçersiz değerler içerenlere "geçersiz pay" denir.
- DPA'da kapsam öğeleri denklik paylarıdır ve her bir payı en az bir kez deneyerek kapsamın %100'üne ulaşılması hedeflenir.
- Birden fazla pay grubu içeren test nesnelerinde, Each Choice kapsamı en basit kapsam kriteridir ve her pay grubundan her bir payın en az bir kez denemesini gerektirir.

4.2.2. Sınır Değer Analizi

- **Sınır Değer Analizi (SDA)**, denklik paylarının sınırlarının denenmesine dayalı bir tekniktir ve sadece sıralı veriler için kullanılabilir.
- SDA, payların minimum ve maksimum değerlerini test ederek geliştiricilerin sınır değerlerinde yapabileceği hataları belirlemeyi amaçlar.
- İki versiyonu bulunur: 2 değerli SDA ve 3 değerli SDA.
- **2 değerli SDA**: Her sınır değeri için iki kapsam öğesi vardır: sınır değer ve bitişik paya ait en yakın komşusu.
- **3 değerli SDA**: Her sınır değeri için üç kapsam öğesi vardır: sınır değer ve bu sınır değerin her iki komşu sınır değeri.

Örneğin;

- 3 değerli SDA, 2 değerli SDA'dan daha titiz bir analiz yapar ve daha fazla hata tespit edebilir.
- **2 değerli SDA:** Sadece x = 10 ve x = 11 gibi sınır değerleri test edileceğinden, hata tespit edilemez çünkü sınır değeri doğru olarak tanımlanmıştır.
- **3 değerli SDA:** Bu durumda x = 9 de dahil olmak üzere sınırın her iki tarafı da test edileceği için hata tespit edilebilir.

4.2.3. Karar Tablosu Testleri

- **Karar Tablolarının Kullanımı:**
  - Farklı test koşul kombinasyonlarının çıktılarını göstermek ve karmaşık mantığı etkin bir şekilde göstermek için kullanılır.
  - Koşulların yerine getirilip getirilmediği ve buna bağlı olarak alınacak aksiyonlar tanımlanır.
  - Koşullar tablonun satırlarını oluştururken, her sütun benzersiz bir koşul kombinasyonunu ve ilişkili aksiyonları belirtir.
- **Karar Tablolarının Yapısı:**
  - Sınırlı girişli karar tablolarında, koşul ve aksiyonlar genellikle Boolean (doğru veya yanlış) değerler olarak gösterilir.
  - Genişletilmiş girişli karar tablolarında ise koşullar ve aksiyonlar birden fazla değer alabilir.
- **Koşul ve Aksiyon Gösterimi:**
  - Koşul için: "T" (doğru), "F" (yanlış), "–" (ilgisiz), "N/A" (uygulanabilir değil).
  - Aksiyon için: "X" (gerçekleşmesi gereken), boşluk (gerçekleşmemesi gereken).
- **Kapsam ve Test Senaryoları:**
  - Kapsam, test senaryolarının tüm uygulanabilir sütunları deneyerek %100 kapsama ulaşmasını gerektirir.
  - Kapsam, denenen sütun sayısının tüm uygulanabilir sütunların sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.
- **Avantajları:**
  - Tüm koşul kombinasyonlarını ele alarak sistemli bir yaklaşım sağlar.
  - Gereksinimlerdeki boşlukları veya çelişkileri bulmaya yardımcı olur.
- **Zorluklar ve Çözümler:**
  - Koşul sayısı arttıkça uygulanacak kural sayısı da artar. Bu durumda, indirgenmiş bir karar tablosu veya risk temelli bir yaklaşım kullanılabilir.

4.2.4. Durum Geçişi Testleri

- **Durum Geçişi Diyagramları (DGD) ve Durum Tabloları:**
  - DGD, bir sistemin olası durumlarını ve bu durumlar arasındaki geçişleri gösterir.
  - Durum tabloları, DGD'nin eşdeğeri olup durumlar, olaylar ve koruma koşullarını içerir.
- **Durum Geçişi Testi ve Kapsama Kriterleri:**
  - **Tüm Durumlar Kapsamı:**
    - Tüm durumları kapsamak için test senaryoları tüm durumları incelemelidir.
    - Kapsam, incelenen durum sayısının toplam durum sayısına bölünmesiyle ölçülür.
  - **Geçerli Geçişler Kapsamı:**
    - Geçerli geçişlerin tamamını test etmek için test senaryoları tüm geçerli geçişleri denemelidir.
    - Kapsam, denenen geçerli geçişlerin sayısının toplam geçerli geçiş sayısına bölünmesiyle ölçülür.
  - **Tüm Geçişler Kapsamı:**
    - Tüm geçişlerin, hem geçerli hem de geçersiz olanların, tamamını kapsamak için test senaryoları tüm geçişleri denemelidir.
    - Bu kapsam, denenen geçişlerin toplam geçiş sayısına bölünmesiyle ölçülür.
- **Kapsama ve Test Senaryoları:**
  - Bir test senaryosu, bir dizi durum değişikliğiyle temsil edilir ve durumlar arasındaki geçişleri içerir.
  - Kapsama, yürütülen test senaryoları tarafından denenen veya kapsanmaya çalışılan geçerli ve geçersiz geçiş sayısının toplam geçerli ve geçersiz geçiş sayısına bölünmesiyle ölçülür.

4.3. Beyaz Kutu Test Teknikleri
4.3.1. Komut Testleri ve Komut Kapsama Yüzdesi

- **Komut Testi:**
  - Kapsam Öğeleri: Çalıştırılabilir komutlar.
  - Amaç: Kod içindeki tüm çalıştırılabilir komutları en az bir kez deneyerek kabul edilebilir bir kapsama seviyesi elde etmektir.
  - Kapsam Ölçümü: Test senaryoları tarafından denenen komut sayısının toplam çalıştırılabilir komut sayısına bölünmesiyle yapılır, yüzde olarak ifade edilir.
  - %100 Komut Kapsama:
    - Tüm çalıştırılabilir komutların en az bir kez denenmesini sağlar.
    - Her komutun hata içerip içermediğini gösterir, ancak tüm hataları tespit etmeyebilir.
    - Tüm karar mantığının test edilmesini garanti etmez, çünkü tüm kod dalları denenmeyebilir.

4.3.2. Dal Testi ve Dal Kapsamı

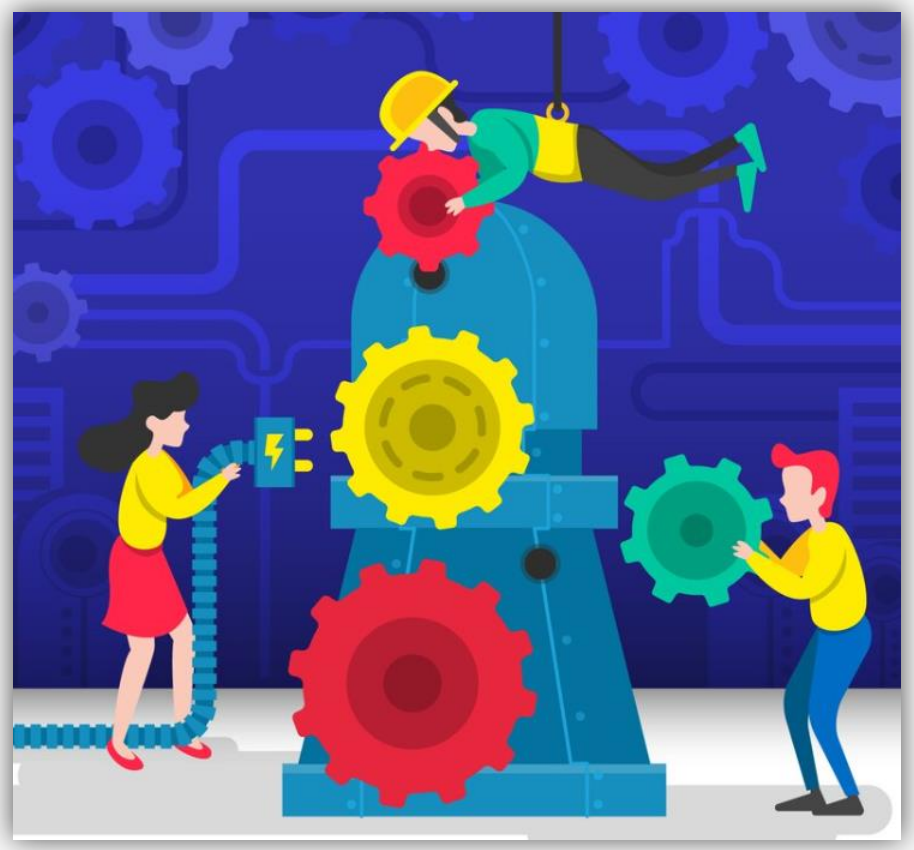
- **Dal Testi:**
  - Kapsam Öğeleri: Dallar, kontrol akış grafiğindeki iki düğüm arasındaki kontrol transferleridir.
  - Amaç: Kod içindeki tüm dalları en az bir kez deneyerek kabul edilebilir bir kapsama seviyesi elde etmektir.
  - Kapsam Ölçümü: Test senaryoları tarafından denenen dal sayısının toplam dal sayısına bölünmesiyle yapılır, yüzde olarak ifade edilir.
  - %100 Dal Kapsama:
    - Tüm koşulsuz ve koşullu dalların test senaryoları tarafından denenmesini sağlar.
    - Koşullu dallar genellikle "if...then" kararlarına, switch/case komutlarına veya döngü kararlarına karşılık gelir.
    - Bir dalın test senaryosuyla denenmesi, o daldaki tüm hataların tespit edileceği anlamına gelmez.
  - İlişki: Dal kapsamı, komut kapsama yüzdesini içerir; yani %100 dal kapsamına ulaşan test senaryoları grubu aynı zamanda %100'lük bir komut kapsama yüzdesine de ulaşır.

4.3.3. Beyaz Kutu Testinin Önemi

- **Beyaz Kutu Test Tekniklerinin Kuvvetli Yanları:**
  - Test sırasında kodun tamamının dikkate alınması, analiz belirsizliklerini veya eksikliklerini göz ardı etmeden hata tespitini kolaylaştırır.
  - Statik testlerde de kullanılabilir.
  - Kullanılabileceği Alanlar:
    - Henüz koşum için hazır olmayan kodlar.
    - Karmaşık veya hatalı kodlar.
    - Kontrol akış grafiği ile modellenen yazılımlar.
- **Zayıf Yönler:**
  - Kodlanmayan gereksinimleri tespit etme yeteneği yoktur.
- **Ölçüm ve Güven:**
  - Sadece kara kutu testi gerçekleştirmek, gerçek kod kapsamının ölçülmesini sağlamaz.
  - Beyaz kutu test teknikleri, objektif bir kapsam ölçümü sağlar ve koda yönelik güveni artırır.

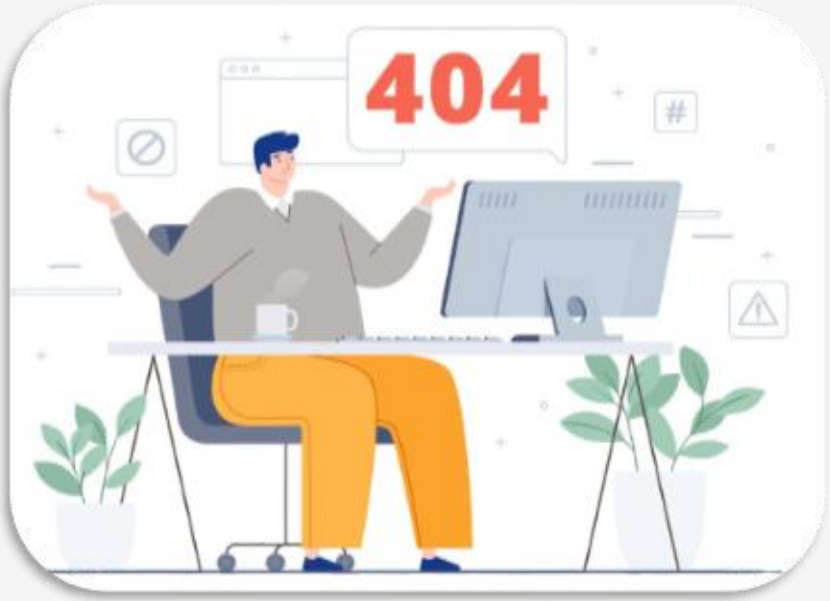
4.4. Tecrübeye Dayalı Test Teknikleri

- Yaygın olarak kullanılan tecrübeye dayalı test teknikleri aşağıdaki bölümlerde ele alınmaktadır:
- Hata tahminleme
- Keşif testi
- Kontrol listesine dayalı test



4.4.1. Hata Tahminleme

- **Hata Tahminleme:**
  - Test uzmanının bilgilerine dayalı olarak insan hatalarının, yazılım hatalarının ve arızaların ortaya çıkmasını sağlamak için kullanılan bir tekniktir.
  - Bilgi Kaynakları:
    - Uygulamanın geçmiş performansı ve çalışma şekli.
    - Geliştiricilerin sıkça yaptığı hatalar ve bunların sonuçları.
    - Benzer uygulamalardaki arıza çeşitleri.
  - Hataların ve Arızaların Çeşitleri:
    - Girdi, çıktı, mantık, hesaplama, arayüz veya veri ile ilgili olabilir.



- **Kusur Ortaya Çıkarmaya Yönelik Saldırılar:**
  - Hata tahminlemenin uygulanması için metodik bir yaklaşımdır.
  - Test uzmanı, olası hataları ve arızaları belirleyerek bunları ortaya çıkaracak testler tasarlar.
  - Listeler tecrübeye, hata ve arıza verilerine veya genel yazılım bilgilerine dayanarak oluşturulabilir.

4.4.2. Keşif Testi

- **Keşif Testi:**
  - Test uzmanı, test nesnesi hakkında bilgi edinirken testler eş zamanlı olarak tasarlanır, koşulur ve değerlendirilir.
  - Amaçları:
    - Test nesnesi hakkında daha fazla bilgi edinmek.
    - Odak testlerle test nesnesini derinlemesine keşfetmek.
    - Test edilmemiş alanlara yönelik testler oluşturmak.
  - Oturum Bazlı Yaklaşım:
    - Belirli bir zaman dilimi içinde gerçekleştirilir.
    - Test uzmanı, test başlatma belgesi kullanarak test hedeflerini belirler.
    - Test oturumunun ardından paydaşlarla bilgilendirme toplantısı yapılır.
  - Test Hedefleri:
    - Üst seviye test koşulları olarak görülür.
    - Kapsam öğeleri test oturumunda tanımlanır ve denenir.
  - Keşif Testlerinin Yararları:
    - Gereksinimler az veya yetersiz olduğunda veya zaman baskısı olduğunda işe yarar.
    - Diğer resmi test tekniklerini tamamlamak için kullanılabilir.
  - Test Uzmanı Nitelikleri:
    - Deneyimli, alan bilgisine sahip, analitik becerilere sahip, meraklı ve yaratıcı olmalıdır.
  - Diğer Test Teknikleri ile Birlikte Kullanım:
    - Denklik paylarına ayırma gibi diğer test teknikleri de içerebilir.

4.4.3. Kontrol Listesine Dayalı Testler

- **Kontrol Listesine Dayalı Testler:**
  - Test uzmanları, bir kontrol listesinde bulunan test koşullarını kapsayacak şekilde testler tasarlar, uygular ve koşar.
  - Kontrol listeleri, tecrübeye, kullanıcı için önemli olan unsurlara veya yazılımın neden ve nasıl başarısız olabileceği bilgisine dayanarak oluşturulabilir.
  - Kontrol listeleri:
    - Otomatik olarak kontrol edilebilecek öğeleri,
    - Giriş/çıkış kriteri olmaya uygun olan öğeleri veya
    - Çok genel olan öğeleri içermez.
  - Kontrol listesi öğeleri genellikle soru şeklindedir ve ayrı ayrı ve doğrudan kontrol edilebilir olmalıdır.
  - Öğeler, gereksinimlere, arayüz özelliklerine, kalite karakteristiğine veya diğer test koşulu biçimlerine ilişkin olabilir.
  - Farklı test çeşitlerini desteklemek için kontrol listeleri oluşturulabilir (ör. kullanılabilirlik testi için 10 kriter).
  - Kontrol listeleri zamanla daha az etkin hale gelebilir ve yeni bulunan hataların yansıtılması için güncellenmelidir.
  - Kontrol listesi, test senaryolarının ayrıntılı olmadığı durumlarda yol gösterici olarak kullanılabilir.
  - Gerçek testlerde bazı değişiklikler olması muhtemeldir, bu da potansiyel olarak daha büyük kapsama ama daha düşük tekrarlanabilirliğe yol açabilir.

4.5. İş Birliğine Dayalı Test Yaklaşımları
4.5.1. İş Birliğine Dayalı Kullanıcı Hikayesi Yazımı

- **Kullanıcı Hikayeleri (3 C'ler):**
  - **Card (Kart):** Kullanıcı hikayesini açıklayan ortam, örneğin dizin kartı veya elektronik panodaki bir giriş.
  - **Conversation (Konuşmalar):** Yazılımın nasıl kullanılacağını belirler, belgeyle veya sözlü olarak olabilir.
  - **Confirmation (Onay):** Kabul kriterlerini tanımlar.
- **Kullanıcı Hikayesi Formatı:**
  - En yaygın format: "[Rol] olarak [gerçekleştirilecek hedef] istiyorum çünkü [rol için ortaya çıkan iş değeri]."
  - Kabul kriterleri bu formatı izler.
- **İş Birlikçi Kullanıcı Hikayesi Yazımı:**
  - Beyin fırtınası ve zihin haritası gibi teknikler kullanılabilir.
  - İş birliği, analiz, yazılım geliştirme ve test etme perspektiflerini dikkate alarak ortak bir vizyon oluşturmayı sağlar.
- **İyi Kullanıcı Hikayeleri (INVEST Kriterleri):**
  - Bağımsız
  - Müzakere edilebilir
  - Değerli
  - Tahmin edilebilir
  - Küçük
  - Test edilebilir
- **Test Edilebilirlik İşareti:**
  - Bir kullanıcı hikayesinin test edilebilir olduğunu gösterir.
  - Eğer paydaş bir kullanıcı hikayesini nasıl test edileceğini bilmiyorsa, hikaye yeterince net değil, değerli bir şeyi yansıtmıyor veya paydaş sadece test konusunda yardıma ihtiyaç duyuyor olabilir.

4.5.2. Kabul Kriterleri

- **Kabul Kriterleri:**
  - Bir kullanıcı hikayesinin, paydaşlar tarafından kabul edilmesi için karşılaması gereken koşulları içerir.
  - Kabul kriterleri, test uzmanları tarafından denenmesi gereken test koşulları olarak görülebilir.
  - Genellikle bir konuşma sonucunda ortaya çıkar.
- **Kabul Kriterlerinin Kullanım Amaçları:**
  - Kullanıcı hikayesinin kapsamını tanımlamak.
  - Paydaşlar arasında fikir birliğini sağlamak.
  - Hem pozitif hem negatif senaryoları açıklamak.
  - Kullanıcı hikayesi kabul testi için esas teşkil etmek.
  - Doğru planlama ve tahminleme yapılmasını sağlamak.
- **Kabul Kriterlerinin Formatları:**
  - Senaryo odaklı: Örneğin, BDD'de kullanılan Given/When/Then formatı.
  - Kural odaklı: Örneğin, madde işaretli kontrol listesi veya girdi-çıktı eşlemesinin tablo haline getirilmiş şekli.
  - Çoğu kabul kriteri bu iki formattan birinde dokümante edilebilir.
  - Ancak, kabul kriterleri iyi tanımlanmış ve açık olduğu sürece ekip başka bir özel format da kullanabilir.

4.5.3. Kabul Testi Güdümlü Yazılım Geliştirme (ATDD)

- **ATDD (Acceptance Test-Driven Development):**
  - Bir önce-test-et yaklaşımıdır.
  - Kullanıcı hikayesini uygulamadan önce test senaryoları oluşturulur.
  - Test senaryoları farklı perspektiflere sahip ekip üyeleri tarafından oluşturulabilir: müşteriler, geliştiriciler ve test uzmanları.
  - Test senaryoları manuel veya otomatik olarak yürütülebilir.
- **Uygulama Süreci:**
  - İlk adım, kullanıcı hikayesinin ve gerekirse kabul kriterlerinin ekip tarafından analiz edildiği ve düzeltildiği bir analiz çalıştayıdır.
  - Ardından, test senaryoları oluşturulur. Bu, ekip tarafından bir bütün olarak veya test uzmanı tarafından yapılabilir.
  - Test senaryoları kabul kriterlerine bağlıdır ve yazılımın nasıl çalıştığına ilişkin örnekler olarak görülebilir.
  - Örnekler ve testler aynı olduğundan, bu terimler sıkça birbirinin yerine kullanılabilir.
- **Test Senaryoları:**
  - İlk test senaryoları genellikle pozitif testlerdir ve istenilen davranışı onaylar.
  - Pozitif testlerin ardından, ekip negatif testler ve fonksiyonel olmayan gereksinim testlerini yapar.
  - Test senaryoları paydaşlar için anlaşılır olacak şekilde ifade edilmelidir.
  - Test senaryoları, gerekli önkoşulları, girdileri ve artkoşulları içeren doğal bir dilde cümleler içerir.
  - Test senaryoları kullanıcı hikayesinin tüm karakteristiklerini ele almalı ve hikayenin ötesine geçmemelidir.
- **Kabul Testleri ve Otomasyon:**
  - Kabulleri test eden kod, uygulama sürecinin bir parçası olarak yazılabilir ve bu testler koşturulabilir gereksinimlere dönüşür.

5.Test Aktivitelerini Yönetme
5.1. Test Planlama
5.1.1. Test Planının Amacı ve İçeriği

Test Planı:

- Bir test projesinin hedeflerini, kaynaklarını ve süreçlerini belirtir.
- Test hedeflerine ulaşmayı sağlayan araçları ve programı dokümante eder.
- Gerçekleştirilen test aktivitelerinin belirlenmiş kriterlerini karşılamaya yardımcı olur.
- Ekip üyeleri ve diğer paydaşlarla iletişim için bir araç olarak hizmet eder.
- Mevcut test politikasına ve stratejisine uyacağını veya uymayacağını açıklar.

- **Test Planlama:**
- Test uzmanına düşünme konusunda rehberlik eder ve gelecekte ortaya çıkabilecek zorluklara karşı hazırlar.
- Test projesi hedeflerine ulaşmak için gereken eforu hesaplamanın faydalı bir yoludur.
- **Tipik İçerik:**
- Test bağlamı (kapsam, hedefler, kısıtlar, esaslar)
- Test projesinin varsayımları ve kısıtları
- Paydaşlar (roller, sorumluluklar, uygunluk, eğitim ihtiyaçları)
- İletişim (şekil, sıklık, dokümantasyon şablonları)
- Risk kaydı (ürün ve proje riskleri)
- Test yaklaşımı (seviyeler, çeşitler, teknikler, çıktılar, kriterler, metrikler, veri ve ortam gereksinimleri, politika ve strateji)
- Bütçe ve zaman çizelgesi

5.1.2. Test Uzmanının Döngü ve Sürüm Planlamasına Katkısı

- **Döngüsel YGYD'lerde Planlama:**
  - İki tür planlama gerçekleşir: sürüm planlaması ve döngü planlaması.
- **Sürüm Planlaması:**
  - Bir ürünün piyasaya sürülmesini planlar.
  - Ürün iş listesini tanımlar ve yeniden tanımlar.
  - Kapsamlı kullanıcı hikayelerini daha küçük gruplara dönüştürür.
  - Test yaklaşımı ve test planının esası olarak hizmet eder.
  - Test uzmanları:
    - Test edilebilir kullanıcı hikayelerinin ve kabul kriterlerinin yazım sürecine katılır.
    - Proje ve kalite riski analizlerinde görev alır.
    - Kullanıcı hikayeleriyle ilişkili test eforunu tahmin eder.
    - Test yaklaşımını belirler ve sürüm için testi planlar.
- **Döngü Planlaması:**
  - Tek bir döngünün sonunu öngörür ve döngüye ait iş listesiyle ilgilidir.
  - Test uzmanları:
    - Kullanıcı hikayelerine ilişkin detaylı bir risk analizine katılır.
    - Kullanıcı hikayelerinin test edilebilirliğini belirler.
    - Kullanıcı hikayelerini görevlere (özellikle test görevleri) ayırır.
    - Tüm test görevleri için test eforunu tahmin eder.
- Fonksiyonel ve fonksiyonel olmayan test nesnesi unsurlarını belirler ve iyileştirir

5.1.3. Giriş Kriterleri ve Çıkış Kriterleri

- **Giriş Kriterleri:**
  - Belirli bir aktivite için önkoşulları tanımlar.
  - Giriş kriterleri karşılanmazsa:
    - Faaliyetin daha zor olması,
    - Daha uzun sürmesi,
    - Daha maliyetli ve daha riskli olması muhtemeldir.
  - Her test seviyesi için tanımlanmalıdır.
  - Tipik giriş kriterleri:
    - Kaynak elverişliliği (insanlar, araçlar, ortamlar, test verisi, bütçe, zaman),
    - Test yazılımı elverişliliği (test esası, test edilebilir gereksinimler, kullanıcı hikayeleri, test senaryoları),
    - Test nesnesinin başlangıç kalite seviyesi (tüm duman testleri geçmiştir).
- **Çıkış Kriterleri:**
  - Bir aktivitenin tamamlandığını anlamak için nelere ulaşılması gerektiğini tanımlar.
  - Tipik çıkış kriterleri:
    - Bütünlük ölçüleri (ulaşılan kapsam seviyesi, çözülmemiş hata sayısı, hata yoğunluğu, başarısız test senaryosu sayısı),
    - Tamamlama kriterleri (planlanan testler koşturuldu, statik test yapıldı, bulunan tüm hatalar raporlandı, tüm regresyon testleri otomatik hale getirildi).
  - Zamanın veya paranın bitmesi de geçerli bir çıkış kriteri olarak kabul edilebilir.
  - Çevik yazılım geliştirmede, çıkış kriterleri genellikle "Tamamlandı Tanımı" olarak adlandırılır ve piyasaya sürülebilir bir öğe için hedef metrikleri tanımlar.
  - Bir kullanıcı hikayesinin geliştirme ve/veya test aktivitelerinin başlaması için yerine getirmesi gereken giriş kriterleri "Hazır Tanımı" olarak adlandırılır.

5.1.4. Tahminleme Teknikleri

- **Test Eforu Tahmini:**
- Test projesinin hedeflerine ulaşmak için gereken testle ilgili iş miktarını tahmin etmeyi içerir.
- Tahminin bir dizi varsayıma dayandığı ve her zaman tahmin hatasına tabi olduğu paydaşlara açıkça belirtilmelidir.
- Küçük görevlere ilişkin yapılan tahminler genellikle büyük görev tahminlerinden daha doğrudur. Bu nedenle, büyük bir görev için tahminleme yapılırken görev daha küçük görevlere bölünerek bunlara yönelik tahminleme gerçekleştirilir.

Tahminleme Teknikleri:

1. **Oranlara Dayalı Tahminleme:**
   - Kuruluş içinde önceden yapılmış projelerden edinilen rakamlar kullanılır.
   - Önceki projelerin oranları yeni projeler için test eforunun tahmin edilmesinde kullanılabilir.
2. **Dışdeğerleme:**
   - Mevcut projede erken ölçümler yapılır.
   - Yeterli gözlem yapıldıktan sonra geri kalan iş için gereken efor dışdeğerleme uygulanarak belirlenir.
3. **Wideband Delphi:**
   - Uzmanlar tecrübeye dayalı tahminler yapar.
   - Her uzmanın tek başına eforu tahmin etmesi istenir, ardından sonuçlar toplanır ve gerektiğinde düzeltilir.
4. **Üç Noktalı Tahminleme:**
   - Uzmanlar tarafından üç tahmin yapılır: en iyimser, en olası ve en kötümser tahmin.
   - Son tahmin bu tahminlerin ağırlıklı ortalaması olarak hesaplanır.

Diğer Teknikler:

- Poker planlama tekniği, Wideband Delphi'nin bir varyantıdır ve çevik yazılım geliştirmede sıkça kullanılır.

5.1.5. Test Senaryosu Önceliklendirme

- Test senaryolarının önceliklendirilmesi, test koşum çizelgesinde belirlenen sırayı etkiler.
- En yaygın kullanılan önceliklendirme stratejileri şunlardır:

1. **Risk Temelli Önceliklendirme**: En önemli risklere sahip test senaryoları öncelikli olarak koşturulur. Örneğin, bir yazılımın kritik bir özelliğinin başarısız olması durumunda ortaya çıkabilecek en yüksek riskli senaryolar önceliklidir.
2. **Kapsama Temelli Önceliklendirme**: Test kapsamını artırmaya yönelik olarak, komut kapsama yüzdesi gibi metrikler kullanılarak test senaryoları sıralanır. Yüksek kapsama sağlayan senaryolar önceliklidir.
3. **Gereksinim Temelli Önceliklendirme**: Paydaşlar tarafından belirlenen gereksinim önceliklerine göre test senaryoları sıralanır. Önemli gereksinimlere dayalı olarak öncelik verilir.

- Test senaryoları, önceliklendirme stratejilerine göre sıralanır ve çalıştırılır.
- Ancak, test senaryoları veya özellikler arasında bağımlılıklar varsa, bu stratejilerin uygulanması zor olabilir. Örneğin, yüksek öncelikli bir test senaryosu, düşük öncelikli bir senaryoya bağımlıysa, önce düşük öncelikli senaryo koşturulmalıdır.
- Test koşum sıralamasında, kullanılabilir kaynakların (örneğin, test araçları, ortamlar, kişiler) erişilebilirliği de dikkate alınmalıdır. Örneğin, belirli bir zaman diliminde yalnızca belirli bir test aracına erişilebiliyorsa, buna göre planlama yapılmalıdır.

5.1.6. Test Piramidi

- Test piramidi, farklı test seviyelerini ve test otomasyonunun bu seviyelere göre nasıl desteklendiğini gösteren bir modeldir.
- Piramit katmanları, test gruplarını ve testlerin ayrıntı düzeylerini temsil eder.
- Piramidin alt katmanları, küçük, izole ve hızlı testlerden oluşur. Bu testler genellikle birim testleri olarak bilinir ve küçük bir fonksiyon parçasını kontrol eder.
- Alt katmanlardaki testlerin sayısı fazla olabilir, çünkü makul bir kapsama sağlamak için çok sayıda test gerekebilir.
- Üst katmanlar ise karmaşık, üst seviye ve uçtan uca testleri temsil eder. Bu testler genellikle daha yavaş ve büyük bir fonksiyonalite parçasını kontrol eder.

- Üst katmanlardaki testlerin sayısı genellikle daha azdır, çünkü kapsamı kontrol etmek için genellikle sadece birkaç test yeterlidir.
- Katmanların sayısı ve adlandırması değişebilir. Örneğin, bir modelde üç katman "birim testleri", "servis testleri" ve "UI testleri" olarak adlandırılabilirken, başka bir modelde "bileşen testleri", "bileşen entegrasyonu testleri" ve "uçtan uca testler" olarak adlandırılabilir.
- Diğer test seviyeleri de test piramidinin içine dahil edilebilir.

5.1.7. Test Çeyrekleri

- Test çeyrekleri, test seviyelerini belirli test çeşitleri, aktiviteleri ve çalışma ürünleri ile gruplar.
- Model, test yönetimini desteklemek için test çeşitlerinin ve test seviyelerinin ilişkisini anlamada kullanılır.
- Testler iş odaklı veya teknoloji odaklı olabilir ve ekibi destekleyebilir veya ürünü eleştirebilir.
- Model, dört çeyrekten oluşur:
    1. Q1: Teknoloji odaklı ve ekibi desteklemek için (örneğin, bileşen ve bileşen entegrasyon testleri).
    2. Q2: İş odaklı ve ekibi desteklemek için (örneğin, fonksiyonel testler, kullanıcı hikayesi testleri).
    3. Q3: İş odaklı, ürünün kritiğini yapmak için (örneğin, keşif testi, kullanılabilirlik testi).
    4. Q4: Teknoloji odaklı, ürünün kritiğini yapmak için (örneğin, duman testleri, fonksiyonel olmayan testler).

5.2. Risk Yönetimi

- Risk yönetimi, hedeflere ulaşma ve ürün kalitesini artırma üzerinde olumlu etkilere sahiptir.
- Temel risk yönetimi adımları risk analizi ve kontrolünü içerir.
- Testlerin risk odaklı olarak seçilmesi, önceliklendirilmesi ve yönetilmesi, risk bazlı test yaklaşımını oluşturur.

5.2.1. Risk Tanımı ve Risk Özellikleri

- Risk, olumsuz sonuçlara yol açabilecek potansiyel olay, tehlike veya durumdur.
- Bir riskin iki ana özelliği vardır: olasılık (riskin gerçekleşme olasılığı) ve etki (riskin sonuçları).
- Risk seviyesi, bir riskin önemini gösterir; yüksek risk seviyeleri daha fazla dikkat gerektirir.

5.2.2. Proje Riskleri ve Ürün Riskleri

- Yazılım testlerinde iki tür risk vardır: proje riskleri ve ürün riskleri.
- Proje riskleri projenin yönetimi ve kontrolüyle ilgilidir ve organizasyonel, insan kaynaklı, teknik ve tedarikçi sorunları içerir.
- Proje riskleri, zaman çizelgesini, bütçeyi ve kapsamı etkileyebilir, bu da projenin hedeflerine ulaşma kapasitesini etkiler.
- Ürün riskleri, ürün gereksinimleriyle ilgilidir ve eksik fonksiyonalite, hatalı hesaplamalar, performans sorunları gibi sorunları içerir.
- Ürün riskleri, kullanıcı memnuniyetsizliği, gelir kaybı, güven kaybı, yüksek bakım maliyetleri gibi olumsuz sonuçlara yol açabilir.

5.2.3. Ürün Riski Analizi

- Ürün riski analizinin amacı, ürün risklerine dikkat çekerek test çabasını ve kalan riski azaltıcı faaliyetleri yönlendirmektir.
- İdeal olarak, ürün riski analizi YGYD'nün erken aşamalarında başlar.
- Analiz, risk belirleme ve risk değerlendirmesinden oluşur.
- Risk belirleme, kapsamlı bir risk listesi oluşturmayı içerir ve beyin fırtınası, çalıştaylar, görüşmeler gibi teknikler kullanılabilir.
- Risk değerlendirmesi, belirlenen risklerin kategorilere ayrılması, risk olasılığı, etkisi ve seviyesinin belirlenmesi, önceliklendirme ve risk azaltma stratejilerinin tasarlanmasını içerir.
- Risk değerlendirmesinde nicel veya nitel bir yaklaşım veya her ikisinin bir kombinasyonu kullanılabilir.
- Ürün riski analizi, testlerin bütünlüğünü ve kapsamını etkiler.
- Sonuçlar, şu amaçlarla kullanılır:
    - Test kapsamını belirlemek.
    - Test seviyelerini ve çeşitlerini önermek.
    - Kullanılacak test tekniklerini ve kapsamı belirlemek.
    - Test çabasını tahmin etmek.
    - Kritik hataları erken tespit etmek için testleri önceliklendirmek.
- Riski azaltmak için ek test dışı aktiviteleri belirlemek

5.2.4. Ürün Risk Kontrolü

- Ürün risk kontrolü, belirlenen ve değerlendirilen ürün risklerine karşı alınan önlemleri içerir.
- Bu kontrol, risk azaltma ve risk gözetimini içerir.
- Risk azaltma, risk seviyesini azaltmak için önerilen aksiyonların uygulanmasını içerir.
- Risk gözetiminin amacı, alınan önlemlerin etkili olmasını sağlamak ve risk değerlendirmesini iyileştirmektir.
- Ürün risk kontrolü için farklı yanıt seçenekleri vardır, bunlar arasında test yoluyla riskin azaltılması, risk kabulü, risk transferi ve acil durum planı bulunur.

Ürün riskini azaltmak için alınabilecek aksiyonlar şunlardır:

- Deneyimli test uzmanlarını seçmek
- Uygun test bağımsızlık seviyesini uygulamak
- Gözden geçirmeler ve statik analizler yapmak
- Uygun test tekniklerini ve kapsamı kullanmak
- Uygun test çeşitlerini uygulamak
- Dinamik testler, özellikle regresyon testleri yapmak