1.Ünite - Yazılım Kalite ve Testi

Yazılım Testi ve Amacı

Yazılım testi, bir yazılımın **beklenen gereksinimleri karşılayıp karşılamadığını** inceleme sürecidir.

Amaçlar:

- Maliyet ve zaman tasarrufu sağlamak
- Hataları önceden bulmak ve tekrarını önlemek
- Ürünün kalitesini ve güvenilirliğini artırmak

Temel Noktalar:

- Test, **kodlamadan önce başlar** ve tüm süreç boyunca sürer.
- **Normal ve anormal koşullar** altında yapılmalıdır.
- Tüm senaryolar test edilemez → **kritik alanlara odaklanılır.**

Doğru Bilinen Yanlışlar

Yanlış	Gerçek
Test pahalıdır.	Hata düzeltmek daha pahalıdır.
Test zaman kaybıdır.	Erken test zaman kazandırır.
Sadece bitmiş yazılım test edilir.	Test süreci fikirden başlar.
Yazılım tamamen test edilebilir.	Sonsuz olasılık nedeniyle imkansızdır.
Test edilmiş yazılım hatasızdır.	%100 hatasız yazılım yoktur.
Hatalar testçiden kaynaklanır.	Ekipte herkes sorumludur.
Testçiler kaliteyi sağlar.	Kalite ekip işidir.
Otomasyon sadece hız içindir.	Gereksinim değişikliklerine uyum sağlar.
Herkes test yapabilir.	Herkes aynı kalitede test yapamaz.
Testçinin görevi hata bulmaktır.	Analiz, raporlama ve öneri de yapar.

Yazılım Testinin 7 İlkesi

1. Test hataların **varlığını**, yokluğunu değil gösterir.
2. Hatasız yazılım **ihtiyacı karşılamıyorsa değersizdir.**
3. Test **erken yapılmalıdır.**
4. **Kapsamlı, kusursuz test mümkün değildir.**
5. Hatalar **belirli alanlarda yoğunlaşır.**
6. **Tarım ilacı paradoksu:** Aynı testler yeni hataları bulamaz.
7. Test **yazılım türüne göre değişir.**

Test Uzmanının Görevleri

- Yazılımı ve kullanıcı ihtiyaçlarını tanımak
- Risk analizi ve test planı hazırlamak
- Testleri yürütmek, hataları raporlamak

- Düzeltmeleri doğrulayıp rapor sunmak

Özet

- Testin amacı **hatasızlık değil, kaliteyi sağlamaktır.**
- **Erken, sürekli, risk temelli test** esastır.
- **Kalite, tüm ekibin sorumluluğudur.**
- Başarılı yazılım, **kullanıcı ihtiyacını** karşılayandır.

2.Ünite - Yazılım Hataları

Yazılım Hatası Kavramı

Yazılım hataları kaçınılmazdır, ancak **doğru tespit ve yönetim** yazılım kalitesini belirler.

- **Hata (Bug):** Yazılımın beklenen işlevi yapmasını engelleyen durum.
- **Kusur (Defect):** Gereken kriterleri karşılamayan sistem veya bileşen.
- **İnsan Hatası (Error):** Geliştirici, analist veya tester'ın yanlış eylemi.
- **Arıza (Failure):** Sistemin beklenenden sapması.
- **Anomali (Anomaly):** Beklentiden sapma; bazen olumlu sonuçlar doğurabilir (örneğin beklenenden fazla kullanıcı).

Hata Türleri

- **Sözdizimi Hataları:** Programlama dilinin kurallarına aykırı ifadeler (örnek: yazım yanlışı).
- **Çalışma Zamanı Hataları:** Program çalışırken ortaya çıkar (örnek: olmayan dosyaya erişim).
- **Mantıksal Hatalar:** Algoritma doğru sonuç vermez (örnek: `*` yerine `+` yazmak).
- **Mimari/Tasarım Hataları:** Yanlış iş sırası, zayıf tasarım veya eksik analiz.
- **Belgelendirme Hataları:** Dokümantasyon eksiklikleri veya yanlış bilgiler.

Hata Yönetimi ve Ölçütler

- **Hata Yoğunluğu:** Toplam hata sayısının kod büyüklüğüne oranı.
- **Tespit Yüzdesi (DDP):** Bulunan hataların toplam hata sayısına oranı.
- **Hata Toleransı:** Sistem yanlış girdilerle de çalışabiliyorsa yüksek toleransa sahiptir.
- **Arıza Oranı** / **Ortalama Arıza Süresi:** Güvenilirliği ölçmede kullanılır.

Hataların izlenmesi, kaydedilmesi ve çözülmesi **hata yönetimi araçları** ile yapılır.
Hata tahminleme ve **kusur ağacı analizi (FTA)** gibi yöntemler test sürecini yönlendirir.

Tarihteki Önemli Yazılım Hataları

- **Mariner-1 (1962):** Kodda yazım hatası → roket yörüngeden çıktı, 80 milyon $ zarar.
- **Therac-25 (1985):** Radyasyon cihazı hatası → 3 hasta öldü.
- **Patriot Füzesi (1991):** Zamanlama hatası → 29 asker hayatını kaybetti.
- **Pentium İşlemci (1993):** Bölme hatası → 475 milyon $ zarar.

- **Mars Climate Orbiter (1999):** Birim uyuşmazlığı → 125 milyon $ kayıp.
- **Y2K (2000):** Yıl formatı hatası → küresel sistem güncellemeleri gerekmiştir.
- **Google Malware (2009):** Yanlış karakter girişi → tüm siteler zararlı olarak işaretlendi.
- **Toyota Prius (2014):** Yazılım hatası → milyonlarca araç geri çağrıldı.

Özet

- Hatasız yazılım **mümkün değildir**, önemli olan hataları **doğru yönetmektir.**
- Hata türleri farklı aşamalarda ve farklı etkilerle ortaya çıkar.
- Tarihte küçük hatalar milyar dolarlık kayıplara ve can kayıplarına yol açmıştır.
- Etkili hata yönetimi, yazılım kalitesinin ve güvenilirliğinin temelidir.

3.Ünite - Yazılım Risk Analizi ve Yazılımda Ölçme

Yazılım Risk Analizi

Yazılım projelerinde **zaman, maliyet ve kalite tahminleri** genellikle riskler nedeniyle sapar.
Risk, projenin hedeflerini etkileyebilecek **belirsiz bir durumdur**.

Risk türleri:

- **Proje riskleri:** Zaman, maliyet, personel uyumsuzluğu, planlama hataları.
- **Teknik riskler:** Yeni teknolojiler, yetersiz test, belirsiz dokümantasyon.
- **İşletme riskleri:** Pazar değişimleri, doğal afetler, döviz dalgalanmaları.

Risk yönetimi süreci:

1. **Risk değerlendirmesi:** Risklerin tanımlanması, analizi ve önceliklendirilmesi.
2. **Risk planlaması:** Risk oluşmadan önlem alma stratejileri geliştirme.

Risk stratejileri:

- **Önleme:** Riskin oluşma olasılığını azaltır.
- **Minimize etme:** Etkisini ve zararını azaltır.
- **Acil eylem planı:** Gerçekleştiğinde düzeltici adımlar uygular.

Risk analizi araçları:

- Risk tablosu (riskin açıklaması, olasılığı, etkisi, alınacak önlem vb.)
- FMEA yöntemi (Risk Öncelik Numarası – RPN = Severity × Priority × Likelihood).

Küçük RPN → yüksek öncelikli risk.

Yazılımda Ölçme

Ölçme: Gerçek dünyadaki varlıkların özelliklerine sayısal değer atama işlemidir.
Amaç, **durumu analiz etmek, geçmişle karşılaştırmak ve öngörüde bulunmaktır.**

Ölçme türleri:

- **Doğrudan ölçme:** Boy, kilo, satır sayısı gibi fiziksel değerler.
- **Dolaylı ölçme:** Test puanı, zeka, performans gibi dolaylı göstergelerle ölçüm.
- **Türetilmiş ölçme:** İki ya da daha fazla ölçümden türetilir (örnek: hız = yol/zaman).

Yazılımda ölçme alanları:

- **Temel ölçüler:** Kod satır sayısı, test süresi, hata sayısı.
- **Türetilmiş ölçüler:**
  - Hata yoğunluğu = hata sayısı / kod uzunluğu
  - Gereksinim değişim oranı = değişen gereksinim / ilk gereksinim

Yazılım büyüklüğü ölçütleri:

- **Uzunluk:** Kod veya doküman satır sayısı.
- **İşlevsellik:** Kullanıcıya sunulan fonksiyon sayısı.
- **Karmaşıklık:** Algoritma veya problem yapısının zorluğu.
- **Tekrar kullanım:** Kodun yeniden kullanılabilirliği.

Özet

- Risk, yazılım projelerinin kaçınılmaz bir parçasıdır.
- Başarılı projeler, riskleri **tanımlayıp yönetebilen** projelerdir.
- Yazılımda ölçme, **kaliteyi ve verimliliği** kontrol etmenin temel aracıdır.
- Hem risk yönetimi hem de ölçme, yazılım kalitesinin sürdürülebilirliği için gereklidir.

4.Ünite - Yazılım Testine Ait Kavramlar

Test Verileri ve Test Senaryoları

Test senaryosu: Belirli bir işlevi veya gereksinimi test etmek için oluşturulan girdiler, ön koşullar, beklenen ve gerçekleşen sonuçlardan oluşan yapıdır.

- **Bloke test senaryosu:** Gerekli önkoşullar sağlanmadığı için çalıştırılamaz.
- **Üst seviye test senaryosu:** Gerçek veriler içermez, soyut olarak yazılır.
- **Alt seviye test senaryosu:** Somut verilerle hazırlanır.
- **Test spesifikasyonu:** Test tasarımını, senaryolarını ve prosedürlerini içeren belgedir.
- **Test verisi:** Testlerde kullanılacak gerçek veya üretilmiş veridir.
- **Test verisi yönetimi:** Test verilerinin oluşturulması, güncellenmesi ve korunması sürecidir.

Test Başlatma, Bitiş ve Kabul

Kabul kriteri: Yazılımın kullanıcı veya müşteri beklentilerini karşılama şartıdır.
Test kapanışı: Test sonuçlarının ve tecrübelerinin arşivlendiği son aşamadır.
Beklenen sonuç: Gereksinimlere göre sistemin göstermesi gereken davranış.
Gerçekleşen sonuç: Test sırasında gözlemlenen davranış.
Test karşılaştırması: Beklenen ve gerçekleşen sonuçların farkını ölçme işlemi.

Test Yönetimi ve Süreci

Test yönetimi: Testin planlanması, gözetimi ve kontrol sürecidir.
Test yöneticisi: Test faaliyetlerinin yürütülmesinden sorumlu kişidir.
Test yürütme: Testlerin manuel veya otomatik olarak uygulanması.
Test yürütme otomasyonu: Testlerin otomatik araçlarla yapılması.
Test süreci: Planlama, analiz, tasarım, yürütme, değerlendirme ve kapanış adımlarını içerir.

Test süreç iyileştirme bildirgesi: Çevik yaklaşıma uygun şekilde test süreçlerini geliştirme değerlerini tanımlar.

Test Planı

Test planı: Testin kapsamını, yaklaşımını, zamanlamasını, kaynaklarını ve görev dağılımını belirleyen dokümandır.
Master test planı: Birden fazla test seviyesini kapsar.
Faz test planı: Belirli bir aşama için hazırlanır.
Seviye test planı: Bir test seviyesini (ör. sistem testi) kapsar.
Test iyileştirme planı: Test sürecinin güçlü ve zayıf yönlerini analiz ederek geliştirme planı oluşturur.

Test Araçları

Test otomasyonu: Test yürütme, analiz ve raporlama işlemlerini kolaylaştıran yazılımlar.
Test ortamı: Testin yürütüldüğü donanım, yazılım ve simülatörlerin bulunduğu yapı.
Test altyapısı: Test araçları, ortam, prosedürler ve dokümantasyonun tümü.
Test aracı: Test sürecinin bir veya birkaç aşamasını destekleyen yazılım ürünüdür.

Test Dokümantasyonu

Test değerlendirme raporu: Tüm testlerin sonuçlarını ve gözlemleri içeren rapor.
Test kaydı: Testlerin yürütülme detaylarının kronolojik kaydı.
Test özet raporu: Test aktiviteleri ve sonuçlarının genel özeti.

Diğer Kavramlar

- **Esas test kümesi:** Yazılımın tamamını test etmek için gereken test grubudur.
- **Test bağımsızlığı:** Testlerin tarafsız yapılmasını sağlar.
- **Test güdümlü geliştirme (TDD):** Önce testlerin yazıldığı yazılım geliştirme yaklaşımı.
- **Test seviyesi:** Birim, entegrasyon, sistem ve kabul testi gibi aşamaları kapsar.
- **Test olgunluk modeli (TMMi):** Test süreçlerini iyileştirme çerçevesi.
- **Test stratejisi:** Organizasyon genelinde test yaklaşımını tanımlar.
- **Test tekrarlanabilirliği:** Aynı testlerin aynı sonuçları verebilme yeteneği.

Özet

- Yazılım testi birçok kavramdan oluşur: senaryo, plan, araç, ortam, süreç ve dokümantasyon.
- Test süreci yalnızca hata bulmak değil, **kaliteyi garanti altına almak** içindir.
- Başarılı test yönetimi; **iyi planlama, doğru araç kullanımı ve sürekli iyileştirme** ile sağlanır.

5.Ünite - Yazılım Test Çeşitleri

Test Seviyeleri

1. Birim Testi:
Yazılımın en küçük parçalarının (fonksiyon, modül vb.) doğru çalıştığını kontrol eder.

- Genellikle geliştiriciler tarafından yapılır.
- Araçlar: JUnit, NUnit, PyTest.

2. Entegrasyon Testi:
Modüller arasındaki veri alışverişi ve etkileşimi kontrol eder.

- “Modüller birlikte çalışıyor mu?” sorusuna yanıt verir.
- Yaklaşımlar: Yukarıdan aşağı, aşağıdan yukarı, karma.

3. Sistem Testi:
Yazılımın tamamı test edilir.

- İşlevsel, performans, güvenlik ve uyumluluk testlerini içerir.

4. Kabul Testi:
Müşteri veya kullanıcı tarafından yapılır.

- Yazılımın gereksinimleri karşıladığı doğrulanır.
- Türleri: Alfa (laboratuvar ortamında) ve Beta (gerçek kullanıcı ortamında).

Test Türleri

İşlevsel Testler

Yazılımın **“ne yaptığı”** kontrol edilir.

- Kara Kutu Testi, Sınır Değer Analizi, Karar Tablosu Testi gibi teknikler kullanılır.

İşlevsel Olmayan Testler

Yazılımın **“nasıl davrandığı”** incelenir.

- Performans, güvenlik, kullanılabilirlik, yük, stres testleri bu gruptadır.

Geriye Dönük Test (Regression Test)

Değişikliklerden sonra sistemin eski işlevlerinin bozulmadığını doğrular.

Tekrar Testi (Re-test)

Düzeltilen hatanın yeniden test edilmesidir.

Test Teknikleri

Beyaz Kutu Testi:
Kodu bilerek yapılan testtir. Kontrol yapıları, döngüler ve karar noktaları incelenir.

Kara Kutu Testi:
Kullanıcı bakış açısıyla yapılır; iç kod bilinmez, sadece girdiler ve çıktılar test edilir.

Gri Kutu Testi:
Hem iç hem dış bilgilerin kullanıldığı karma yaklaşımdır.

Diğer Test Çeşitleri

- **Duman (Smoke) Testi:** Ana işlevlerin temel çalışmasını hızlıca kontrol eder.
- **Sanity Testi:** Yeni değişikliklerin genel işleyişi bozmadığını doğrular.
- **Performans Testi:** Yazılımın hız, tepki süresi ve kararlılığını ölçer.
- **Stres Testi:** Aşırı yük altında sistem davranışını test eder.
- **Yük Testi:** Belirli bir kullanıcı yükü altında performans ölçümü.
- **Kullanılabilirlik Testi:** Arayüzün kullanıcı dostu olup olmadığını inceler.
- **Güvenlik Testi:** Yetkisiz erişim ve veri sızıntılarına karşı korumayı test eder.
- **Uyumluluk Testi:** Yazılımın farklı cihaz, tarayıcı veya işletim sistemlerinde çalışabilirliğini kontrol eder.

Özet

- Testler **seviyelere** (birimden kabule) ve **türlere** (işlevsel/olmayan) ayrılır.
- Testin amacı, yalnızca hata bulmak değil, **kaliteyi korumaktır.**
- Her test türü farklı riskleri azaltır ve yazılımın güvenilirliğini artırır.
- **Doğru test stratejisi**, hem kullanıcı memnuniyetini hem de ürün başarısını belirler.

6.Ünite - Yazılım Test Sınıflandırmaları

Testlerin Amaçlarına Göre

- **Doğrulama (Verification):** Yazılımın gereksinimlere uygun geliştirilip geliştirilmediğini kontrol eder.
- **Geçerleme (Validation):** Yazılımın kullanıcı ihtiyaçlarını karşılayıp karşılamadığını ölçer.

Test Seviyelerine Göre

- **Birim Testi:** Kodun en küçük parçaları (fonksiyon, sınıf) test edilir.
- **Entegrasyon Testi:** Modüller arasındaki veri ve kontrol akışı test edilir.
- **Sistem Testi:** Yazılım bir bütün olarak ele alınır.
- **Kabul Testi:** Kullanıcı veya müşteri onayı için yapılır.

Test Yaklaşımlarına Göre

- **Beyaz Kutu Testi:** Kodun iç yapısı ve akışı incelenir.
- **Siyah Kutu Testi:** Kodun içi bilinmeden, sadece giriş-çıkışa bakılır.
- **Gri Kutu Testi:** Hem kullanıcı hem geliştirici bakışı birleştirilir.

Testin Zamanına Göre

- **Statik Test:** Kod çalıştırılmadan yapılır (gözden geçirme, analiz).
- **Dinamik Test:** Kod çalıştırılarak hatalar bulunur.

Testin Otomasyon Durumuna Göre

- **Manuel Test:** Test senaryoları insan tarafından uygulanır.
- **Otomatik Test:** Araçlar ve scriptlerle otomatik yapılır.

Diğer Test Türleri

- **Regresyon Testi:** Yeni değişikliklerin mevcut özellikleri bozup bozmadığını kontrol eder.
- **Yük Testi:** Sistem yüksek kullanıcı sayısında nasıl davranıyor test edilir.
- **Performans Testi:** Yazılımın hız, tepki süresi, kaynak kullanımı ölçülür.
- **Güvenlik Testi:** Sistem zafiyetleri tespit edilir.
- **Kurtarma (Recovery) Testi:** Sistem çökmeden sonra toparlanabiliyor mu kontrol edilir.
- **Uyumluluk Testi:** Yazılım farklı ortam ve cihazlarda doğru çalışıyor mu test edilir.

Özet

- Yazılım testleri; *ne zaman, nasıl, kim tarafından ve neyi amaçlayarak*yapıldığına göre sınıflandırılır.

- Amaç; her düzeyde hatayı erken bulmak, kaliteyi garanti altına almak ve kullanıcıya güvenilir bir ürün sunmaktır.

7.Ünite - Yazılım Testlerinde Kutu Yaklaşımı

Kara Kutu Yaklaşımı

- Kodun iç yapısı bilinmeden test yapılır; yalnızca **girdi ve çıktı** incelenir.
- Uygulamanın **fonksiyonelliğini** test eder.
- Kod incelemesi gerekmez, kullanıcı gözünden test yapılır.
- **Bulabileceği hatalar:**
    - Eksik veya hatalı fonksiyonlar
    - Arayüz ve veri bağlantısı hataları
    - Davranış, performans, başlatma/sonlandırma sorunları

Kara kutu test türleri:
Ad-Hoc, Araştırma, Fonksiyonellik, Stres, Yük, Kullanılabilirlik, Duman, Yenilenme, Seviye, Kullanıcı Kabul, Alfa, Beta testleri.

Kara kutu teknikleri:

- **Denklik Sınıfı:** Benzer davranış gösteren veri grupları oluşturulur.
- **Sınır Değer:** Değişkenlerin alt ve üst sınırları test edilir.
- **Karar Tablosu:** Girdi-çıktı ilişkileri tabloya yazılır.
- **Sınıflandırma Ağacı:** Girdiler hiyerarşik şekilde modellenir.
- **İkili Test:** İki değişkenin kombinasyonları test edilir.
- **Durum Geçişi:** Sistem durumlarının geçişleri incelenir.
- **Kullanıcı Senaryosu:** Kullanıcı davranışlarına göre test yapılır.

Beyaz Kutu Yaklaşımı

- Kodun **iç yapısı** ve **mantığı** test edilir.
- Geliştiriciler tarafından yapılır.
- **Avantajı:** Kod optimizasyonu ve hata nedeninin kolay tespiti.
- **Dezavantajı:** Zaman alır ve yüksek teknik bilgi gerektirir.

Beyaz kutu test türleri:
Birim Testi, Statik/Dinamik Analiz, Açıklama Kapsamı, Güvenlik Testi, Değişim Testi.

Beyaz kutu teknikleri:

- Ortam Kontrolü
- Kod Yürütme
- Resmî İnceleme
- Akış Kontrolü Testi
- Temel Yol Testi
- Veri Akış Testi
- Döngü Testi

Gri Kutu Yaklaşımı

- **Kara** ve **Beyaz Kutu** yöntemlerinin birleşimidir.
- Testçi kodun tamamını göremez ama bazı iç verilere ve tasarıma erişebilir.
- **Avantajı:** Tarafsız test sağlar, veri tabanı ve dokümanlardan yararlanır.
- **Dezavantajı:** Kod erişimi sınırlı, test kapsamı dar olabilir.

Gri kutu teknikleri:
Ortogonal Dizi Testi, Matris Testi, Regresyon Testi, Model Testi.

Genel Karşılaştırma

Özellik	Kara Kutu	Beyaz Kutu	Gri Kutu
İç yapı bilgisi	Yok	Tam	Kısmi
Test eden kişi	Kullanıcı/Testçi	Geliştirici	Testçi
Kapsam	Az	Geniş	Orta
Dış etkiler	Önemli	Önemsiz	Kısmi
Uygulama türü	Fonksiyonel	Yapısal	Karma

Özet

- Kara kutu, yazılımın **ne yaptığına**,
- Beyaz kutu, **nasıl yaptığına**,
- Gri kutu ise **her ikisine birden** odaklanır.

8.Ünite - Yazılım Test Otomasyonları

Yazılım test otomasyonu kavramını, amaçlarını, özelliklerini ve piyasada bulunan çeşitli araçların karşılaştırmasını ele almaktadır.

Yazılım Test Otomasyonu Kavramı

Yazılım testinin temel motivasyonu, yazılımın beklenen şekilde çalıştığını ve her zaman doğru sonuçlar ürettiğini kanıtlamaktır. Test süreçleri manuel (insanlar tarafından) veya otomatik (makineler/yazılımlar tarafından) olarak gerçekleştirilebilir.

- **Otomasyon Testi:** Test planı kapsamını en üst düzeye çıkarmak ve yazılım güvenilirliğini artırmak için bir çözümdür. Manuel testlere göre daha hızlıdır, maliyeti düşürebilir ve insan hatalarını ortadan kaldırmaya yardımcı olur. Özellikle aynı testin çok fazla sayıda ve farklı girdilerle yapılması gereken durumlarda gereklidir.
- **Manuel Test:** Kullanıcı arayüzü testlerinde daha verimli olabilir ancak otomasyon testleri kadar güvenilir sonuçlar üretmez, daha yavaştır ve karmaşık projeler için uygun değildir.

Özellik	Otomasyon Testi	Manuel Test
Hız	Daha hızlıdır.	Daha yavaştır.
Tekrarlama	Birçok testi defalarca çalıştırabilir.	Bir veya iki defa çalıştırılacak testlerde uygundur.
Maliyet	Maliyetli olabilir.	Genellikle daha az maliyetlidir.
Sonuçlar	Daha doğru ve stabil sonuçlar üretebilir.	Otomasyon testleri kadar güvenilir değildir.
Verimlilik	Regresyon testlerinde verimli çalışır.	Kullanıcı arayüzü testlerinde daha verimlidir.

Web Uygulama Testinin Adımları

Web uygulama testleri, doğruluğu sağlamak, güveni artırmak ve riskleri azaltmak gibi avantajlar sunar16. İzlenmesi gereken beş ana adım şunlardır:

1. **Fonksiyonellik Testi:** Uygulamanın doğru çalışıp çalışmadığını kontrol eder.

2. **Arayüz Testi:** Web sunucusu ile uygulama sunucusu arasındaki etkileşimi kontrol eder.
3. **Uyumluluk Testi:** Uygulamanın çeşitli cihazlar ve tarayıcılarla (tarayıcı ve işletim sistemi uyumluluğu) uyumlu olmasını sağlar.
4. **Performans Testi:** Uygulamanın ağır yük altında çalışıp çalışmadığını ve yanıt verip vermediğini kontrol eder.
5. **Güvenlik Testi:** Uygulamanın yetkisiz erişim veya kötü amaçlı yazılımlara karşı korunmasını sağlar.

Web Tabanlı Test Otomasyon Araçları ve Karşılaştırması

Piyasada Acunetix, FitNesse, JMeter, Katalon Studio, LoadRunner, QTP (HP UFT), Ranorex Studio, Sahi Pro, Selenium, Telerik Test Studio, TestComplete, TestIO, Testing Whiz ve Webload gibi birçok web tabanlı test otomasyon aracı bulunmaktadır.

Bu araçlar, doğru seçim yapabilmek için çeşitli kriterlere göre karşılaştırılır:

1. Genel Özellikler

- **Test Stili:** Araçların birçoğu farklı test stillerini (API, Masaüstü, Mobil, Web) desteklerken, bazıları yalnızca tek bir stili (Acunetix, FitNesse, Webload) destekler.
- **Açık Kaynaklı/Lisanslı & Maliyet:** İncelenen 14 araçtan sadece **FitNesse, JMeter ve Selenium** tamamen açık kaynaklı ve ücretsizdir. Kalan araçlar lisanslıdır ve genellikle yüksek maliyetlidir.
- **Kararlı Sürüm:** Tüm araçlar sık sık güncellenmektedir, bu da değişen yazılım ihtiyaçlarını karşılayabildiklerini gösterir.

2. Gereksinimler

- **Donanım Gereksinimi (CPU/RAM):** Araçlar genellikle yüksek CPU ve RAM gereksinimlerine sahiptir. Örneğin, JMeter çapraz platform desteği nedeniyle diğer araçlardan daha yüksek bellek gerektirir.
- **Kullanıcı Deneyimi Gereksinimi:** Bazı araçlar (Acunetix, FitNesse, Katalon Studio, QTP, Ranorex, Telerik Test Studio, TestComplete, TestIO, Webload) kullanımı kolay olarak belirtilirken, **JMeter, LoadRunner, Sahi Pro, Selenium, Testing Whiz** gibi popüler araçlar programlama deneyimi gerektirir.

3. Teknik Uyumluluklar

- **Dil Uyumluluğu:** Bazı araçlar tek bir dili (QTP: VBScript) desteklerken, TestComplete gibi bazıları birden çok dili (C++, C#, DelphiScript, JavaScript, Python, VBScript vb.) destekler.
- **Platform Uyumluluğu:** Çoğu araç Windows'ta çalışır, bazıları (FitNesse, JMeter, Katalon Studio, Selenium) çapraz platform desteği sunar, ancak genellikle MAC OS desteği sınırlıdır.
- **Tarayıcı Uyumluluğu:** Web tabanlı araçların çoğu tüm tarayıcılarda çalışır, ancak bazıları (QTP) belirli bir sürümün üzerindeki tarayıcıları bekler.
- **IDE Uyumluluğu:** Bazı araçlar kendi özgün IDE'lerini (QTP, Ranorex, TestComplete, Testing Whiz, Webload) kullanırken, bazıları mevcut IDE'leri (JMeter, Selenium: Java destekli IDE'ler; Katalon Studio: Java, Android SDK; Telerik: Visual Studio) kullanır.

4. Test Özellikleri

- **Test Türü:** Her araç farklı alanlarda uzmanlaşır. Örneğin, Acunetix güvenlik testlerine, FitNesse kabul testlerine, LoadRunner ve JMeter performans/yük testlerine odaklanır. Araç seçimi, uygulamada yapılacak test tipine göre belirlenmelidir.
- **Diğer Araçlarla Entegrasyon:** Bamboo, Jenkins ve Jira en yaygın entegrasyonlardır. **Selenium**, diğer ücretsiz veya lisanslı birçok araçla entegre çalışabilmesiyle öne çıkar.
- **Test Raporu:** Çoğu araç CSV, HTML, XML ve PDF gibi standart formatları destekler. **Selenium**'un yerel bir raporlama olanağı sunmaması önemli bir dezavantajdır.

5. Teknik Destek

- **Müşteri Desteği:** Acunetix, Katalon Studio, QTP, Ranorex, TestComplete, TestIO, Testing Whiz ve Webload gibi birçok lisanslı araçta $7/24$ çevrimiçi teknik destek mevcuttur. **Selenium** profesyonel destek sunmazken, JMeter sınırlı topluluk desteği ile ücretsizdir.

- **Bloglar:** FitNesse, LoadRunner ve Sahi Pro dışındaki tüm araçların web sitesinde blog sistemi bulunur.
- **Hata Takibi:** Yalnızca **Acunetix, JMeter, Selenium ve Telerik Test Studio** araçlarında hata takip özelliği bulunur.

Sonuç

Yazılım test otomasyonu araçları, tekrar eden veya yüksek veri içeren testlerde zaman kazandırır ve stabil sonuçlar sağlar. Ancak literatürdeki çalışmalar, hiçbir aracın diğerlerinden kesin olarak iyi olmadığını göstermektedir. Bu nedenle, en uygun otomasyon aracı seçimi; uygulamanın özellikleri, finansal koşullar ve teknik destek gibi birçok kritere göre dikkatlice yapılmalıdır.

9.Ünite - Yazılım Kalitesi

---

1. Kalite Nedir?

Kalite, bir ürünün **gereksinimleri karşılama derecesidir**.
Elle tutulur ürünlerde ölçmesi kolaydır; yazılımda ise soyut olduğu için çeşitli **standartlarla** ölçülür.

Kalite;

- kalite kontrol,
- kalite yönetimi,
- kalite güvencesi,
- kalite planlama
  gibi süreçlerin birleşimidir.

---

2. Doğru Bilinen Yanlışlar

- **“Kalite ölçülemez.”** → Ölçülebilir. (Hata maliyeti, uygunsuzluk maliyeti vb.)
- **“Kalite başta pahalıdır.”** → Baştan sağlamak daha ucuzdur.
- **“Hataları alt kademe yapar.”** → Üst kademenin kararları daha büyük hatalara sebep olabilir.
- **“Kaliteyi kalite birimi başlatır.”** → Üst yönetimden başlamalıdır.

---

3. Kalite Maliyeti

Bir yazılımın kaliteli olması için veya kalitesizliğin giderilmesi için harcanan her şey bu kapsamdadır.

- **Uygunsuzluk maliyeti:** Hataların bedeli.
- **Uygunluğu sağlama maliyeti:** Kaliteyi sağlamak için yapılanlar.
- **Hata bulma maliyeti:** Test, gözden geçirme.
- **Hata önleme maliyeti:** Eğitim, süreç iyileştirme.

---

4. Yazılım Kalitesi

Kaliteli yazılım;

- kabul edilebilir hata seviyesine sahip,
- müşteri gereksinimlerini karşılayan,
- zamanında ve bütçesinde teslim edilen,
- bakımı kolay olan yazılımdır.

İki tür kalite:

- **Fonksiyonel kalite:** Yazılım istenilen işi doğru yapıyor mu?
- **Yapısal kalite:** Dayanıklılık, sürdürülebilirlik, güvenilirlik.

Hata ne kadar geç aşamada bulunursa **düzeltme maliyeti o kadar artar**.

---

5. YKG (Yazılım Kalite Güvencesi)

Amaç: **Hataları bulmak değil, oluşmasını önlemek.**

YKG > Doğrulama (verification) > Geçerleme (validation) > Test

- **Statik yöntemler:** Kod çalıştırmadan yapılan incelemeler.
- **Dinamik yöntemler:** Kod çalıştırılarak yapılan testler.

Doğrulama: Ürünü doğru geliştirdik mi?
Geçerleme: Doğru ürünü geliştirdik mi?

---

6. Yazılım Kalitesini Artırma Yolları

- Gereksinimlerin iyi analiz edilmesi
- Düzenli kod iyileştirme (refactoring)
- Sürekli regresyon testi
- Hata analizi
- Sürekli entegrasyon / sürekli teslimat
- Sağlıklı ekip ve süreç yönetimi

---

7. Yazılım Olgunluk ve Yetenek Modelleri

Yazılım süreçlerinin ne kadar olgun ve kontrollü olduğunu gösteren modeller:

CMM

5 seviye (1: Başlangıç → 5: Optimize).
Büyük firmalar için geliştirilmiş olgunluk modeli.

ISO 9000

Dışa yönelik kalite güvencesi sağlayan standartlar dizisi.

TRILLIUM

Telekom odaklı, yol haritası temelli olgunluk modeli (5 seviye).

SPICE (ISO 15504)

Detaylı süreç değerlendirme modeli.
6 seviye (0–5 arası).

TICKIT

ISO 9001'in yazılım uyarlaması.
Kalite sisteminin hem belgelerini hem uygulamasını denetler.

Sonuç

Yazılım kalitesi; süreç, ekip, standart, test ve yönetim birleşimidir.
Kalite sonradan eklenen bir şey değildir — **başından sona sürekli sağlanır**.

10.Ünite - Yazılım Kalite Metrikleri

Yazılım kalitesini ölçmek için kullanılan metriklerin ne olduğunu, nasıl sınıflandırıldığını ve ne işe yaradığını açıklar. Aşağıdaki özet sade ve kısa tutulmuştur.

---

1. Metrik Nedir?

Metrik, bir yazılımın veya geliştirme sürecinin **ölçülebilir özelliklerini sayısallaştırma yöntemidir**.
Ölçemedikçe iyileştirme yapılamaz; bu yüzden kalite için metrik zorunludur.

---

2. Yazılım Metriklerinin Amacı

- Yazılımın bakımını kolaylaştırmak
- Kaliteyi izlemek ve iyileştirmek
- Hataları erken tahmin etmek
- Proje maliyet ve çabasını daha doğru hesaplamak
- Daha iyi yönetim kararları sunmak

---

3. Metrik Türleri

Ürün Metrikleri

Yazılımın kendisini ölçer. Kod satırı, karmaşıklık, modüller arası bağımlılık gibi.

Süreç Metrikleri

Geliştirme sürecinin performansını ölçer. Gecikme, hata bulunma hızı, test kapsamı gibi.

Proje Metrikleri

Projenin ilerlemesini ve maliyetini ölçer. Efor, bütçe, zaman çizelgesi gibi.

---

4. Temel Kalite Özellikleri

Metrikler bu özellikleri ölçmek için kullanılır:

- Doğruluk
- Güvenilirlik
- Verimlilik
- Bütünlük
- Bakım yapılabilirlik
- Taşınabilirlik
- Test edilebilirlik

---

5. Yaygın Yazılım Metrikleri

Kod Satırı Metrikleri (LOC)

Kod boyutunu ölçer. Üretkenlik ve büyüklük tahminlerinde kullanılır ama tek başına kaliteyi göstermez.

McCabe Siklomatik Karmaşıklığı

Kodun karar noktalarını sayarak modülün ne kadar karmaşık olduğunu ölçer. Karmaşıklık arttıkça test ihtiyacı artar.

Halstead Metrikleri

Operatör ve operand sayılarına dayanarak kodun karmaşıklığını ve geliştirme çabasını tahmin eder.

Fonksiyon Noktası Analizi

Kod satırından bağımsız olarak yazılımın sunduğu fonksiyonların büyüklüğünü ölçer. Dilden bağımsızdır.

Hata Metrikleri

Kaç hata bulunduğu, hata yoğunluğu, hatanın bulunma süresi.
Kalite düzeyinin en net göstergelerinden biridir.

Test Metrikleri

Test kapsamı, test başarı oranı, test çalıştırma süresi.
Ürünün ne kadar güvenle teslim edilebileceğini gösterir.

---

6. Olgunluk Modelleri ve Metrikler

CMMI, SPICE gibi olgunluk modelleri; süreç geliştirme ve kaliteyi ölçmek için metrik kullanımını zorunlu kılar. Süreç geliştikçe metrikler daha doğru ve verimli kullanılır.

---

7. Metriklerin Doğru Kullanılması

- Tek metrik hiçbir zaman kaliteyi tam temsil etmez.
- Farklı metrikler birlikte kullanılmalıdır.
- Metrikler projeyi cezalandırmak için değil, iyileştirmek için kullanılmalıdır.
- Metriklerin düzenli olarak izlenmesi gerekir.

Sonuç

Yazılım kalite metrikleri; kaliteli ürün geliştirmek, süreci kontrol etmek ve hataları azaltmak için gereklidir. Kodun karmaşıklığı, hata oranı, test kapsamı gibi veriler projeyi daha öngörülebilir ve yönetilebilir hale getirir. Metrikler doğru kullanıldığında yazılım kalitesi doğal olarak yükselir.

11.Ünite - Kullanılabilirlik Kavramı ve Testi

Bir yazılımın kullanıcı tarafından ne kadar kolay ve verimli kullanılabildiğini ele alır. Aşağıdaki özet sade ve anlaşılır tutulmuştur.

1. Kullanılabilirlik Nedir?

Kullanılabilirlik, bir kullanıcının bir sistemi **etkili, verimli ve memnuniyetle** kullanabilme derecesidir.
Temel amaç, kullanıcıyı zorlamayan, öğrenmesi kolay bir sistem sunmaktır.

---

2. Kullanılabilirliğin Temel Özellikleri

- **Etkinlik:** Kullanıcı görevi doğru bir şekilde tamamlayabiliyor mu?
- **Verim:** Görev kısa sürede ve minimum çabayla yapılabiliyor mu?
- **Memnuniyet:** Kullanıcı deneyimi olumlu mu?

- **Öğrenilebilirlik:** Sistem kolay öğreniliyor mu?
- **Hataya dayanıklılık:** Hata yapıldığında sistem toparlanmayı kolaylaştırıyor mu?

---

3. Kullanılabilirlik ve Kullanıcı Deneyimi

Kullanılabilirlik daha teknik tarafı ölçer;
Kullanıcı deneyimi ise duygusal ve algısal tarafı kapsar.
İkisi birlikte ele alındığında ürün hem güçlü hem kullanılabilir olur.

---

4. Kullanılabilirliği Etkileyen Faktörler

- Kullanıcı arayüz tasarımı
- Düzen ve görsel hiyerarşi
- Geri bildirim mekanizmaları
- Tutarlılık
- Görev akışlarının netliği
- Hedef kullanıcı kitlesinin özellikleri

---

5. Kullanılabilirlik Testi

Kullanılabilirlik testi gerçek kullanıcılarla yapılır ve kullanıcıların sistemi nasıl kullandığını gözlemlemeye dayanır.

Kullanıcıdan genellikle şu bilgiler alınır:

- Bir görevi nasıl tamamladığı
- Ne kadar sürede yaptığı
- Nerede zorlandığı
- Hangi noktada hata yaptığı
- Genel memnuniyet düzeyi

---

6. Kullanılabilirlik Testinin Türleri

- **Laboratuvar testi:** Kontrollü ortamda gözlem yapılır.
- **Saha testi:** Kullanıcının doğal ortamında yapılır.
- **Uzaktan test:** Çevrimiçi gözlem veya kayıtlar üzerinden yapılır.
- **A/B testi:** İki tasarım karşılaştırılır.

---

7. Test Sürecinin Aşamaları

- Hedef kitlenin belirlenmesi
- Görevlerin hazırlanması
- Test ortamının oluşturulması
- Kullanıcı gözlemi
- Bulguların analizi ve raporlanması

Amaç, hataları bulmak değil, **kullanıcıyı zorlayan noktaları keşfetmek** ve tasarımı geliştirmektir.

---

8. Kullanılabilirliğin Önemi

Kullanılabilirliği yüksek bir sistem:

- Daha hızlı öğrenilir
- Daha az hata yapılır
- Daha memnun kullanıcılar üretir
- Müşteri desteği maliyetini düşürür
- Rekabet avantajı sağlar

Sonuç

Kullanılabilirlik; etkinlik, verim, memnuniyet ve öğrenilebilirlik gibi unsurların birleşimidir.
Kullanılabilirlik testi ise bu unsurların gerçek kullanıcılar üzerinden ölçülmesidir.
Doğru tasarlanmış testler, yazılımın daha anlaşılır, güvenilir ve kullanıcı dostu olmasını sağlar.

12.Ünite - Gözden Geçirme

Yazılımı çalıştırmadan yapılan statik test türü olan gözden geçirmenin ne olduğunu, nasıl yapıldığını ve neden önemli olduğunu açıklar. Aşağıdaki özet sade ve hızlı anlaşılır şekilde hazırlanmıştır.

1. Gözden Geçirme Nedir?

Yazılımı çalıştırmadan; gereksinim, tasarım, kod, test planı, kullanım kılavuzu gibi belgelerin**manuel olarak incelenmesidir**.
Amaç, hataları **erken aşamada** bulmak ve maliyetleri azaltmaktır.

---

2. Gözden Geçirme Süreci

Süreç altı adımdan oluşur:

- **Planlama:** Hedefler, roller, kapsam belirlenir.
- **Başlama:** Belgeler dağıtılır, süreç anlatılır.
- **Kişisel hazırlık:** Herkes belgeyi inceleyip olası hataları belirler.
- **Toplantı:** Hatalar tartışılır ve kayıt altına alınır.
- **Yeniden çalışma:** Yazar hataları düzeltir.
- **Takip:** Hataların gerçekten çözüldüğü kontrol edilir.

---

3. Rol ve Sorumluluklar

- **Yönetici:** Planlama, kaynak ayırma, kararlardan sorumlu.
- **Moderatör:** Süreci yönetir, toplantıları yürütür.
- **Yazar:** İncelenen belgenin sahibidir.
- **Gözden Geçiriciler:** Teknik bilgiye sahip, hata bulan kişiler.
- **Katip:** Bulunan hataları ve kararları kaydeder.

---

4. Başarılı Gözden Geçirme İçin Gerekenler

- Net hedefler ve iyi tanımlanmış süreç

- Doğru kişilerin seçilmesi
- Güncel kontrol listeleri
- Küçük parçalara ayrılmış belgeler
- Katılımcıların yeterli hazırlığı
- Yönetim desteği
- Güven ortamı ve yapıcı iletişim

Başarısızlığa yol açan durumlar: kötü planlama, yetersiz eğitim, yanlış ekip seçimi, eksik materyal ve toplantı disiplininin bozulması.

---

5. Gözden Geçirme Türleri

Dört temel tür vardır:

Gayri Resmî Gözden Geçirme

Hızlı ve dokümansızdır. Ekip arkadaşının kontrol etmesi gibi.
Çevik süreçlerde yaygın.

Üzerinden Geçme

Belgeyi birlikte okuyup tartışarak hataları bulma yöntemidir.

Teknik Gözden Geçirme

Teknik uzmanlar tarafından yapılır. Daha yapılandırılmıştır, yazar dışında uzmanlar yer alır.

Teftiş

En resmî yöntemdir. Roller, giriş/çıkış kriterleri ve kayıtlar zorunludur.
Hataları bulma oranı en yüksek olan gözden geçirme türüdür.

---

6. Gözden Geçirme Teknikleri

Bireysel inceleme sırasında kullanılan teknikler:

- **Kurgusuz gözden geçirme:** Serbest okuma, sorunları not etme.
- **Kontrol listesine dayalı:** Belirli sorular üzerinden sistematik inceleme.
- **Senaryo ve prova:** Belgeyi kullanım senaryoları üzerinden değerlendirme.
- **Role dayalı:** Belgeyi farklı kullanıcı rollerine göre inceleme.
- **Bakış açısına dayalı:** Farklı paydaşların (kullanıcı, testçi, yönetici) bakışıyla inceleme.

Sonuç

Gözden geçirme, hataları erken bulmak için kullanılan güçlü bir statik test yöntemidir.
Doğru roller, disiplinli süreç ve uygun teknikler kullanıldığında hem maliyeti düşürür hem kaliteyi yükseltir.

13.Ünite - Yazılım Test Sertifikasyonları

Yazılım testi alanında uzmanlaşmak isteyenler için uluslararası standartları belirleyen kuruluşları, sertifika türlerini ve Türkiye'deki faaliyetleri ele almaktadır.

1. Yazılım Test Sertifikasyonunun Önemi

Yazılım testi, günümüzde ayrı bir uzmanlık alanı haline gelmiştir. Artan popülerlik ve iş gücü talebi nedeniyle, işverenlerin yetkin adayları seçebilmesi için uluslararası geçerliliği olan sertifikalar önem kazanmıştır. Bu alandaki en önemli kuruluş **ISTQB**'dir.

---

2. ISTQB Nedir?

ISTQB (International Software Testing Qualifications Board - Uluslararası Yazılım Testi Yeterlilikler Kurulu):

- 2002 yılında Belçika'da kurulmuş, kar amacı gütmeyen uluslararası bir dernektir.
- Yazılım testi yeterliliklerinin belgelendirilmesinde dünya lideridir.
- Test uzmanlarının kariyerlerinde ilerlemelerini sağlamayı ve mesleğin değerini tanıtmayı amaçlar.
- Sınav sorularının içeriğini, kapsamını ve akreditasyon kurallarını belirler.

---

3. ISTQB Sertifikalarının Avantajları

Sertifikalar hem bireyler hem de şirketler için çeşitli faydalar sağlar:

- **Bireyler İçin:**
  - Uluslararası geçerliliği olan bir yetkinlik kanıtı sunar.
  - Test mesleği için "Ortak Dil" ve sözlük oluşturur.
  - Kariyer ilerlemesini destekler ve profesyonel güvenilirlik sağlar.
- **Şirketler İçin:**
  - Müşterilere güven vererek rekabet avantajı sağlar.
  - Test süreçlerini standartlaştırarak verimliliği artırır.
  - Danışmanlık şirketlerinin marka değerini yükseltir.

---

4. Sertifikasyon Seviyeleri ve Yapısı

ISTQB sertifikaları üç ana uzmanlık seviyesinde yapılandırılmıştır:

1. **Temel Seviye (Foundation Level):** Giriş seviyesidir. Test uzmanı, analist, mühendis veya yönetici olmak isteyen herkes için temel oluşturur. Diğer sertifikaları alabilmek için bu sertifikaya sahip olmak **ön koşuldur**.
2. **İleri Seviye (Advanced Level):** Kariyerinde ilerlemiş ve pratik deneyime sahip kişiler içindir (Test Yöneticisi, Test Analisti vb.).
3. **Uzman Seviyesi (Expert Level):** Konu hakkında derinlemesine uzmanlık gerektirir.

Ayrıca sertifikalar **Agile (Çevik)**, **Core (Çekirdek)** ve **Specialist (Uzmanlık)** olmak üzere farklı kategorilere ayrılır.

---

5. Türkiye'de Yapılan ISTQB Sınavları

Ülkemizde ISTQB müfredatının tamamı olmamakla birlikte, şu 5 temel sınav uygulanmaktadır:

- ISTQB Certified Tester Foundation Level (Temel Seviye)
- ISTQB Certified Tester Foundation - Agile Tester Extension (Çevik Test)
- Test Manager - Advanced Level (Test Yöneticisi)
- Test Analyst - Advanced Level (Test Analisti)
- Technical Test Analyst - Advanced Level (Teknik Test Analisti)

---

6. TTB (Turkish Testing Board)

TTB (Türk Test Kurulu):

- 2006 yılında ISTQB'ye bağlı olarak kurulmuştur.
- Türkiye'deki bilişim profesyonellerinin uluslararası standartlarda eğitilmesini ve sertifikalanmasını sağlar.
- Sınavları düzenler ve uluslararası içeriği Türkçeleştirir.
- Sektörü bilgilendirmek amacıyla konferanslar düzenler ve raporlar yayınlar.

---

7. Türkiye'deki Etkinlikler ve Raporlar

TTB tarafından yürütülen iki önemli çalışma bulunmaktadır:

- **TestIstanbul Konferansı:**
  - 2010 yılından beri her yıl düzenlenen uluslararası bir konferanstır.
  - Mobil test, test otomasyonu, DevOps, çevik test gibi her yıl farklı bir tema işlenir.
  - Binlerce katılımcı ve onlarca global firmanın katılımıyla gerçekleşir.
- **Türkiye Yazılım Kalite Raporu (TSQR):**
  - 2011 yılından beri her yıl yayınlanır.
  - Bilişim profesyonellerinin katılımıyla sektörün durumunu, hedeflerini ve trendlerini analiz eder.

Sonuç

Yazılım testinde profesyonelleşmek için ISTQB sertifikaları küresel bir standarttır. Türkiye'de TTB aracılığıyla yürütülen bu süreçler, hem bireysel yetkinliği belgelemek hem de sektörün kalite standartlarını yükseltmek için kritik bir rol oynamaktadır.

14.Ünite - Temel Seviye Yazılım Test Sertifikasyon Sınavı

ISTQB Temel Seviye sınavına hazırlık amacıyla, yazılım testinin temellerinden araç desteğine kadar olan tüm konuların genel bir tekrarını ve özetini içerir.

1. Yazılım Testinin Temelleri

Yazılım testi, sadece hataları bulmak değil, aynı zamanda gereksinimlerin karşılandığını doğrulamak ve ürün kalitesini artırmaktır.

- **Test ve Hata Ayıklama (Debugging) Farkı:** Test, yazılımdaki arızaları ortaya çıkarır; hata ayıklama ise bu arızaların kök nedenini bulup düzeltir.
- **Hata Zinciri:** İnsanın yaptığı bir **yanlışlık (error)**, kodda bir **kusura (defect/bug)** yol açar; bu kusur çalıştırıldığında bir **arıza (failure)** meydana gelir.
- **Test Süreci:** Test planlama, analiz, tasarım, uyarlama, koşum ve tamamlama aşamalarından oluşur.
- **Test Psikolojisi:** Test uzmanları, "kötü haber getiren kişi" olarak algılanabilir; bu nedenle iletişim yapıcı olmalıdır.

---

2. Yazılım Geliştirme Yaşam Döngüsü Boyunca Test

Test aktiviteleri, seçilen yazılım geliştirme modeline (Sıralı veya Döngüsel/Çevik) uygun olmalıdır.

- **Test Seviyeleri:** Birim, Entegrasyon, Sistem ve Kabul testleri gibi farklı seviyeler vardır.
- **Bakım Testleri:** Yazılım canlıya alındıktan sonra yapılan değişiklikler, platform taşımaları (migration) veya kullanımdan kaldırma (retirement) durumlarında tetiklenir.

- **Etki Analizi:** Yapılan bir değişikliğin sistemin diğer bölümlerinde yaratabileceği yan etkileri belirlemek için kullanılır.

---

3. Statik Testler

Yazılım kodunu çalıştırmadan yapılan incelemelerdir (gözden geçirme ve statik analiz).

- **Önemi:** Hataları (gereksinim, tasarım veya kodlama hataları) erken aşamada bulmayı sağlar.
- **Maliyet Etkisi:** Erken bulunan hataların düzeltilmesi, dinamik testlerde veya canlı ortamda bulunan hatalara göre çok daha az maliyetlidir.

---

4. Test Tasarım Teknikleri

Test teknikleri, test senaryolarını oluşturmak için kullanılan yöntemlerdir ve üç ana kategoriye ayrılır:

- **Kara Kutu Teknikleri (Davranışsal):** Yazılımın iç yapısına bakılmaksızın girdilere ve çıktılara odaklanır.
  - *Denklik Paylarına Ayırma:* Verileri benzer davranması beklenen sınıflara ayırır.
  - *Sınır Değer Analizi:* Hataların genellikle uç noktalarda olduğu varsayımına dayanır.
  - *Karar Tablosu Testi:* Karmaşık iş kurallarını ve kombinasyonları test eder.
  - *Durum Geçiş Testi:* Sistemin durumlar arasındaki geçişlerini inceler.
- **Beyaz Kutu Teknikleri (Yapısal):** Kodun iç mantığına odaklanır.
  - *Komut Kapsamı:* Çalıştırılabilir komutların test edilme oranıdır.
  - *Karar Kapsamı:* Karar noktalarının (doğru/yanlış çıktıları) test edilme oranıdır.
- **Tecrübeye Dayalı Teknikler:** Test uzmanının deneyimine dayanır.
  - *Hata Tahminleme:* Olası hataları öngörerek test tasarlamaktır.
  - *Keşif Testi:* Testlerin eş zamanlı tasarlanıp uygulandığı dinamik bir yaklaşımdır.

---

5. Test Yönetimi

Test süreci, uygun planlama, organizasyon ve risk yönetimi gerektirir.

- **Test Organizasyonu:** Bağımsız test ekipleri, testlerin etkinliğini artırabilir.
- **Roller:**
  - *Test Yöneticisi:* Planlama, izleme, kontrol ve stratejiden sorumludur.
  - *Test Uzmanı:* Test tasarımı, veri hazırlığı ve koşumundan sorumludur.
- **Riskler:**
  - *Ürün Riski:* Yazılımın beklenen fonksiyonu yerine getirememesi (kalite riski).
  - *Proje Riski:* Bütçe aşımı, personel sorunları veya tedarikçi problemleri.

---

6. Test Araçları

Test araçları, manuel aktiviteleri otomatize ederek verimliliği artırır.

- **Araç Kategorileri:** Test yönetim araçları, test tasarım araçları, test koşum araçları ve performans/izleme araçları olarak sınıflandırılır.
- **Araç Seçimi ve Pilot Proje:** Bir aracı kuruma yaymadan önce, güçlü ve zayıf yönlerini anlamak için pilot bir projede denenmesi esastır.

Sonuç

Bu ünite, ISTQB müfredatının bir özetidir. Yazılım testinin temel prensipleri, yaşam döngüsü içindeki yeri, statik ve dinamik teknikler, test yönetimi ve araç desteği konularını kapsar. Başarılı bir test süreci için doğru tekniklerin seçilmesi, risklerin yönetilmesi ve uygun araçların kullanılması kritik öneme sahiptir.