İSTANBUL MEDENİYET ÜNİVERSİTESİ

2025-2026 AKADEMİK YILI GÜZ DÖNEMİ

BİL403 YAZILIM MÜHENDİSLİĞİ

7. DERS

Dr. Öğr. Üyesi Ertürk ERDAĞI
erturk.erdagi@medeniyet.edu.tr

Yazılım Kalitesine Genel Bakış

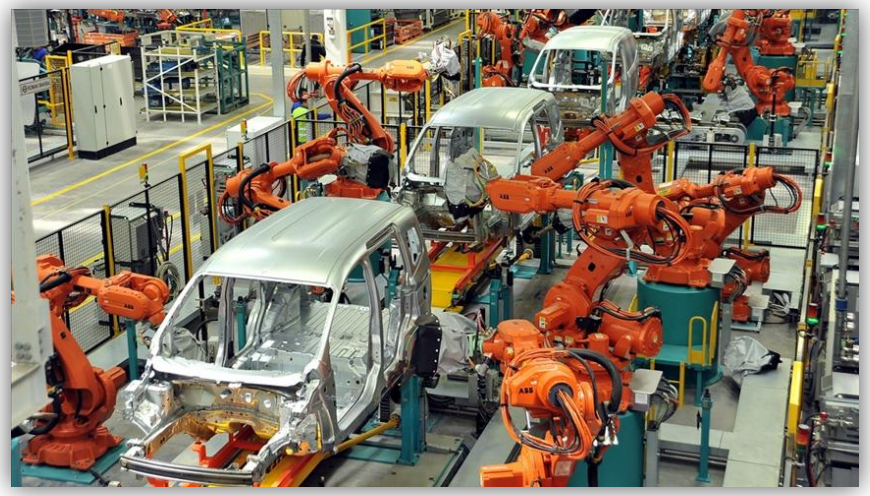
- Yazılım kalitesi, bir yazılım ürününün **belirlenen gereksinimleri ne ölçüde karşıladığı** ve **kullanıcı beklentilerini ne kadar tatmin ettiği** ile ilgilidir.
- Yani yazılım sadece "çalışmakla" değil, **doğru, güvenilir, verimli ve sürdürülebilir** biçimde çalışmakla kaliteli sayılır.
- **Tanım (IEEE 610.12):** "Yazılım kalitesi, bir yazılımın belirli gereksinimleri ve kullanıcı ihtiyaçlarını karşılayabilme derecesidir."
- Bir yazılımın kalitesi iki yönlü incelenir:

Boyut	Açıklama	Örnek
Ürün Kalitesi	Yazılımın teknik performansı ve işlevsel doğruluğu	Hız, güvenilirlik, hata oranı
Süreç Kalitesi	Yazılımın nasıl geliştirildiği	Test oranı, dokümantasyon, ekip işbirliği

Yazılım Kalitesine Genel Bakış

- Bir **mobil bankacılık uygulaması** düşünelim:
- Giriş ekranı sorunsuz çalışıyor ✅
- Ancak 10 saniyede açılıyor ❌
- Bu yazılım **fonksiyonel olarak doğru**, ama **performans açısından düşük kaliteli**. Dolayısıyla "çalışması" tek başına **kaliteli olduğu anlamına gelmez.**

- Bir otomotiv fabrikasında kalite, "her araç testten geçsin" demektir.
- Yazılımda kalite ise, "her yeni sürüm test edilmeden canlıya çıkmasın" anlamına gelir.



Kalite Kavramının Tarihsel Gelişimi

- **1. Üretim Odaklı Dönem (1950–1970)**
  - Kalite kavramı önce **imalat sektöründe** ortaya çıktı (özellikle otomotiv ve elektronik).
  - Amaç: **"Ürünü hatasız üretmek."**
  - Bu dönemde kalite ≈ *kontrol etmek* anlamına geliyordu.
- **2. Yazılımın Ortaya Çıkışı (1970–1990)**
  - Yazılım hataları maliyetli hale geldi ("software crisis").
  - İlk kalite modelleri (McCall, Boehm) bu dönemde geliştirildi.
  - "Test et, hata bul, düzelt" yaklaşımı benimsendi.
  - **Kalite güvence (Quality Assurance)** kavramı doğdu.
- **3. Süreç Odaklı Yaklaşım (1990–2000)**
  - ISO 9001, CMM ve SPICE standartları çıktı.
  - Kalite artık sadece ürünle değil, **süreçle** de ilişkilendirildi.
  - "Kaliteli süreç → kaliteli ürün" anlayışı gelişti.
- **4. Modern Dönem (2000–günümüz)**
  - Agile ve DevOps yaklaşımları kaliteyi **sürekli entegrasyon ve test** ile ilişkilendirdi.
  - Kalite: sadece test değil, **tüm yaşam döngüsüne** entegre bir kültür haline geldi.
  - Kullanıcı deneyimi (UX), güvenlik, sürdürülebilirlik gibi boyutlar öne çıktı.

Yazılım Kalitesine Genel Bakış

- Yazılımda kaliteyi en başta planlamak mı, yoksa sonunda kontrol etmek mi daha etkilidir?

Yazılım Kalitesine Genel Bakış

- Yazılımda kaliteyi en başta planlamak mı, yoksa sonunda kontrol etmek mi daha etkilidir?
  - Kalite en baştan planlanmalıdır; çünkü hatayı erken düzeltmek daha ucuzdur.

Kalite Neden Önemlidir?

- Kalite, yazılımın başarısında belirleyici rol oynar.
- Kötü kalite, sadece hatalı ürün değil; **maddi kayıp, güvenlik riski ve kullanıcı kaybı** demektir.
- **1) Maliyet Açısından**
  - Hataların erken tespiti → düşük maliyet
  - Hataların geç fark edilmesi → çok yüksek maliyet
- **Yazılım Mühendisliği Kanunu (Boehm, 1981):**
  - Hata ne kadar geç bulunursa düzeltme maliyeti o kadar kat artar.

Aşama	Hata Bulma Maliyeti
Analiz Aşaması	1x
Tasarım Aşaması	5x
Test Aşaması	15x
Kullanımda	100x

Kalite Neden Önemlidir?

- **2. Güvenlik Açısından**
- Zayıf yazılım kalitesi → **güvenlik açıkları**.
- Eksik doğrulama, hatalı kimlik doğrulama, veri sızıntısı gibi problemler doğurur.
- **3. Kullanıcı Memnuniyeti Açısından**
- Kullanıcılar, sadece işlev değil, **deneyim** bekler. Yavaş, kararsız veya karmaşık bir arayüz, teknik olarak doğru olsa da **kullanıcı açısından düşük kaliteli** kabul edilir.
- Örnek: Bir e-ticaret sitesinin 2 saniyeden uzun yüklenmesi, satışlarda %30 düşüşe neden olabilir (Google araştırması, 2022).

Yazılım Kalitesi ile Ürün Kalitesi Arasındaki Fark

- İyi bir süreç, iyi bir ürün doğurur. Kaliteli süreçler olmadan kalıcı ürün kalitesi sağlanamaz.

Özellik	**Yazılım Kalitesi**	**Ürün Kalitesi**
Odak	Geliştirme sürecinin nasıl yürütüldüğü	Nihai yazılım ürününün kendisi
Kapsam	Gereksinim analizi, tasarım, test, bakım süreçleri	Çalışma hızı, güvenilirlik, kullanıcı deneyimi
Kontrol	Süreç boyunca sürekli denetim	Ürün tamamlandıktan sonra test
Amaç	Kaliteli bir geliştirme süreci kurmak	Kullanıcıya kaliteli sonuç sunmak

- **Örnek:** Bir yazılım ekibi Agile yöntemle çalışıyor, kod gözden geçirme (code review) yapıyor, test otomasyonu kurmuşsa — bu ekip **yüksek yazılım kalitesi** süreci yürütüyor demektir.
- Sonuçta ortaya çıkan uygulama **ürün kalitesi** olarak da yüksek olacaktır.
- Soru : Kaliteli bir yazılım sürecinden düşük kaliteli ürün çıkar mı?

Yazılım Kalitesi ile Ürün Kalitesi Arasındaki Fark

- İyi bir süreç, iyi bir ürün doğurur. Kaliteli süreçler olmadan kalıcı ürün kalitesi sağlanamaz.

Özellik	Yazılım Kalitesi	Ürün Kalitesi
Odak	Geliştirme sürecinin nasıl yürütüldüğü	Nihai yazılım ürününün kendisi
Kapsam	Gereksinim analizi, tasarım, test, bakım süreçleri	Çalışma hızı, güvenilirlik, kullanıcı deneyimi
Kontrol	Süreç boyunca sürekli denetim	Ürün tamamlandıktan sonra test
Amaç	Kaliteli bir geliştirme süreci kurmak	Kullanıcıya kaliteli sonuç sunmak

- **Örnek:** Bir yazılım ekibi Agile yöntemle çalışıyor, kod gözden geçirme (code review) yapıyor, test otomasyonu kurmuşsa — bu ekip **yüksek yazılım kalitesi** süreci yürütüyor demektir.
- Sonuçta ortaya çıkan uygulama **ürün kalitesi** olarak da yüksek olacaktır.
- Soru : Kaliteli bir yazılım sürecinden düşük kaliteli ürün çıkar mı?
- Çok nadiren. Çünkü süreç kaliteliyse, ürün genellikle yüksek kaliteli olur. Ancak gereksinimler yanlış belirlenmişse, süreç iyi olsa bile ürün doğru sonucu vermez.

Kalite ve Yazılım Yaşam Döngüsü İlişkisi

- Yazılım kalitesi, yaşam döngüsünün **her aşamasına entegre edilmelidir.** Kalite sonradan "eklenemez"; süreç boyunca üretilir.

- Yazılım Yaşam Döngüsü Aşamaları ve Kalite İlişkisi

Aşama	Kalite Odak Noktası	Örnek
Gereksinim Analizi	Gereksinimler açık, ölçülebilir olmalı	"Sistem hızlı çalışsın" yerine "1 sn içinde yanıt versin"
Tasarım	Modüler ve okunabilir yapı	SOLID ilkelerine uygun tasarım
Kodlama	Kod standartlarına uyum	PEP8, naming conventions
Test	Otomatik test, hata tespiti	Unit test, integration test
Dağıtım	Doğru ortam konfigürasyonu	CI/CD, sürüm yönetimi
Bakım	Geri bildirimlerin izlenmesi	Kullanıcı raporları, hata takibi

- Sizce kalite hangi aşamada kontrol edilmeli — test aşamasında mı, yoksa en baştan mı?

Kalite ve Yazılım Yaşam Döngüsü İlişkisi

- Yazılım kalitesi, yaşam döngüsünün **her aşamasına entegre edilmelidir.** Kalite sonradan "eklenemez"; süreç boyunca üretilir.

- Yazılım Yaşam Döngüsü Aşamaları ve Kalite İlişkisi

Aşama	Kalite Odak Noktası	Örnek
Gereksinim Analizi	Gereksinimler açık, ölçülebilir olmalı	"Sistem hızlı çalışsın" yerine "1 sn içinde yanıt versin"
Tasarım	Modüler ve okunabilir yapı	SOLID ilkelerine uygun tasarım
Kodlama	Kod standartlarına uyum	PEP8, naming conventions
Test	Otomatik test, hata tespiti	Unit test, integration test
Dağıtım	Doğru ortam konfigürasyonu	CI/CD, sürüm yönetimi
Bakım	Geri bildirimlerin izlenmesi	Kullanıcı raporları, hata takibi

- Sizce kalite hangi aşamada kontrol edilmeli — test aşamasında mı, yoksa en baştan mı?
- Kalite en baştan oluşturulmalı; sadece testle kalite sağlanamaz.

Yazılım Kalitesinin Boyutları

- Yazılım kalitesi soyut bir kavramdır; “iyi” ya da “kötü” demek yetmez.
- Bu yüzden kaliteyi **ölçülebilir boyutlara ayırmak** gerekir.
- **ISO/IEC 25010 standardı**, yazılım ürün kalitesini **8 temel özellik** ve bunlara bağlı **alt özellikler** ile tanımlar.
- **1-Fonksiyonel Uygunluk (Functional Suitability)**
- Yazılımın, kullanıcı ihtiyaçlarını **doğru ve eksiksiz biçimde** karşılayabilme yeteneğidir.
- Fonksiyonel doğruluk (Correctness): İşlev doğru sonucu üretir.
- Fonksiyonel eksiksizlik (Completeness): Tüm gerekli işlevler mevcut.
- Fonksiyonel uygunluk (Appropriateness): Gereksiz işlev yok.
- **Örnek:** Bir e-fatura sistemi:
  - **Fatura kesebiliyor**
  - **Ancak dövizli işlem desteği yok**
  - Bu durumda **fonksiyonel olarak eksik** bir yazılımdır.

Yazılım Kalitesinin Boyutları

- **2-Güvenilirlik (Reliability)**
- Yazılımın, belirli koşullar altında **tutarlı ve kararlı biçimde çalışabilme** yeteneğidir.
- Olgunluk (Maturity): Hata oranı düşüktür.
- Kurtarılabilme (Recoverability): Hata sonrası toparlanabilir.
- Kullanılabilirlik Sürekliliği (Availability): Sistem kesintisiz çalışır.
- Örnek: Bir bankacılık sisteminde para transferi sırasında bağlantı koparsa işlem yarıda kalmamalıdır; sistem **otomatik kurtarma** mekanizmasıyla devam etmelidir.
- **3-Kullanılabilirlik (Usability)**
- Yazılımın, kullanıcılar tarafından **öğrenilmesi, anlaşılması ve etkin kullanılabilirliği**.
- Öğrenilebilirlik (Learnability)
- Anlaşılabilirlik (Understandability)
- Kullanıcı hatasına dayanıklılık (Error Tolerance)
- Örnek : Bir e-Devlet portalında kullanıcılar basit işlemi (örneğin “adres beyanı”) 5 adımda buluyorsa → düşük kullanılabilirlik. Mobil uygulamada tek dokunuşla ulaşabiliyorsa → yüksek kullanılabilirlik.

Yazılım Kalitesinin Boyutları

- **4-Verimlilik (Performance Efficiency)**
- Yazılımın kaynakları (zaman, bellek, işlem gücü) **ne kadar verimli kullandığı**.
- **Zaman davranışı (Time Behaviour):** Yanıt süresi.
- **Kaynak kullanımı (Resource Utilization):** CPU, bellek, ağ kullanımı.
- **Kapasite (Capacity):** Aynı anda desteklenen kullanıcı sayısı.
- Örnek : Bir online sınav sistemi 50 kullanıcıyı kaldırıyor ama 200 kullanıcıda çöküyorsa → **verimlilik problemi** vardır.
- **5-Bakım Yapılabilirlik (Maintainability)**
- Yazılımın, **kolayca güncellenebilmesi, hatalarının düzeltilebilmesi** ve geliştirilebilir olması.
- Analiz edilebilirlik (Analyzability)
- Değiştirilebilirlik (Modifiability)
- Test edilebilirlik (Testability)
- Örnek : Bir yazılımda her değişiklikte tüm sistemi yeniden derlemek gerekiyorsa → **bakımı zor**. Modüler yapıdaysa sadece etkilenen modül değiştirilir → **bakımı kolay.**

Yazılım Kalitesinin Boyutları

- **6-Taşınabilirlik (Portability)**
- Yazılımın farklı **ortamlarda veya platformlarda çalışabilme** yeteneği.
- Örnek : Bir masaüstü uygulaması Windows'ta çalışıyor ama macOS'ta çalışmıyorsa taşınabilir değildir. Web tabanlı veya platform bağımsız sistemler (ör. Java, Python, Flutter) → **yüksek taşınabilirlik.**
- **7-Uyumluluk (Compatibility)**
- Yazılımın **diğer sistemlerle veya bileşenlerle birlikte sorunsuz çalışabilmesi**.
- Birlikte çalışabilirlik (Interoperability)
- Eşzamanlı var olma (Co-Existence)
- Örnek : Bir hastane otomasyon sistemi laboratuvar modülüyle veri alışverişi yapamıyorsa → **uyumsuzluk** vardır. Aynı veritabanını paylaşabiliyorsa → **uyumlu sistem.**
- **8-Güvenlik (Security)**
- Yazılımın, **bilgiyi yetkisiz erişim, değişiklik veya kayıptan koruma** düzeyi.
- Kimlik doğrulama (Authentication), Yetkilendirme (Authorization), Veri bütünlüğü (Integrity), Gizlilik (Confidentiality), İzlenebilirlik (Accountability)

İç Kalite – Dış Kalite – Kullanım Kalitesi

- ISO/IEC 25010, sadece ürünün teknik yönünü değil, **kullanıcı deneyimi** boyutunu da dikkate alır.

Kalite Türü	Açıklama	Örnek
İç Kalite	Yazılımın iç yapısı, kod kalitesi, mimari tasarımı	Kod okunabilirliği, modülerlik, test kapsamı
Dış Kalite	Yazılımın çalışma anındaki davranışı	Yanıt süresi, hata oranı, bellek kullanımı
Kullanım Kalitesi	Kullanıcının yazılımla etkileşiminden doğan kalite algısı	Kullanılabilirlik, memnuniyet, hatasız işlem

- **Örnek:**
- Bir e-öğrenme platformu:
- **İç kalite:** Kodlar modüler, test oranı %80 : Olumlu
- **Dış kalite:** Zaman zaman yavaşlama var : Olumsuz
- **Kullanım kalitesi:** Öğrenciler memnun, çünkü arayüz basit : Olumlu
- Sonuç : İç kalite yüksek olsa da, dış kalite zayıfsa kullanıcı memnuniyeti düşer.

Yazılım Kalitesinin Değerlendirilmesi

- Bir yazılım projesinin **kaliteli olup olmadığını** anlamanın en doğru yolu onu **ölçmektir.**
- "Kaliteli" veya "iyi yazılmış" ifadeleri soyuttur; bu nedenle kaliteyi **sayısal ölçütlerle** değerlendirmek gerekir.
- Kalite ölçümü, yazılımın sadece ürün kalitesini değil, **geliştirme sürecinin olgunluğunu** da ortaya koyar.
- **Yazılım kalite ölçütleri (software metrics)**, bir yazılımın belirli bir niteliğini sayısal olarak ifade eden ölçümlerdir. Amaç kaliteyi **objektif** hale getirmektir.
- Hata Oranı (Defect Density)
- Belirli bir kod hacmi (ör. 1000 satır) başına düşen hata sayısı.
- $$Hata\ Yoğunluğu = \frac{\text{Toplam Hata Sayısı}}{\text{Kod Satırı}}$$
- Hata oranı azaldıkça yazılım kalitesi artar, ancak sıfır hata ≠ mükemmel yazılım (gizli hatalar olabilir).

Yazılım Kalitesinin Değerlendirilmesi

- Yanıt Süresi (Response Time)
- Bir sistemin isteğe verdiği yanıtın ortalama süresidir.
  Özellikle performans ve kullanıcı memnuniyeti açısından önemlidir.
- **Örnek:** Bir e-ticaret sitesinde:
- Sayfa yüklenme süresi 1,5 saniye, 5 saniyeyi geçerse kullanıcı memnuniyeti düşer.
- **İdeal:** Web uygulamaları: < 2 sn, Mobil uygulamalar: < 1 sn
- Kullanılabilirlik Testleri (Usability Testing)
- Gerçek kullanıcılar üzerinde yapılan testlerle **arayüzün kolaylığı, anlaşılırlığı ve memnuniyet düzeyi** ölçülür.
- **Yöntem:**
- Kullanıcıya görev verilir (“Bir hesap oluştur.”)
- Görev tamamlanma süresi, hata sayısı, memnuniyet skoru ölçülür.
- **Örnek:** Bir eğitim portalında 10 öğrenciden 8’i “ödev yükleme” adımını bulamıyorsa → kullanılabilirlik düşüktür.

Yazılım Kalitesini Ölçmek İçin Kullanılan Araçlar

- Yazılım kalitesi sadece gözlemle değil; **otomatik araçlarla** analiz edilerek değerlendirilir.
- Code Coverage (Kod Kapsamı) : Otomatik testler çalıştırıldığında **kodun ne kadarının test edildiğini** gösteren metriktir. Yani: "Yazdığımız testler, gerçekten tüm kodu kapsıyor mu?"
- 10.000 satır kodun 7.500 satırı test edilmiş → %75 **code coverage**
- Yüksek coverage önemlidir ama yeterli değildir.
- Tüm kod test edilmiş olabilir ama **yanlış testler** varsa kalite hâlâ düşüktür.
- SonarQube : Açık kaynaklı bir **kod kalite analizi ve ölçüm aracı**. Kodun okunabilirliğini, karmaşıklığını, güvenlik açıklarını ve teknik borç miktarını analiz eder.
- Değerlendirme Kategorileri: Bugs (Hatalar), Vulnerabilities (Güvenlik açıkları), Code Smells (Kötü kod pratikleri), Duplications (Kod tekrarları)
- **Örnek:**
- "10 code smells" → okunabilirlik problemi
- "5 vulnerabilities" → potansiyel güvenlik riski
- SonarQube Notlandırması: A → Mükemmel, B → İyi, C → Orta, D → Zayıf, E → Kritik

Kalite Güvence (QA) ve Kalite Kontrol (QC) Arasındaki Fark

- Bu iki kavram sıkça karıştırılır ama amaçları farklıdır:

Özellik	Quality Assurance (QA)	Quality Control (QC)
Odak Noktası	Süreç	Ürün
Zamanlama	Geliştirme sürecinde	Geliştirme sonunda
Amaç	Hataları önlemek	Hataları tespit etmek
Yöntem	Standartlar, süreç iyileştirme, dokümantasyon	Testler, incelemeler
Sorumlu	Tüm ekip	Test ekibi
Örnek	Kod inceleme (Code Review), ISO 9001 süreçleri	Birim testi, kullanıcı kabul testi

Yazılım Kalite Modelleri

- Yazılım kalitesini **ölçülebilir, karşılaştırılabilir ve yönetilebilir** hale getirmek için **modeller** geliştirilmiştir.
- **1-McCall Kalite Modeli (1977) :** Bu model, yazılım kalitesi konusundaki **ilk sistematik yaklaşımlardan biridir.** Odak noktası: Yazılımın **ürün nitelikleri**dir.

Ana Kategori	Açıklama	Örnek Faktörler
Ürün İşletilebilirliği (Product Operation)	Yazılım çalışırken gösterdiği davranış	Doğruluk, Güvenilirlik, Verimlilik, Kullanılabilirlik
Ürün Revizyonu (Product Revision)	Yazılımın bakım yapılabilirliği	Bakım yapılabilirlik, Test edilebilirlik, Esneklik
Ürün Geçişi (Product Transition)	Yazılımın farklı ortamlara uyumu	Taşınabilirlik, Yeniden kullanılabilirlik, Uyumluluk

- **2- Boehm Kalite Modeli (1978) :** McCall modelini temel alır, ancak kullanıcı ve geliştirici bakış açısını birleştirir. Amaç: Kalite faktörlerini **hiyerarşik olarak yapılandırmak.**

Modelin Yapısı

Seviye	Açıklama
Yüksek Seviye Faktörler (Primary)	Kullanıcı açısından kalite
Orta Seviye Faktörler (Intermediate)	Ürünün genel nitelikleri
Düşük Seviye Faktörler (Primitive)	Teknik ölçülebilir metrikler

Boehm'in Üç Ana Kalite Kategorisi

Ana Kategori	Açıklama	Örnek Faktörler
Ürün İşletilebilirliği (Product Operation)	Yazılım çalışırken gösterdiği davranış	Doğruluk, Güvenilirlik, Verimlilik, Kullanılabilirlik
Ürün Revizyonu (Product Revision)	Yazılımın bakım yapılabilirliği	Bakım yapılabilirlik, Test edilebilirlik, Esneklik
Ürün Geçişi (Product Transition)	Yazılımın farklı ortamlara uyumu	Taşınabilirlik, Yeniden kullanılabilirlik, Uyumluluk

Yazılım Kalite Modelleri

- **3-ISO/IEC 9126 Kalite Modeli (1991) :** ISO (Uluslararası Standardizasyon Örgütü) ve IEC (Elektronik Komisyonu) tarafından yayımlanan ilk uluslararası yazılım kalite standardıdır.

Özellik	Açıklama
Fonksiyonellik (Functionality)	Yazılımın işlevleri doğru yerine getirmesi
Güvenilirlik (Reliability)	Yazılımın hata oranı ve kararlılığı
Kullanılabilirlik (Usability)	Kullanıcı dostu arayüz ve öğrenilebilirlik
Verimlilik (Efficiency)	Performans ve kaynak kullanımı
Bakım yapılabilirlik (Maintainability)	Kolay güncellenebilirlik
Taşınabilirlik (Portability)	Farklı ortamlarda çalışabilme yeteneği

- **Örnek:** Bir e-öğrenme platformu:
  - Ders videoları düzgün açılıyor → **Fonksiyonellik**
  - Sistem çökmeden çalışıyor → **Güvenilirlik**
  - Menüleri basit ve net → **Kullanılabilirlik**

Yazılım Kalite Modelleri

- 4-ISO/IEC 25010 Kalite Modeli (Güncel) : **Modern standart model** olarak kabul edilir ve hâlen uluslararası ölçütlerde kullanılmaktadır.
- Ürün Kalitesi – 8 Ana Özellik

Özellik	Açıklama
Fonksiyonel Uygunluk	Gereksinimleri karşılama derecesi
Performans Verimliliği	Zaman, bellek ve kaynak kullanımı
Uyumluluk	Diğer sistemlerle entegrasyon
Kullanılabilirlik	Öğrenilebilirlik, erişilebilirlik
Güvenilirlik	Tutarlılık ve hata toleransı
Güvenlik	Veri gizliliği ve bütünlüğü
Bakım yapılabilirlik	Kolay değiştirme ve test etme
Taşınabilirlik	Farklı ortamlarda çalışabilme

- Kullanım Kalitesi – 5 Alt Ozellik

Özellik	Açıklama
Etkinlik (Effectiveness)	Kullanıcı amacına kolay ulaşır
Verimlilik (Efficiency)	Zaman/çaba dengesi
Memnuniyet (Satisfaction)	Kullanıcının genel tatmini
Özgürlük (Freedom from risk)	Kullanıcının zarar görmemesi
Kapsamlılık (Context coverage)	Farklı koşullarda kararlılık

Yazılım Standartları

- Yazılım projeleri büyüdükçe, farklı ekipler, şirketler ve ülkeler arasında **ortak bir dil** ve **kalite anlayışı** gerekliliği ortaya çıkar. İşte bu noktada **yazılım standartları**,
  - yazılımın **nasıl geliştirileceğini**,
  - **hangi süreçlerin izleneceğini**,
  - **nasıl ölçüleceğini ve değerlendirileceğini** belirleyen uluslararası rehberlerdir.
- Uluslararası yazılım standartları:
  - **ISO/IEC 12207** – Yazılım Yaşam Döngüsü Süreçleri
  - **ISO/IEC 25010** – Yazılım Ürün Kalitesi
  - **ISO/IEC 15504 (SPICE)** – Süreç iyileştirme
  - **CMMI (Capability Maturity Model Integration)** – Olgunluk modeli
  - **IEEE 829 / 830 / 1012** – Test ve gereksinim dokümantasyonu standartları
- **Örnek Durum:**
- Bir şirket, iki farklı ekip tarafından geliştirilen modülleri birleştiriyor. Her ekip farklı dokümantasyon formatı, test yöntemi ve kodlama stili kullanıyor. Sonuç: Entegrasyon karmaşası, hatalar, maliyet artışı.
- Eğer ekipler **ortak bir standart** (ör. ISO 12207 süreci) izleseydi,
  - roller,
  - teslimatlar
  - doğrulama adımları net olurdu.

Vize ile ilgili sorularınız ?

- Toplamda 10 soru olacaktır: Puanlama : 8-8-8-8-8-12-12-12-12-12 şeklinde olacaktır.
- Süreniz bir aksilik olmazsa 60 dakika olacaktır.
- **Konu başlıkları**
  - Yazılım Geliştirme Modelleri
  - Agile yöntemi
  - Kanban metodu
  - UML diyagramı
  - Yazılım mimarileri
  - Tasarım yaklaşımları
  - CI/CD süreçleri
  - Yazılım kalitesi
  - Gereksinim düzenlemesi

İSTANBUL MEDENİYET ÜNİVERSİTESİ

KATILIMINIZ İÇİN TEŞEKKÜR EDERİM.

YOKLAMA İÇİN İMZANIZI ATMAYI UNUTMAYINIZ.
CLASSROOM ÜZERİNDEN SINIFA DAHİL OLMAYI UNUTMAYINIZ

Sınıf Kodu : pua2tnoe

Dr. Öğr. Üyesi Ertürk ERDAĞI
erturk.erdagi@medeniyet.edu.tr