

Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ313 Yazılım Mühendisliği

2. Pratikte Yazılım Geliştirme

Dersin Genel Amacı



Bilgi ve Beceriler:

- Zaten bilgisayar bilgisine sahipsiniz.
- Programlama yapabiliyorsunuz.
- Çalışırken daha fazla şey öğreneceksiniz.

Gerçek Hayat Projeleri:

- Başarı yalnızca kod yazmaya bağlı değildir.
- Üretimdeki projelerdeki aksama milyonlarca dolara mal olabilir.
 - Instagram çözerse saatlik gelir kaybı ~1 milyon \$
 - Whatsapp çözerse iş anlaşmaları gecikir
- Kısa sürede proje sorumlusu olabilirsiniz ve çok büyük bütçeleri yönetiyor olabilirsiniz.



Ders Çıkarma:

- Hatalarınızı şimdi yapın.
- Hatalardan ders çıkarın.
- Bu gelecekteki büyük sorumluluklara hazırlıktır.

SORU?



Saatlik kar miktarı **100.000TL** olan bir işletme'nin **2** saat boyunca çökmesi durumunda yaklaşık müşteri güven kaybı **%20** olmuştur. Bu işletmenin ilgili ay içerisindeki zararını hesaplayınız.




 Veriler:

- Saatlik kâr: 100.000 TL
- Çökme süresi: 2 saat
- Müşteri güven kaybı: %20
- 1 ay \approx 30 gün = 720 saat





 Hesaplama:

1. Doğrudan kayıp = $100.000 \times 2 = 200.000$ TL
2. Aylık kâr = $100.000 \times 720 = 72.000.000$ TL
3. Güven kaybı zararı = $72.000.000 \times \%20 = 14.400.000$ TL

 Toplam Zarar = $200.000 + 14.400.000 = 14.600.000$ TL





◆ Kuruluşlar Yazılım Geliştiricilerine Neden Güvenir?

-  **Yetkinlik:**
 - Geliştirilen yazılımın kalitesi, doğrudan firmanın başarısını etkiler.
 - Etkisiz ya da hatalı yazılım, şirketi maddi ve itibari kayba uğratabilir.
-  **Gizlilik:**
 - Geliştiriciler ve sistem yöneticileri, **ticari sırlar, müşteri verileri ve kişisel bilgiler** gibi kritik verilere erişebilir.
 - Gizliliğin ihlali, **etik sorunlar** ve **hukuki yaptırımlar** doğurur.
-  **Yasal Ortam:**
 - Yazılım, fikri mülkiyet (copyright, lisanslar), veri koruma yasaları (KVKK, GDPR) gibi karmaşık yasal çerçeveler içinde geliştirilir.
 - Geliştiriciler, **lisans ihlallerine** ve **yasal sorumluluklara** dikkat etmelidir.
-  **Kabul Edilebilir Kullanım & Kötüye Kullanım:**
 - Yazılımın veya bilgisayar kaynaklarının kötüye kullanımı (ör. zararlı yazılım, internet solucanı) kuruluşu felce uğratabilir.
 - Geliştiriciler, **etik dışı kullanım** risklerini önceden öngörmeli ve gerekli tedbirleri almalıdır.



Akademik Dürüstlük ve Mesleki Uygulama

Yazılım Mühendisliğinde İşbirliği ve Etik

- Yazılım mühendisliği **işbirlikçi bir faaliyettir.**
- Birlikte çalışmak çok faydalıdır, ancak bazı görevler **bireysel çalışma** da gerektirir.
- Her zaman:
 -  Elinizdeki kaynaklara güvenin
 -  İşbirliği yaptığınız kişilere itibar edin

Doğru ve Profesyonel Eylem

- Başkalarının uzmanlığından yararlanmak
- Önceki çalışmalara dayanmak
- Uygun atıflarda bulunmak**
- Kullanılan yazılımların **lisans koşullarına uymak**
 - Örnl: MIT Lisansı → Serbest kullanım, ama atıf zorunluluğu
 - GPL Lisansı → Kod paylaşıldığında aynı lisansla açık tutulmalı
 - Apache Lisansı → Türev çalışmalarda lisans metnini koruma şartı

Etik Olmayan Davranış (Akademik İntihal)

- Başkalarının çalışmalarını **atıf yapmadan kullanmak**
- Yazılım lisanslarını ihlal etmek
 - Örnl: Kapalı kaynak bir kodu izinsiz paylaşmak
 - Açık kaynak kodunu alıp atıfsız “kendi ürünüm” gibi sunmak

Yazılım Pahalıdır

Yazılımın Maliyeti ve Başarı Ölçütleri

- Yazılım pahalıdır.
- En büyük maliyet genellikle geliştirme ekibinin maaşlarıdır.

Parayı Kim Ödüyor?

- Bu kişi veya kuruluşun beklentileri nelerdir?
- Onlar için başarı tanımı nedir?

Başarı / Başarısızlık

- Başarı: İhtiyaçları karşılayan, zamanında ve bütçeye uygun yazılım.
- Başarısızlık: Gereksinimleri karşılamayan, geciken veya bütçeyi aşan yazılım.

Farklı Bakış Açıları

- Teknik kişiler: Kod kalitesi, performans, teknik mükemmellik
- Yönetim / müşteri: Zaman, maliyet, işlevsellik, kullanıcı memnuniyeti

Çeşitlilik

Yazılım ürünleri çok çeşitlidir

- Uygulamalar:** Web siteleri, akıllı telefon uygulamaları
- Sistem Yazılımı:** İşletim sistemleri, derleyiciler
- İletişim:** Yönlendiriciler, telefon anahtarları
- Veri İşleme:** Telefon faturalandırması, emekli maaşları
- Gerçek Zamanlı:** Hava trafik kontrolü, otonom araçlar
- Gömülü Yazılım:** Aygıt sürücüler, denetleyiciler
- Mobil Cihazlar:** Dijital kamera, GPS, sensörler
- Bilgi Sistemleri:** Veritabanları, dijital kütüphaneler
- Ofis Uygulamaları:** Kelime işleme, video konferanslar
- Bilimsel:** Simülasyonlar, hava durumu tahmini
- Grafik ve Medya:** Film yapımı, tasarım



Çeşitlilik

Yazılım Geliştirme Sanatı

- Yazılım ürünleri **çok çeşitlidir**.
- Müşteri gereksinimleri **çok farklıdır**.
- Yazılım mühendisliği için **tek bir standart süreç yoktur**.
 - En iyi dil, işletim sistemi, platform, veritabanı ya da geliştirme ortamı **diye bir şey yoktur**.

Yazılım Geliştiricisinin Rolü

- Yetenekli bir yazılım geliştiricisi, **farklı yaklaşımlar, yöntemler ve araçlar** hakkında bilgi sahibidir.
- Yazılım geliştirme sanatı:
 - Her proje için **uygun yöntemleri seçmek**
 - Bu yöntemleri **etkili bir şekilde uygulamaktır**

Client (İstemci), Customer (Müşteri) ve User (Kullanıcılar)

Müşteri (Client)

- **Client (müşteri):** Yazılım geliştirme ekibinin **belirli ihtiyaçlarını karşılamak için yazılım geliştirdiği kişi veya kurumdur.**
- **Customer:** Ürünü **satın alan veya lisanslayan** taraftır. Bazen client ile aynı kişi olabilir, bazen farklı olabilir.
- **User (kullanıcı):** Yazılımı **fiilen kullanan** kişidir.

Örneklerle Ayrım

- **Bir banka için geliştirilen yazılımda:**
 - **Client:** Banka yönetimi (yazılımı geliştirenlerden talep eden taraf)
 - **Customer:** Banka (parayı ödeyen kurum)
 - **User:** Banka çalışanları ve müşterileri (yazılımı kullanan kişiler)
- **Genel amaçlı paket (ör. Microsoft Excel):**
 - **Client:** Microsoft'un hedef aldığı kurum/iş alanı
 - **Customer:** Excel lisansını satın alan kişi veya kuruluş
 - **User:** Excel'i kullanan herkes
- **Netflix:**
 - **Client:** Netflix
 - **Customer:** Ailen
 - **User:** Sen



Risk

Yazılım Projelerinde Karşılaşılan Zorluklar

- Birçok (hatta çoğu) yazılım geliştirme projesi çeşitli zorluklarla karşılaşır.

Yaygın Sorunlar

- Beklendiği gibi çalışmıyor (**işlev**) 
- Bütçeyi aştı (**Maliyet**) 
- Geç teslimat (**Zaman**) 

Rekabet Eden Hedefler

- Her yazılım projesinde **işlev**, **maliyet** ve **zaman** arasında bir denge vardır.
- Eklenen her yeni işlev → **geliştirme, test ve bakım maliyetini artırır.**
- Projeyi finanse eden kişi/kuruluş için en kritik soru:

“Benim için en önemli olan nedir?”

- Daha fazla işlev mi?
- Daha düşük maliyet mi?
- Daha kısa teslim süresi mi?

Risk



Risk

- Yazılım geliştirme projelerinin **başarısızlıkları şirketleri iflas ettirebilir.**
- Birçok durumda başarısızlıklar, **üst düzey yöneticilerin işlerine mal olur.**

Knight

Gerçek Dünya Örnekleri

•Apple Maps (2012):

- Lansmanda hatalar ve yanlış yönlendirmeler yüzünden kullanıcı güveni sarsıldı.
- Apple CEO'su Tim Cook, resmi özür yayımladı ve ilgili yönetici görevden alındı.

•Knight Capital (2012):

- Bir algoritma hatası 45 dakikada **440 milyon dolar** kayba yol açtı.
- Şirket iflasın eşiğine geldi ve kısa sürede satılmak zorunda kaldı.

•Heathrow Terminal 5 Açılışı (2008):

- Bagaj yönetim sistemindeki yazılım sorunları yüzünden binlerce bagaj kayboldu.
- Havayolu itibar kaybetti, milyonlarca pound zarar yazıldı.

•Denver Havalimanı Bagaj Sistemi (1990'lar):

- Otomatik bagaj sisteminde yıllarca süren yazılım sorunları yüzünden açılış ertelendi.
- Ekstra maliyet: **560 milyon dolar.**



Çıkarılacak Ders

- Yazılım projelerindeki riskler sadece teknik değil, **finansal ve yönetsel** boyutlarıyla da kritiktir.
- Bir hata, hem **şirketi hem de yöneticileri** doğrudan etkileyebilir.

Risk

Yazılım Projelerinde Başarı ve Başarısızlık

- Yazılım projelerinin önemli bir kısmı başarısız olur.
- Tahminlere göre projelerin **%50'si hiç kullanılmaz.**
- Çoğu zaman başarısızlığın nedeni:
 - Yazılım geliştiricilerin **yanlış yazılım** üretmesi.

Yazılım Ekibinin Yapması Gerekenler

- Müşterinin (Client)** yazılımdan ne beklediğini anlamak.
- Client'ın firması** açısından bu beklentilerin ne anlama geldiğini kavramak.
- Customer** (parayı ödeyen) ve **kullanıcıların** beklentilerini öğrenmek.

Unutulmaması Gereken Nokta

- Yazılım ekibi teknik öneriler ve içgörüler sağlayabilir.
- Ancak bir yazılım projesinde **başarının birincil ölçütü:**
Müşteri (Client) memnuniyetidir.

Riski En Aza İndirgeme: Müşteri (Client) ile İletişim

- **Fizibilite Çalışmaları:**

- Projeye başlanıp başlanmayacağına karar verme süreci.

- **Gereksinimler ve Tasarımın Ayrılması:**

- Gereksinimler: Müşterinin (client) ne istediği.
- Tasarım: Geliştiricilerin bu gereksinimleri nasıl karşılayacağı.

- **Kilometre Taşları ve Yayımlama (Release):**

- Geliştiricilerin ilerlemeyi müşterilere düzenli olarak bildirmesi veya göstermesi.

- **Kabul ve Kullanıcı Testi:**

- İstemcinin, yazılımın gereksinimleri karşıladığını test etmesi.
- Gerçek kullanıcılarla yapılan denemeler.

- **Devir Teslim:**

- Müşterinin, uzun süre çalıştırılabilen ve bakımı yapılabilen bir yazılım paketini teslim almasını sağlamak.



Riski En Aza İndirme: Görünürlük

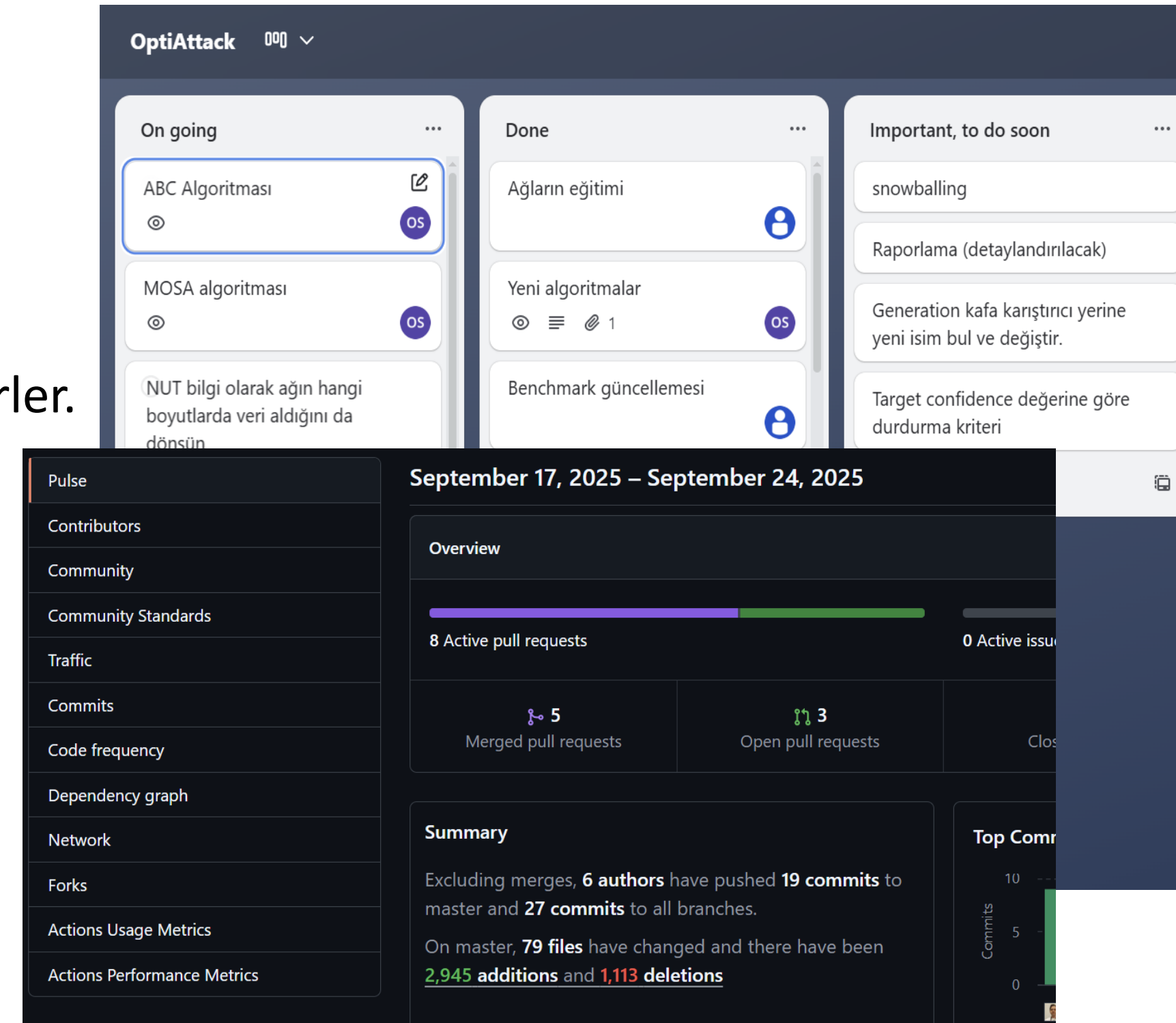
- Görünürlük:** Sorumluluk taşıyan kişiler, projede neler olup bittiğini bilmelidir.
- Bir problem, ancak **yönetici tarafından fark edilirse** çözülür.
- Yöneticiler, çeşitli zorluklar veya ilerleme durumları hakkında **ekipten gelen raporlara güvenmek zorundadır.**

Yazılım Geliştiriciler Açısından

- Süreci takip etmede zorlanabilirler.
- İlerleme konusunda genellikle **fazla iyimserdirler.**
- Raporlamayı bazen **boşa harcanan zaman** olarak görürler.

Çözüm

- Projeler için **düzenli ilerleme raporları** hazırlanmalıdır.
- En iyi görünürlük, **çalışan bir yazılım** ile sağlanır.



Riski En Aza İndirme: Kısa Geliştirme Döngüleri

- Risk, **çalışan yazılımların sık sık teslim edilmesi** ile en aza indirilebilir (aylar yerine haftalar).
- Bu sayede **client, customer ve kullanıcılar** geliştiricilerin çalışmalarını düzenli olarak değerlendirebilir.
- Kısa döngüler, **değişen koşullara uyum sağlama fırsatı** sunar.
- Bu yaklaşım, **çevik yazılım geliştirmenin temel ilkelerinden biridir.**

Dersin Teması: Ölçekleme

- Büyük ve daha büyük sistemlerin **farklı ihtiyaçları** vardır.
- Büyük bir proje **100 ila 10.000+ kişi-yılı** sürebilir.
- Her büyük sistem, **sürekli değişen birçok kişi** tarafından geliştirilir.
- Büyük projelerde, gereksinimler proje tamamlanmadan önce **birçok kez değişir**.
- Hiçbir büyük sistem aslında **tamamen bitmiş değildir**.
- Herkes, genellikle projenin yalnızca **bir kısmına hakimdir**.

BZ313 Kapsamındaki Projeler

- BZ313 projeleri yaklaşık **0.3 insan/yıl** büyüklüğündedir.
- Bu, bir üretim ortamındaki çevik sürecin **1–2 sprint** boyutuna denk gelir.



Kişi-Yıl Hesabı

•1 Kişi-Yıl = 1 kişi × 1 yıl = ~2000 saat

Gerçek Dünya Örnekleri

- WhatsApp (2014): 50 kişi × 5 yıl = 250 kişi-yıl
- Instagram (satış öncesi): 13 kişi × 2 yıl = 26 kişi-yıl
- Windows 10: 1000 kişi × 3 yıl = 3000 kişi-yıl



Ders Teması: Ekipler

- Çoğu yazılım geliştirme bir **ekip** tarafından gerçekleştirilir.
- Takımın etkinliği**, projenin başarısını belirleyen en önemli faktörlerden biridir.
- Çoğu büyük yazılım projesi, **önceki projeler üzerine inşa edilir**.
 - Yeni bir sistemi tamamen sıfırdan başlatmak oldukça nadirdir.
- Başkalarının çalışmalarını kullanarak ilerlemek, yazılım geliştirme süreçlerinde **temel bir beceridir**.
- Birçok yazılım **artırımlarla (increment)** geliştirilir.
 - Bu artımlardan farklı ekipler sorumlu olabilir.

Büyük Projelerin Yönetimi

- **Büyük çaplı projeler**, yalnızca teknik becerilerle değil, aynı zamanda güçlü bir **yönetim ekibi** ile başarılı olabilir.

Proje Yönetimi

- Projenin kapsamı, hedefleri ve zaman çizelgesinin belirlenmesi
- Risk yönetimi ve kriz durumlarına hazırlık
- İlerleme takibi ve raporlama

Personel Yönetimi

- Çok sayıda ekip üyesinin koordinasyonu
- Uzmanlık alanlarına göre görev dağılımı
- Motivasyon, iletişim ve ekip içi işbirliğinin sağlanması

Ekonomi, Hukuki Süreç ve Çevre Faktörleri

- Bütçenin planlanması ve maliyet kontrolü
- Sözleşmeler, lisanslar, fikri mülkiyet hakları
- Regülasyonlar ve yasal uyumluluk (örneğin, KVKK veya GDPR)
- Çevresel koşullar (farklı ülkelerde çalışma kültürü, zaman dilimi farkları, ekonomik dalgalanmalar)

Geliştirme Süreci

- Uygun yazılım geliştirme metodolojisinin seçilmesi (Agile, Waterfall, Spiral vb.)
- Gereksinim analizi → Tasarım → Kodlama → Test → Dağıtım aşamalarının yönetimi
- Kalite güvencesi ve bakım planlarının oluşturulması

Yazılım Geliştirme Süreçleri

Temelde;

Sürecin iyi yönetilmesi ile iyi yazılımlar ortaya çıkar

Sürecin iyi yönetilmesi ile riskler azalır

Sürecin iyi yönetilmesi ile görünürlük iyileşir

Sürecin iyi yönetilmesi takım çalışmasını olanaklı kılar

Çeşitli Yazılım Süreçleri

Çeşitli Yazılım Süreçleri

- Yazılım ürünleri **çok çeşitli** alanlarda kullanılır.
- Bu nedenle, **tüm yazılım geliştirme projeleri için tek bir standart süreç yoktur.**
- Ancak, başarılı yazılım projelerinin hepsi benzer sorunlarla karşılaşır ve bunları çözmek zorundadır.
- Bu nedenle her yazılım projesinde yer alması gereken **ortak adımlar** vardır.

Tüm Yazılım Geliştirme Süreçlerinde İşlem Adımları

Fizibilite ve planlama

Gereksinimler

Kullanıcı arayüzü tasarımı

Sistem tasarımı

Program geliştirme (tasarım ve kodlama)

Kabul ve yayınlama

İşletme ve bakım

Bu adımlar geliştirme döngüsü boyunca birçok kez tekrarlanabilir.

Bu adımlar arasında keskin bir ayrım yapmak ve herhangi bir anda ne yapacağınızı açıkça belirtmek önemlidir.

Not

- Yukarıdaki adımların tamamı **test**, **güvenlik** ve **performans** gibi süreçleri barındırır.



Fizibilite Çalışması

- Bir fizibilite çalışması, **projeye başlamadan önce** yapılır.

Yanıtlanması Gereken Sorular

- Önerilen projenin **kapsamı** nedir?
- Proje **teknik olarak uygulanabilir mi?**
- Öngörülen **faydalar** nelerdir?
- **Maliyetler** nedir, zaman çizelgesi nasıl olmalıdır?
- Kullanılabilir **mevcut kaynaklar** var mı?
- **Riskler** nelerdir ve nasıl yönetilebilir?

Sonuç

- Fizibilite çalışmasının ardından, **projeye devam edip etmeme** kararı verilir.
- Ayrıntılı olarak, **Ders 4'te** fizibilite konusuna değinilecektir.



Gereksinimler

- Gereksinimler, **sistemin işlevini müşterinin bakış açısından** tanımlar.
- **Client, Customer ve User** ile yapılan görüşmelerle:
 - Sistemin işlevselliği
 - Kısıtlamaları
 - Hedefleribelirlenir.

Gereksinim Süreci

Bu süreç bazen parçalara ayrılır:

- **Gereksinim Analizi** – İhtiyaçların toplanması
- **Gereksinim Tanımı** – Beklentilerin netleştirilmesi
- **Gereksinim Spesifikasyonu** – Teknik ve resmi dokümantasyonun hazırlanması

Kritik Nokta

- Gereksinimler üzerinde **anlaşmaya varılamaması** veya bunların **yeterince tanımlanmaması**, yazılım projelerinin başarısız olmasının **en büyük nedenlerinden biridir**.

İleri Konular

- **Ders 6, 7 ve 8** → Gereksinimler ayrıntılı olarak ele alınacaktır.



Kullanıcı Arayüzü Tasarımı

- **Kullanılabilirlik**, modern uygulama ve yazılım sistemlerinde büyük önem taşır.
- İyi bir kullanıcı arayüzü tasarımı ile sağlanabilir.
- Kullanıcı arayüzleri, **kullanıcı geri bildirimleri** ile değerlendirilir.

Değerlendirme Süreci (İteratif)

1. Kullanıcı arayüzünü tasarlama
2. Kullanıcılarla test etme
3. Arayüzü revize etme
4. Süreci tekrar etme

İleri Konular

- **Ders 9, 10 ve 11** → Kullanıcı Deneyimi ayrıntılı olarak işlenecektir.

Sistem Tasarımı

Tasarım

- Tasarım, sistemin **yazılım geliştiricisi bakış açısıyla tanımlanmasıdır**.
- Sistem Tasarımı:**
 - Hem donanım hem de yazılım bileşenlerini kapsar.
 - Gereksinimleri karşılayan bir **sistem mimarisi** oluşturmayı hedefler.
 - **Güvenlik** ve **performans**, tasarımın önemli parçalarıdır.
- Ders 12–15:** Sistem Tasarımı ayrıntılı olarak ele alınacaktır.

Model

- Modeller, genellikle:
 - Gereksinimleri
 - Sistem tasarımını
 - Program tasarımınıtemsil etmek için kullanılır.
- Bu ders kapsamında, **Birleştirilmiş Modelleme Dili (UML)**'nin temel kavramları öğretilacaktır.

Program Geliştirme

Program Geliştirme

- Sistem tasarımı ve kullanıcı arayüzü tasarımı harmanlanarak, **gereksinimleri karşılayan programlar** oluşturulur.

Program Geliştirmenin İki Boyutu

- Program Tasarımı
 - Uygulama (Implementation / Kodlama)
- Çoğu zaman bu iki süreç **eş zamanlı yürütülür** veya **birleştirilir**.

İleri Konular

- **Ders 16, 17 ve 18** → Program Geliştirme ayrıntılı olarak ele alınacaktır.

Kabul ve Yayınlama (Release)

Kabul Testi

- Sistem, **client tarafından sağlanan gereksinimlere göre** test edilir.
- Test süreci, seçilmiş **customer** ve **kullanıcılarla birlikte** yürütülür.

Teslimat ve Yayınlama

- **Başarılı bir kabul testinden sonra:**
 - Sistem, **client'a teslim** edilir.
 - Ardından **yayınlanır** veya **customer'lara pazarlanır**.

İleri Konular

- **Ders 19, 20 ve 21** → Kabul ve yayınlama ayrıntılı olarak ele alınacaktır.

İşletme (Operation) ve Bakım (Maintenance)

İşletim (Operation)

- Sistem, **pratik olarak kullanıma sunulur**.
- Kullanıcılar, sistemi gerçek iş süreçlerinde kullanmaya başlar.
- Performans, güvenlik ve kullanıcı deneyimi sürekli izlenir.

Bakım (Maintenance)

- Ortaya çıkan **hatalar (errors)** ve **sorunlar (problems)** tanımlanır.
- Bu hatalar giderilir ve sistem güncel tutulur.
- Bakım, yazılım yaşam döngüsünde en çok zaman ve maliyet alan aşamalardan biridir.

Gelişim (Evolution)

- Gereksinimler zamanla değiştikçe, sisteme **yeni işlevler eklenir**.
- Değişen teknik ortama (yeni donanım, işletim sistemi, standartlar) **uyum sağlamak için güncellemeler** yapılır.
- Yazılım, sürekli geliştirme ve iyileştirme sürecindedir.

Aşamalı Kullanımdan Kaldırma (Phase Out)

- Sistemin **hizmetten çekilmesi** sürecidir.
- Yerini yeni bir sistem alabilir veya tamamen sonlandırılabilir.
- Veri aktarımı, kullanıcıların yeni sisteme geçişi ve destek süreci bu aşamada planlanır.

👉 Bu bütün süreç, genel olarak **Yazılım Yaşam Döngüsü (Software Life Cycle)** olarak adlandırılır.

Tüm Yazılım Geliştirme Süreçlerinde Kalite Kontrol

- Gereksinimleri doğrulama
- Sistem ve program **tasarımının** doğrulanması
- Kullanılabilirlik testi
- Program testi
- Kabul testi
- Hata düzeltme ve bakım

Bu adımlardan bazıları sistemin ömrü boyunca birçok kez tekrarlanacaktır

Test Kategorileri

Test

“Test” terimi, farklı bağlamlarda kullanılabilir ve kolayca karıştırılabilir:

Kullanıcı Testi

- Kullanıcı arayüzünün sürümleri, **gerçek kullanıcılar** tarafından sınanır.
- Kullanıcı deneyimleri, **gereksinimlerde veya tasarımda değişikliklere** yol açabilir.

Program Testi

- Geliştirme ekibi tarafından yapılır.
- Amaç: Hataları bulmak, performans ve doğruluğu kontrol etmek.
- İki temel türü vardır:
 - **Birim Testi (Unit Test):** Bileşenlerin tek tek test edilmesi.
 - **Sistem Testi:** Bileşenlerin bir araya getirildiğinde bütün olarak test edilmesi.

Kabul Testi

- Client** tarafından yapılır.
- Sistemin son sürümü veya belirli parçaları, **gereksinimlere uygunluk** açısından test edilir.

İleri Konular

- Ders 19, 20 ve 21:** Güvenilirlik ve test ayrıntılı olarak işlenecektir.

Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ313 Yazılım Mühendisliği

2. Pratikte Yazılım Geliştirme

Ders Sonu