YAZILIM MÜHENDİSLİĞİ KAPSAMLI ÇALIŞMA NOTLARI

1. YAZILIM MÜHENDİSLİĞİ TEMEL KAVRAMLARI

Yazılım Nedir?

- **Tanımlar**:
  - Elektronik araçların birbirleriyle haberleşebilmesini sağlayan makina komutları
  - Elektronik cihazların belirli işleri yapmasını sağlayan programlar
  - Problemi çözmek için bilgisayar diliyle oluşturulmuş anlamlı ifadeler bütünü
  - Bir bilgisayarda donanıma hayat veren programlar, yordamlar ve belgelemelerin tümü
- **Yazılım Bileşenleri**:
  - Mantık
  - İnsan
  - Belge
  - Program
  - Veri
  - Algoritma

Yazılım Mühendisliği Nedir?

- Bilgisayar programlarının tasarımı, geliştirilmesi, test edilmesi ve bakımı konularını ele alan mühendislik dalı
- 1968 yılında NATO konferansında ortaya çıkan kavram
- IEEE ve ACM gibi mesleki kuruluşların etkisiyle gelişmiştir
- Diğer mühendislik dallarıyla karşılaştırıldığında çok yeni bir alan

Yazılım Mühendisliğinin Tarihsel Gelişimi

- 1950-1960: Sınırlı dağıtım, müşteri yazılımı
- 1960-1970: Veritabanları, çoklu kullanıcılı sistemler
- 1970-1980: Gerçek-zamanlı sistemler
- 1980-1990: Nesneye yönelik teknolojiler, uzman sistemler
- 1990-2000: Yapay sinir ağları, paralel işleme
- 2000+: Dağıtık sistemler, gömülü sistemler, düşük maliyetli çözümler

Yazılım Türleri

- **Sistem yazılımı**: İşletim sistemleri, sürücüler

- **Gerçek-zamanlı yazılım**: Kontrol sistemleri
- **İş yazılımı**: Muhasebe, insan kaynakları
- **Mühendislik ve bilimsel yazılım**: Analiz, simülasyon
- **Gömülü yazılım**: Cihaz kontrolü
- **Kişisel bilgisayar yazılımı**: Ofis uygulamaları
- **Yapay zeka yazılımı**: Öğrenme, karar verme

2. YAZILIM MÜHENDİSLİĞİ EĞİTİMİ

Yazılım Mühendisliği Mezunlarının Yetenekleri

1. Takım çalışması yapabilme
2. Gereksinimleri belirleme ve dönüştürme
3. Çelişen amaçları düzenleme, uzlaşma bulma
4. Etik, sosyal, yasal ve ekonomik faktörleri entegre etme
5. Teorileri, modelleri ve teknikleri anlama ve uygulama
6. Geliştirme ortamında etkin çalışma ve liderlik
7. Yeni modelleri ve teknolojileri öğrenebilme

Yazılım Mühendisliği Eğitiminin Bilgi Alanları

1. **Temeller**: Teorik ve bilimsel temeller
2. **Profesyonel Uygulama**: Teknik iletişim, etik ve sosyal sorumluluk
3. **Gereksinimler**: Sistem amacı ve kullanım alanı belirleme
4. **Tasarım**: Sistem gerçekleştirme stratejileri
5. **Yazılım Oluşturma**: Bileşenlerin geliştirilmesi
6. **Yazılım Sınama ve Doğrulama**: Statik ve dinamik test teknikleri
7. **Yazılım Gelişimi**: Sistemin bakımı ve desteği
8. **Yazılım Süreci**: Yaşam döngüsü modelleri ve süreç yönetimi
9. **Yazılım Kalitesi**: Ürün ve süreç kalitesi
10. **Yazılım Yönetimi**: Planlama, düzenleme ve izleme

Yazılım Mühendisliği Eğitiminin Temel Bileşenleri

1. Etkin İletişim ve Grup Çalışması Yetenekleri
2. Bir Uygulama Alanında Deneyim
3. Bir Ekip Projesi
4. Çalışma Ortamında Deneyim

5. Yaşam boyu Öğrenme Araçları
6. Bilgisayar Bilimi Temelleri
7. Yazılım Mühendisliği Temelleri
8. Mühendislik Uygulamaları ve Etiği

3. YAZILIM YAŞAM DÖNGÜSÜ VE SÜREÇ MODELLERİ

Yazılım Yaşam Döngüsü Temel Adımları

1. **Analiz**: Gereksinimlerin ayrıntılı çıkarılması, sorunların belirlenmesi
2. **Kaynak Planlama**: Personel ve donanım gereksinimleri, fizibilite çalışması
3. **Tasarım**: Yazılım sisteminin temel yapısının oluşturulması (mantıksal ve fiziksel)
4. **Gerçekleştirim**: Kodlama, test etme ve kurulum
5. **Bakım**: Hata giderme ve yeni eklentiler yapma

Yazılım Süreci Modelleri

1. **Gelişigüzel Model**
   - Herhangi bir model ya da yöntem yok
   - Geliştiren kişiye bağımlı
   - İzlenebilirliği ve bakımı zor
   - 60'lı yıllarda, tek kişilik üretim ortamı
2. **Barok Modeli**
   - Yaşam döngüsü adımlarının doğrusal gelişimi
   - 70'li yıllar
   - Belgelemeyi ayrı bir süreç olarak ele alır
   - Aşamalar arası geri dönüşler tanımlı değil
3. **Çağlayan (Şelale) Modeli**
   - Yazılım yaşam döngüsü adımları baştan sona izlenir
   - İyi tanımlı projeler için uygun
   - Geri dönüşler iyi tanımlanmış
   - Sorunları:
     - Yineleme gerektirir
     - Uzun teslim süresi
     - Yanlışların düzeltilme maliyeti yüksek
     - Ekip motivasyonu düşük
4. **V Modeli**

- Sol taraf üretim, sağ taraf sınama işlemleri
- Üç temel çıktı:
  - Kullanıcı Modeli: Kullanıcı ilişkileri ve kabul sınaması
  - Mimari Model: Sistem tasarımı ve sınama
  - Gerçekleştirim Modeli: Modüllerin kodlanması ve sınanması
- Kullanıcı katkısını artırır

5. **Helezonik (Spiral) Model**
   - Risk analizi odaklı
   - Dört aşama:
     1. Planlama: Amaçlar, bütünleştirme
     2. Risk Analizi: Seçenekler ve risklerin belirlenmesi
     3. Üretim: Ara ürünün üretilmesi
     4. Kullanıcı Değerlendirmesi: Sınama ve değerlendirme
   - Avantajları:
     - Kullanıcı katkısı yüksek
     - Yönetici izlemesi kolay
     - Erken kodlama ve sınama

6. **Evrimsel Model**
   - Coğrafik olarak geniş alanlara yönelik
   - Her aşamada tam işlevsel ürünler
   - Pilot uygulama-test et-güncelle-dağıt
   - Başarısı ilk evrimin başarısına bağlı
   - Sorunları: Değişiklik denetimi, konfigürasyon yönetimi

7. **Artırımsal Model**
   - Her sürüm öncekini kapsar, işlevler artar
   - Uzun projeler için uygun
   - Bir taraftan kullanım, diğer taraftan üretim

8. **Araştırma Tabanlı Model**
   - Belirsizlik üzerine çalışma
   - "Yap-at" prototipi
   - Sonuçlar belirgin değil
   - Sabit fiyat sözleşmelerine uygun değil

Yourdon Yapısal Sistem Tasarım Metodolojisi

- Çağlayan modelini temel alan kolay uygulanabilir model
- Aşamaları:
  - Planlama
  - Analiz
  - Analizden Tasarıma Geçiş
  - Tasarım

4. YAZILIM İSTERLERİ VE ÇÖZÜMLEMESİ

Yazılım İsterleri Belirtim Yöntemleri

1. **Girdi/çıktı belirtimi**: Girdilerin üreteceği tepkiler veya çıktıları almak için verilecek girdiler
2. **Belirleyici örnekler**: İyi seçilmiş örneklerle sistemin açıklanması
3. **Modeller**: Matematiksel veya çizgesel modeller ile kesin belirtim

İsterler Belgesinin Özellikleri

- Özel terimler tanımlanmalı
- Genel > ayrıntı sıralaması olmalı
- Çizelge ve resimler kullanılmalı
- İsterler çizelgelerle ilintilendirilmeli
- İsterler somut ve ölçülebilir olmalı
- Bölümler arası tutarlılık sağlanmalı
- Kullanıcı gerekleri ve başarım düzeyleri belirtilmeli
- İşlevsel isterler, veri tabanı gereksinimleri açıklanmalı
- Donanım ve yazılım kısıtları belirtilmeli
- Zaman, öncelik, yazılım mülkiyeti gibi kısıtlar belirtilmeli

Yazılım Niteliği Ölçütleri

1. **Doğruluk**: İsterlere uygunluk düzeyi
2. **Güvenilirlik**: Sonuçların doğruluğu ve duyarlılığı
3. **Verimlilik**: Bilgisayar kaynakları, zaman ve program uzunluğu
4. **Korunmuşluk**: Yetkisiz kullanıma ve zarara karşı koruma
5. **Kullanılabilirlik**: Öğrenim ve kullanım kolaylığı
6. **Bakım kolaylığı**: Hata nedenlerini bulma ve giderme kolaylığı
7. **Esneklik**: Değişiklik yapma kolaylığı
8. **Sınanabilirlik**: İsterlerin gerçekleştirildiğini test etme kolaylığı

9. **Taşınabilirlik**: Farklı ortamlara aktarılabilme
10. **Desteleyicilik**: Yeni uygulamalarda kullanılabilirlik
11. **Uyumluluk**: Başka yazılımlarla işbirliği yapabilme

5. BİLİŞİM ETİĞİ

Bilişim Etiği Temel İlkeleri

1. Bilgisayar başka insanlara zarar vermek için kullanılamaz
2. Başka insanların bilgisayar çalışmaları karıştırılamaz
3. Bilgisayar ortamında başka insanların dosyaları karıştırılamaz
4. Bilgisayar hırsızlık yapmak için kullanılamaz
5. Bilgisayar yalan bilgiyi yaymak için kullanılamaz
6. Bedeli ödenmeyen yazılım kopyalanamaz ve kullanılamaz
7. Başka insanların bilgisayar kaynakları izin almadan kullanılamaz
8. Başka insanların entelektüel bilgileri başkasına mal edilemez
9. Kişi yazdığı programın sosyal hayata etkilerini dikkate almalıdır
10. Bilgisayarı saygı duyulacak şeyler için kullanmalıyız

Mesleki Etik Kuralları

- Adil, dürüst ve güvenilir olmak
- Ahlaki değerler doğrultusunda hareket etmek
- Mesleki yasa, kural ve standartlara uymak
- Toplumda bilişim bilincinin oluşmasına katkıda bulunmak
- Yükümlülüklere ve sözleşmelere uymak
- Özel bilgilerin gizliliğine özen göstermek
- Müşterileri doğru bilgilendirmek
- Haksız rekabetten kaçınmak
- Tüketici haklarına saygılı davranmak
- Çalışanların haklarını korumak

İnternet Etiği

- Gerçek hayatta yapılmaması gerekenleri internette de yapmamak
- Farklı kültürlere saygı göstermek
- Yeni kullanıcılara anlayışlı davranmak
- Sataşmalara karşılık vermemek

- Özel hayata saygılı olmak
- Büyük harflerle yazı yazmamak (bağırmak anlamına gelir)
- Duyguları ifade eden sembolleri kullanmak: :-) ;-) :-(
- İnternet olanaklarını kötü amaçla kullanmamak
- Telif haklarına saygı göstermek
- Spam ve zincir mesajlardan kaçınmak

6. YAZILIM TASARIMI

Tasarım Aşamaları

1. **Veri Tasarımı**: Veri yapıları ve veri modellerinin oluşturulması
2. **Mimari Tasarım**: Yazılım birimlerinin yapısal parçaları ve ilişkileri
3. **Yordamsal Tasarım**: Yapısal birimlerin yordam ve fonksiyonlara dönüştürülmesi
4. **Arayüz Tasarımı**: İnsan-makine etkileşimi ve alt sistem arayüzleri

Tasarım İlkeleri

- **Soyutlama**: En az ayrıntı ile denetim ve anlaşılabilirliği artırmak
- **Bilgi Gizleme**: Modül iç yapılarını gizleyerek karmaşıklığı azaltmak
- **Bütünlük**: Tüm isterlerin eksiksiz karşılanması için denetim

İyi Bir Tasarımın Özellikleri

1. İsterler ile izlenebilirlik
2. Kod ve testler ile izlenebilirlik
3. Programlama dilinden bağımsızlık
4. Yüksek işlevsellik, başarım ve güvenilirlik
5. Hata kotarma yeteneği
6. Kolay öğrenme ve kullanım
7. Tekrar kullanılabilirlik
8. Ürün ailesine temel oluşturma
9. Kolay anlaşılabilirlik
10. Değiştirilebilirlik
11. Standartlara uygunluk
12. Diğer tasarımlarla birleştirilebilme

Tasarım Süreci

- **Ön Tasarım**: İsterlerin veri ve mimari tasarımına dönüştürülmesi

- **Ayrıntılı Tasarım**: Veri ve mimari tasarımın ayrıntılı veri yapıları ve algoritmalara dönüştürülmesi

Tasarım Gösterim Yöntemleri

- **Yapısal Programlama Gösterimi**: Metinsel anlatım
- **Tasarım Dilleri**: Anahtar sözcükler, veri tipleri, alt program tanımları
- **Grafiksel Gösterim**: Diyagramlar ve şemalar
- **UML**: Nesneye yönelik çözümleme ve tasarım için standart dil
- **Akış Diyagramları**:
  - **Statik**: Sınıf, nesne, bileşen, varlık-ilişki, yapı
  - **Dinamik**: Veri akış, etkileşim, durum, akış

7. ARAYÜZ TASARIMI

Kullanıcı Arayüzü Önemi

- İşin kalitesini artırır
- Kullanıcı tatminini yükseltir
- İşgücü verimliliğini artırır
- Sistem güvenliğini sağlar

Kullanılabilirlik Kriterleri

1. **İşlevsellik**: Görev gereksinimlerini karşılama
2. **Kontrol Edilebilirlik**: Kullanıcı kontrolüne olanak tanıma
3. **Esneklik**: Yapı, bilgi ve kullanıcı ihtiyaçlarına uygunluk
4. **Hata Yönetimi**: Hata önleme, azaltma ve giderme
5. **Kullanıcıya Uygunluk**: Fiziksel, zihinsel ve psikolojik özelliklere uyum
6. **Kendi Kendini Betimleme**: Geri besleme, kılavuzluk ve destek
7. **Tutarlılık**: Konum, biçim ve format tutarlılığı
8. **İş Yükü**: Fiziksel ve zihinsel iş yükünün optimizasyonu
9. **Öğrenilebilirlik**: Hızlı öğrenme ve hatırlama

Arayüz Türleri

- **İçsel Arayüzler**: Yazılımın kendi iç öğeleri arasındaki arayüzler
- **Dışsal Arayüzler**: Yazılımın dış dünya ile arayüzü
- **Bileşen Arayüzleri**: Bileşenler arası tanımlı arayüzler

Arayüz Tasarım İlkeleri

- **Anlaşılabilir Dil**: Teknik terimler ve kısaltmalardan kaçınma
- **Mantıksal Sıralama**: İşlemlerin mantıklı düzende sunulması
- **Hızlı Bilgi Girişi**: Kullanıcının verimli çalışmasını sağlama
- **Yeterli Bilgi**: Gerekli ve yeterli bilginin sunulması
- **Tekdüzelik**: Tutarlı tasarım
- **Fonksiyonellik**: Sistemin durumunu gösterme
- **Kılavuz Bilgi**: İşlem yapma konusunda rehberlik
- **Kolay Terk**: İşlemin kolay terkedilebilmesi
- **Hızlı Geri Bildirim**: İşlemlerin doğruluğu hakkında bilgi
- **Uyarılar**: Sesli ve görüntülü uyarılar
- **Standart Görünüm**: Yaygın standartlara uygunluk

Kullanıcı Arayüz Geliştirme Süreci

1. **Çözümleme**: Sistem amaç ve işlevlerinin belirlenmesi
2. **Tasarım**: Arayüz türleri, giriş/çıkış yöntemleri ve hata mesajlarının belirlenmesi
3. **Gerçekleştirim**: Tasarımın kodlanması, kütüphane oluşturulması
4. **Test**: İşlevsellik ve kullanılabilirlik testi

8. YAZILIM GERÇEKLEŞTİRİMİ (KODLAMA)

Programlama Dilleri ve Gelişimi

- **1. Nesil Diller**: Makine düzeyinde kodlama
- **2. Nesil Diller**: Fortran, COBOL gibi temel diller
- **3. Nesil Diller**: C, Pascal gibi modern ve yapısal diller
- **4. Nesil Diller**: SQL, C++, XML, UML
- **5. Nesil Diller**: Konuya yönelik, ardışık düşünmeden programlama

Dil Seçim Kriterleri

- Uygulama alanının özellikleri
- Donanım özellikleri
- Başarım gereksinimleri
- Veri yapıları özellikleri
- Personel bilgi düzeyi
- Derleyici ve geliştirme ortamı
- Bakım ortamı

Dillerin Özellikleri

- **Genel Özellikler**:
  - Tasarımda koda geçiş kolaylığı
  - Amaca uygunluk
  - Dilin etkinliği
  - Derleyici etkinliği
  - Taşınabilirlik
  - Geliştirme araçları
  - Bakım kolaylığı
  - Tip kontrolü
  - Denetim yapıları
- **Nesneye Yönelik Dil Özellikleri**:
  - Modülerlik
  - Otomatik bellek yönetimi
  - Sınıflar
  - Kalıtım
  - Çokeşlilik
- **Gerçek Zamanlı Dil Özellikleri**:
  - Kuvvetli tip kontrolü
  - Dinamik bellek yönetimi
  - Parametre geçirme teknikleri
  - Hata yakalama
  - Soyut veri tipleri
  - Zaman belirtimi
  - Okunabilirlik, taşınabilirlik

Kod Etkinliği

- **Etkinlik Arttırma Yöntemleri**:
  - Aritmetik işlem hazırlığı
  - Uygun uzunlukta yordamlar
  - Etkin veri yapıları
  - İşaretçi (pointer) kullanımı
  - Uygun veri tipi ve uzunluğu

- **Bellek Etkinliği**:
  - Dinamik bellek kullanımı
  - Uygun veri tipi ve boyutu
  - Gereksiz alanlardan kaçınma
- **Giriş/Çıkış Etkinliği**:
  - Minimum G/Ç isteği
  - Tamponlama
  - Veri doğrulama
  - Veri geçerliliği kontrolü

Temel Kodlama İlkeleri

1. **Soyutlama**: Tekrardan kaçınma
2. **Bilgi Gizleme**: Gerekli olanın saklanması
3. **Otomasyon**: Çalışma zamanı belirleme
4. **Çok Düzeyli Korunma**: Seviyeli önlem
5. **Etiketleme**: Anlam kazandırma
6. **Belirgin Arayüz**: Açık tanımlama
7. **Taşınabilirlik**: Donanım bağımsızlığı
8. **Güvenlik**: Veri ve erişim güvenliği
9. **Basitlik**: Karmaşıklıktan kaçınma
10. **Genel Yapı**: Metinsel biçim kuralları
11. **Sözdizimsel Tutarlılık**: Anlamsal benzerlikler
12. **Sıfır-bir-sonsuz**: Sabit sayıya göre tasarım yapmama

Modül Oluşturma

- İlişkili yordamları ve verileri bir dosyada toplama
- Aynı tip işlevlere sahip yordamları belirleme
- Anlaşılabilir arayüz oluşturma
- Veri yapısı ve değişkenleri başta belirtme
- Evrensel veri tipleri ve makrolar oluşturma

9. PROJE YÖNETİMİ TEKNİKLERİ

SWOT Analizi

- **Strengths (Güçlü Yönler)**: Avantajlar, kaynaklar, farklılıklar

- **Weaknesses (Zayıf Yönler)**: İyileştirilmesi gereken alanlar
- **Opportunities (Fırsatlar)**: Ekonomik, teknolojik, politik gelişmeler
- **Threats (Tehditler)**: Mevcut ve potansiyel engeller

SWOT Analizi Uygulama Adımları

1. Amaç belirleme
2. Toplantı planlama (yer, zaman, katılımcı)
3. SWOT toplantısı (güçlü/zayıf yönler, fırsatlar/tehditler)
4. Değerlendirme toplantısı
5. Strateji belirleme

Balık Kılçığı Tekniği (Ishikawa Diyagramı)

- Problem ile nedenleri arasındaki ilişkiyi gösteren grafik
- **Uygulama Adımları**:
  1. Problemin belirlenmesi
  2. Teknik kullanımı hakkında bilgi
  3. Nedenlerin üretilmesi
  4. Yapılandırılmış beyin fırtınası tekniği
- **Diyagram Oluşturma**:
  - Problem başlığını diyagramın baş kısmına yazma
  - Ana nedenleri omurgaya 45 derecelik açıyla yerleştirme
  - Alt nedenleri ana nedenlere bağlama

Zihin Haritalama

- Beyne yol gösteren, ilişkileri ve kavramları bir arada sunan teknik
- **Avantajları**:
  - Mantıksal ve görsel düzeni birlikte kullanma
  - Geniş alana genel bakış sağlama
  - Yaratıcı çözümler üretme
  - Az kağıt kullanımı
  - Beynin sağ ve sol bölümünü birlikte kullanma
- **Oluşturma**:
  - Ana konuyu resim/imge ile ifade etme
  - Dışarıya doğru dallar çizme
  - Her dalda akılda kalıcı anahtar kelimeler kullanma

- İlgili dalları ilişkilendirme

10. KURUMSAL KAYNAK PLANLAMASI (ERP)

ERP Temel Modülleri

- **Lojistik**:
  - Malzeme Yönetimi (Satınalma ve Stok)
  - Satış-Dağıtım
  - Üretim Planlama
  - Üretim Kontrol
  - Kalite Yönetimi
  - Bakım Yönetimi
- **Mali (Finans)**:
  - Mali Muhasebe
  - Maliyet Muhasebesi ve Kontrol
- **İnsan Kaynakları**:
  - Personel İdaresi
  - Bordro
  - Zaman Yönetimi
  - Personel Planlaması
  - İşe Yerleştirme
  - Eğitim ve Toplantı Yönetimi

Tedarik Zinciri Yönetimi

- Tedarikçiden müşteriye uzanan süreç
- Satınalma, üretim, dağıtım entegrasyonu
- Mali ve maliyet muhasebesi entegrasyonu
- İş akışı bilgisinin paylaşımı

ERP Gereksinimleri

- e-iş: İşletme modelinin parçası
- Tedarik zinciri optimizasyonu
- Müşteri İlişkileri Yönetimi (CRM)
- E-ticaret entegrasyonu
- Web tabanlı hizmetler

- Uygulama Servis Sağlayıcılığı (ASP)

11. MODERN YAZILIM MÜHENDİSLİĞİ TRENDLERİ

Yapay Zekâ ve Yazılım Tasarımı

- **Güncel Uygulamalar**:
  - Kullanıcı davranışı öğrenme
  - Kişiselleştirilmiş deneyimler
  - Tahmine dayalı özellikler
  - YZ tabanlı öneri motorları
  - Chatbot entegrasyonu
  - Ses tanıma sistemleri

Büyük Veri Analitiği

- **Yazılım Tasarımına Etkisi**:
  - Kullanıcı davranışlarının analizi
  - Veri odaklı tasarım yaklaşımları
  - Netflix, Spotify gibi platformların veri kullanımı
  - **Süreç**: Veri toplama > Analiz > Veri odaklı tasarım

Nesnelerin İnterneti (IoT)

- **Yazılım Tasarımına Etkisi**:
  - Fiziksel cihaz entegrasyonu
  - Dijital ve fiziksel etkileşim tasarımı
  - Akıllı ev, sağlık ve şehir uygulamaları
  - Sensör verileri ve gerçek zamanlı yanıtlar

Bulut Bilişim

- **Yazılım Tasarımına Etkisi**:
  - Esneklik, erişilebilirlik, ölçeklenebilirlik
  - Merkezi olmayan, dağıtık mimariler
  - SaaS modelleri ve entegre tasarım
  - Veri güvenliği odaklı tasarım

Mobil ve Çok Platformlu Tasarım

- Platform bağımsızlık

- Duyarlı tasarım
- Çoklu cihaz desteği
- Çok platformlu tasarım standartları

Artırılmış ve Sanal Gerçeklik

- Fiziksel ve sanal ortamlar arası etkileşim
- Çok boyutlu kullanıcı deneyimi
- Sürükleyici arayüzler

Siber Güvenlik

- Veri gizliliği
- Kullanıcı bilgilerinin korunması
- Yetkilendirme mekanizmaları
- Şifreleme

Blockchain

- Dağıtık sistem tasarımı
- Merkezi olmayan uygulamalar (dApp)
- Güvenilirlik ve şeffaflık odaklı tasarım

Güncel Yazılım Mimarileri

- **Microservices**: Uygulamayı küçük, bağımsız hizmetlere böler
- **Serverless**: Sunucu yönetimi gerektirmeyen, olay odaklı mimari
- **Containerization**: Docker ve Kubernetes ile izole konteynerler

CI/CD Süreçleri

- **Continuous Integration**: Kod değişikliklerinin sürekli entegrasyonu ve testi
- **Continuous Deployment**: Otomatik üretim ortamına dağıtım
- **Otomasyon**: Test, build ve deployment süreçlerinde

Test ve Yazılım Kalitesi

- **Test Driven Development (TDD)**: Önce test, sonra kod
- **Birim Testleri**: Küçük kod parçalarının testi
- **Entegrasyon Testleri**: Modüllerin birlikte çalışmasının testi