HEMŞİRELİKTE BİLİŞİM DERNEĞİ
NURSING INFORMATICS ASSOCIATION

YZ DESTEKLİ GEREKSİNİM ANALİZİ

Eğitim Notları

YAPAY ZEKA
VERİ ANALİTİĞİ
HEMŞİRELİK BİLİŞİMİ
İHTİYAÇ ODAKLI ÇÖZÜMLER

Dr. Nuran AYDIN
Hemşirelikte Bilişim Derneği
Yönetim Kurulu Başkanı

HEMŞİRELİKTE BİLİŞİM DERNEĞİ
NURSING INFORMATICS ASSOCIATION

İçindekiler

BÖLÜM-1

SAĞLIKTA YAPAY ZEKA DESTEKLİ GEREKSİNİM ANALİZİNE STRATEJİK BAKIŞ

1.1.Sağlıkta Dönüşümün Yeni Gerçeği

Sağlık sektörü, tarihindeki en büyük dijital dönüşüm sancılarını yaşamaktadır. Bu dönüşümün merkezinde sadece yazılımlar değil, bu yazılımları klinik gerçekliğe uyduracak olan "insan faktörü" yer almaktadır. Dr. Nuran Aydın tarafından tasarlanan bu kursun temel amacı, sağlık profesyonellerini (hemşireler, hekimler, teknisyenler) pasif birer teknoloji kullanıcısı olmaktan çıkarıp, teknolojik çözümlerin mimarı haline getirmektir.

Sağlık sektörü, dijitalleşmenin en karmaşık ve en kritik uygulama alanlarından biridir. Bu karmaşıklığın temel nedeni; insan hayatının doğrudan sistemlere bağlı olması, süreçlerin çok aktörlü yapısı ve hata toleransının son derece düşük olmasıdır.

Geleneksel sağlık sistemleri:

- İnsan odaklı fakat veri açısından dağınık
- Süreç bazlı fakat standardizasyonu zayıf
- Reaktif (olay sonrası müdahale) yapıdadır

Dijital dönüşüm ile birlikte bu yapı:

☞ **Veri odaklı**

☞ **Proaktif (erken uyarı)**

☞ **Karar destekli** bir modele evrilmektedir.

Bu dönüşümün merkezinde ise iki kritik kavram yer alır:

- **Yapay Zeka (Artificial Intelligence)**
- **Gereksinim Analizi (Requirement Analysis)**

1.2. Neden Gereksinim Analizi Sağlıkta Kritik?

Sağlık bilişimi projelerinin başarısızlık nedenleri incelendiğinde, en yaygın problemin teknik yetersizlik değil, **yanlış tanımlanmış gereksinimler** olduğu görülmektedir.

Kritik gerçek: Yanlış problem → doğru çözüm yoktur

Bir sistem:

- teknik olarak mükemmel olabilir
- kullanıcı dostu olabilir
- hızlı çalışabilir

Ama eğer **yanlış problemi çözüyorsa**, klinik olarak değersizdir.

Klinik örnek: Bir hastanede geliştirilen ilaç yönetim sistemi düşünelim:

Gereksinim hatası: "Hemşire ilaç girişini hızlı yapabilmeli"

Gerçek ihtiyaç: "Yanlış ilaç uygulaması engellenmeli"

Sonuç:

- Sistem hızlı çalışır ✓
- Ama hasta güvenliği artmaz ✗

1.3. Yapay Zeka: Karar Veren mi, Destekleyen mi?

Sağlıkta yapay zekaya dair en büyük yanlış anlaşılma:

☞ "Yapay zeka karar verir"

Gerçek: ☞ Yapay zeka **karar vermez, karar destekler**

Bu yaklaşım literatürde şu kavramla ifade edilir:

Augmented Intelligence (Artırılmış Zeka)

Bu modele göre:

- İnsan → klinik uzmanlık sağlar
- Yapay zeka → veri analizi ve öngörü sağlar

Klinik örnek:

Bir yoğun bakım hastasında:

- Yapay zeka: "Sepsis riski %82"
- Hekim: klinik değerlendirme yapar
- Karar: hekim tarafından verilir

1.4. Gereksinim Analizi: Teknik Süreç Değil, Klinik Düşünme Biçimi

Gereksinim analizi çoğu zaman yanlış şekilde algılanır:

❌ "Yazılım ekibi için liste hazırlamak"
❌ "İstek toplamak"

Oysa sağlıkta gereksinim analizi: Klinik problemi anlama ve modelleme sürecidir

Bu süreçte sorulması gereken sorular:

- Problem gerçekten ne?
- Bu problem ne sıklıkla oluşuyor?
- Klinik riski nedir?
- Mevcut süreç neden başarısız?
- İnsan hatası mı, sistem hatası mı?

1.4. Sistem Hataları vs İnsan Hataları

Sağlık sistemlerinde yapılan çalışmalar şunu gösterir: Hataların %80'den fazlası sistem kaynaklıdır.

Bu nedenle modern yaklaşım:

❌"Kimi suçlayalım?"
☞ "Sistemde ne yanlış?"

Örnek:

Bir hemşire yanlış ilaç verdiğinde:

Geleneksel yaklaşım:

- Dikkatsiz hemşire ❌

Modern yaklaşım:

- Barkod sistemi var mı?
- Uyarı mekanizması var mı?
- İş yükü ne durumda?
- Sistem hatayı engelleyebiliyor mu?

1.5. Dijital Sistemler Neden Başarısız Olur?

Sağlık projelerinde başarısızlık nedenleri:

1. Yanlış gereksinim
2. Kullanıcı dahil edilmemesi
3. Klinik gerçeklikten kopuk tasarım
4. Süreç analizi yapılmaması
5. Teknoloji odaklı yaklaşım

Kritik içgörü:

☞ Sağlıkta teknoloji projeleri **IT projesi değildir**

☞ Bunlar **klinik güvenlik projeleridir**

1.6. Gereksinim Analizi + Yapay Zeka = Yeni Paradigma

Geleneksel analiz:

- İnsan gözlemine dayanır
- Yavaş ve subjektiftir

YZ destekli analiz:

- Büyük veri kullanır
- Örüntüleri ortaya çıkarır
- Riskleri öngörür

Örnek kullanım:

- İlaç hatası tahmini
- Yoğun bakım risk skorlama
- Taburculuk gecikme analizi

BURADA DURUP DÜŞÜNMEMİZ GEREKEN SİSTEM BİZDEN NE BEKLİYOR SORUSUNA YANITTIR

Sistem gelecek 10 yıl ve sonrasında sağlık profesyonellerinden aşağıdaki yeterlilik ve yetkinlikleri beklemektedir. Kısa bir ifade ile mevcut diplomalar bize yetmeyecek. Bu durumda gerekli yeterlilik ve yetkinliklerimizi arttırma yolunda çaba sarfetmemiz gerekiyor. Bu nedenle şu an bu kurstasınız.

Gelecek 10 yılın yeni pozisyonlar:

1. **Klinik Yapay Zeka Entegrasyon Uzmanı (Clinical AI Integration Specialist)**
2. **Tıbbi Prompt Mühendisi (Medical Prompt Engineer)**
3. **Uzaktan Hasta İzleme (RPM) ve Veri Analisti**
4. **Genomik ve Hassas Tıp Hemşiresi (Genomic & Precision Medicine Nurse)**
5. **Sağlık Algoritması Denetçisi / Etik Uyum Hemşiresi**
6. **Dijital Terapötik Koçu (Digital Therapeutics Coach)**
7. **Robotik Cerrahi ve Bakım Koordinatörü**
8. **Sanal Gerçeklik (VR) Rehabilitasyon Hemşiresi**
9. **Yaşam Süresi ve Sağlıklı Yaşlanma Danışmanı (Longevity Nurse)**
10. **Klinik İş Akışı Mimarı (Clinical Workflow Architect)**

Dijital Çağ Yetkinlik Seti

1. Veri Okuryazarlığı
2. Algoritmik Düşünme
3. Prompt (İstem) Mühendisliği
4. Sistem ve Süreç Analizi
5. Epistemik Disiplin (Klinik Doğrulama)
6. Disiplinlerarası İletişim (Teknik Çevirmenlik)
7. Dijital Etik ve Mahremiyet Yönetimi
8. Sinyal/Gürültü Yönetimi
9. Hibrit (İnsan-Makine) Çalışma Yeteneği
10. Dijital Empati ve Liderlik

Bu kursta öğreneceksiniz:

- Gereksinim analizi nasıl yapılır
- Klinik süreç nasıl modellenir
- Yapay zeka nasıl entegre edilir
- Hatalar nasıl önlenir

1.7. Kritik Mesaj

☞ **Doğru analiz = hayat kurtarır**

☞ **Yanlış analiz = sistematik hata üretir**

BÖLÜM-2

YAPAY ZEKA NEDİR ?

KAVRAMDAN KLİNİK GERÇEKLİĞE

2.1. Yapay Zeka Nedir? (Sağlık Perspektifi)

Yapay zeka (YZ), verilerden öğrenerek tahmin, sınıflandırma ve öneri üreten sistemler bütünüdür. Ancak sağlık alanında bu tanım yeterli değildir.

Sağlıkta YZ: Veriyi klinik anlamlı bilgiye dönüştüren sistemdir

Temel fark:

- Veri → sayısal bilgi
- Bilgi → anlamlandırılmış veri
- Klinik bilgi → karar üretmeye uygun bilgi

YZ'nin görevi bu dönüşümü sağlamaktır.

2.2. Sağlıkta Yapay Zekanın Evrimi

Yapay zeka sağlıkta üç temel evrim aşamasından geçmiştir:

1 Kural Tabanlı Sistemler (Rule-Based)

- "Eğer X ise Y yap" mantığı
- Örnek: Ateş > 38 İSE→ uyarı ver

☞ Basit
❌ Esnek değildir

2 Makine Öğrenmesi (Machine Learning)

- Veriden öğrenir
- Örüntü bulur

☞ Daha güçlü
❌ Açıklanabilirlik düşük

3 Derin Öğrenme (Deep Learning)

- Büyük veri ile çalışır
- Görüntü, ses, metin analiz eder

☞ Yüksek doğruluk
❌ "Black box" problemi

2.3. Klinik Karar Destek Sistemleri (CDSS) ve YZ

Klinik karar destek sistemleri (CDSS), sağlık profesyonellerine doğru zamanda doğru bilgiyi sunarak karar süreçlerini destekler.

YZ ile güçlendirilmiş CDSS: Statik değil, öğrenen sistemdir

CDSS Temel Akışı

Veri Kaynakları
(EHR, lab, vital bulgular)
↓
Entegrasyon
↓
Kural Motoru + YZ
↓
Risk Analizi
↓
Uyarı / Öneri
↓
Kullanıcı (Hekim/Hemşire)
↓
Geri Bildirim

Kritik nokta:

☞ Uyarı sayısı arttıkça güven azalır

☞ Doğru uyarı → güven oluşturur

Bu probleme : **Alert Fatigue (Uyarı Yorgunluğu) denir.**

2.4. Yapay Zekanın Klinik Kullanım Alanları

1. Erken Uyarı Sistemleri

- Sepsis riski
- Kardiyak arrest tahmini

2. Tanı Destek Sistemleri

- Radyoloji görüntü analizi
- Patoloji sınıflandırma

-

3. Operasyonel Optimizasyon

- Yatak yönetimi
- Taburculuk planlama

4. Kişiselleştirilmiş Tıp

- Hasta bazlı tedavi önerileri

2.5. Dijital İkiz (Digital Twin) Kavramı

Dijital ikiz:

DİJİTAL İKİZ le çalışabilir misiniz ? -2

3. Geleceği Simüle Etme Gücü (Klinik Karar Destek)
- Simülasyon Tasarımı ve Uygulama
- İlaç Dinamiği Modelleme
- Klinik Risk Tahminleme
- Bilişsel Esneklik

4. Yaşam Döngüsü Boyunca Takip (Kestirimci Hemşirelik)
- Boylamsal Veri Okuma
- Kestirimci Bakım Planlaması
- Dijital Arşiv ve Transfer Yönetimi
- Önleyici Sağlık Koçluğu

5. Karar Destek Mekanizması (Etik ve Klinik Liderlik)
- Algoritmik Eleştiri ve Denetim
- Etik Muhakeme ve Savunuculuk
- Kanıt Süzgeci ve Literatür Entegrasyonu
- Şeffaf Teknoloji İletişimi

Sizden beş yıl sonra dijital ikiz ile çalışabiliyor olmanız olacak

☞ Bu : **Hastanın dijital ortamda simülasyon modelidir**

Bu model:

- Klinik veriler
- Genetik bilgi
- Yaşam tarzı ile beslenir.

Kullanım:

- Tedavi simülasyonu
- Risk tahmini
- Kişiselleştirilmiş bakım

2.6. Yapay Zekanın Sınırlılıkları

YZ güçlüdür ama kusursuz değildir.

Temel riskler:

1. Veri Kalitesi Problemi :"Garbage in, garbage out"

2. Bias (Önyargı) :

- Eğitim verisi dengesizse
- Sonuçlar hatalı olur

3. Açıklanabilirlik Sorunu: "Neden bu sonucu verdi?" bilinmez

4. Aşırı Güven: Kullanıcılar sorgulamayı bırakabilir

2.7. Klinik Güvenlik Perspektifi

Sağlıkta YZ sistemleri: **Tıbbi cihaz gibi değerlendirilmelidir**

Bu nedenle:

- Validasyon yapılmalı
- Klinik testlerden geçmeli
- Sürekli izlenmeli

Kritik ilke: "Önce zarar verme" (Primum non nocere)

2.8. Yapay Zeka ve İnsan İş Birliği

Sağlıkta en doğru model: **Human + AI = Better Care**

Rol dağılımı:

İnsan	Yapay Zeka
Klinik karar	Veri analizi
Etik değerlendirme	Örüntü bulma
Deneyim	Hız

2.9. Gereksinim Analizine Etkisi

YZ, gereksinim analizini dönüştürür:

Geleneksel:

- Gözlem
- Görüşme

YZ destekli:

- Log analizi
- Veri madenciliği
- Risk tahmini

Örnek:

YZ şunu söyleyebilir: “Gece vardiyasında ilaç hatası %35 artıyor”

Bu:

- Gereksinim üretir
- Sistem tasarımını değiştirir

2.11. Kritik Mesaj

☞ Yapay zeka güçlüdür ama tek başına yeterli değildir

☞ En iyi sonuç: **insan + sistem iş birliği**

BÖLÜM-3

GEREKSİNİM ANALİZİ: KLİNİK PROBLEMİ SİSTEME DÖNÜŞTÜRME SANATI

3.1. Gereksinim Analizi Nedir?

Gereksinim analizi klasik tanımıyla:

"Bir sistemin ne yapması gerektiğini tanımlama süreci"

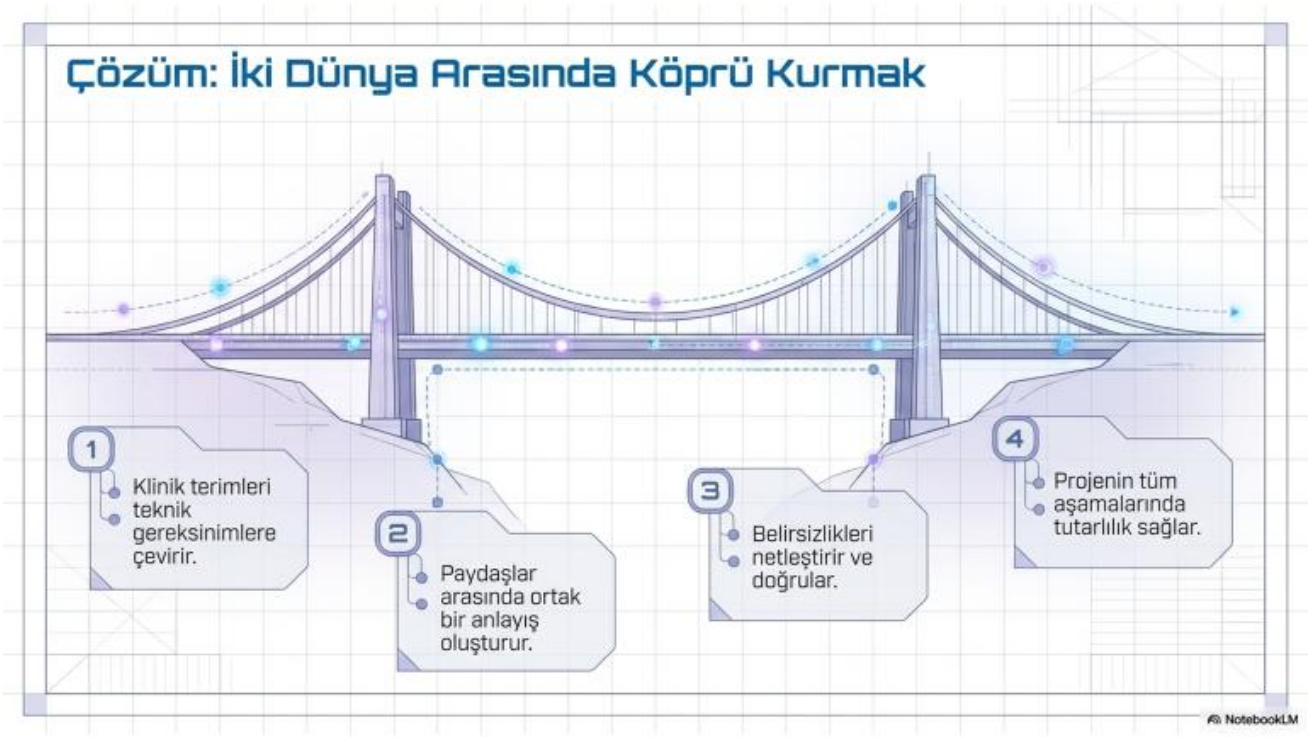
Ancak sağlıkta bu tanım yetersizdir.

Sağlıkta gerçek tanım:

☞ **Gereksinim analizi, klinik bir problemi güvenli ve ölçülebilir bir sisteme dönüştürme sürecidir.**

"Kayıp Tercüme" Problemi

Dijital projelerin %70'inden fazlasının başarısızlık sebebi teknik imkansızlıklar değil, **gereksinimlerin yanlış tanımlanmasıdır.** Sağlıkçıların dili "hasta bakımı, semptom, empati" odaklıyken; yazılımcıların dili "algoritma, veri tabanı, cache" odaklıdır. Bu iki dil arasındaki boşluğu dolduracak olan yetkinlik "Gereksinim Analizi"dir.



3.2. Neden Bu Kadar Kritik?

Sağlık projelerinde en büyük hata:

✕ Çözümle başlamak

☞ Problemle başlamak

Kritik kural: “Eğer problemi yanlış tanımlarsan, en iyi sistem bile hatalıdır.”

Gerçek saha örneği:

Problem algısı: “Hemşireler sisteme veri girmekte zorlanıyor”

Gerçek problem: “Veri girişi sırasında kritik kontroller yapılmıyor → hasta riski oluşuyor”

3.3. Gereksinim Türleri (Sağlık Özelinde)

Gereksinimler dört ana gruba ayrılır:

1 İş Gereksinimleri (Business Requirements) : Organizasyonun hedeflerini tanımlar

Örnek:

- Hasta güvenliğini artırmak
- İlaç hatalarını %30 azaltmak

2 Paydaş Gereksinimleri (Stakeholder Requirements): Kullanıcıların ihtiyaçları

Örnek:

- Hemşire hızlı işlem yapmalı
- Hekim kritik uyarıları görmeli

3 Fonksiyonel Gereksinimler (Functional): Sistem ne yapmalı?

Örnek: "Yanlış ilaç seçildiğinde sistem uyarı vermeli"

4 Non-Fonksiyonel Gereksinimler: Sistem nasıl olmalı?

Örnek:

- Performans
- Güvenlik
- Kullanılabilirlik

3.4. Klinik Gereksinim Nasıl Yazılır?

En kritik beceri: **doğru yazım**

❌ **Kötü gereksinim:** "Sistem kullanıcı dostu olmalı"

☞ Ölçülemez

✅ **İyi gereksinim:** "Hemşire ilaç girişini 3 adımda tamamlayabilmelidir"

☞ Ölçülebilir ✓

Altın kural: Gereksinim = **Net + Ölçülebilir + Test edilebilir**

3.5. Klinik Senaryo → Gereksinim Dönüşümü

Örnek: İlaç hatası

Durum: Yanlış ilaç uygulanıyor

Analiz:

- Ne zaman oluyor? → Yoğun saatlerde
- Neden? → Kontrol mekanizması yok
- Risk? → Yüksek

Gereksinim: "Sistem, ilaç uygulaması öncesinde hasta–ilaç eşleşmesini doğrulamalıdır."

3.6. Kullanıcı Öyküsü Yaklaşımı

Geleneksel gereksinim yerine modern yaklaşım:

Kullanıcı Öyküsü: "Bir [kullanıcı] olarak, [ihtiyaç], böylece [değer]"

Örnek:

"Bir hemşire olarak, yanlış ilaç vermemek için sistemin uyarı vermesini istiyorum."

Vaka Örneği: Bir hemşire "Hasta verilerine hızlı ulaşmak istiyorum" dediğinde, yazılımcı arka planda veri tabanını optimize eder. Ancak hemşirenin asıl ihtiyacı, acil bir durumda hastanın alerji bilgisini ana ekranda kırmızı görmektir. İşte bu farkı yakalamak analizin özüdür.

Avantaj:

- Kullanıcı odaklı
- Daha anlaşılır
- Agile uyumlu

3.7. Kabul Kriterleri

Her gereksinim test edilebilir olmalıdır.

Örnek:

Gereksinim: Sistem yanlış ilaçta uyarı verir

Kabul kriterleri:

- Yanlış ilaç seçildiğinde uyarı çıkmalı
- Uyarı bypass edilememeli
- Doğru ilaçta uyarı çıkmamalı

3.8. Gereksinim Hataları (En Yaygın)

1. Belirsizlik: "Hızlı olmalı"

2. Fazla genel ifade: "Sistem iyi çalışmalı"

3. Klinik bağlam eksikliği: Risk tanımlanmamış

4. Kullanıcı yok sayılması

3.9. Gereksinim Toplama Teknikleri

1. Gözlem (Shadowing): Gerçek süreç görülür

2. Mülakat: Derin bilgi alınır

3. Workshop: Paydaşlar birlikte çalışır

4. Doküman Analizi: Mevcut sistem incelenir

5. Veri Analizi (YZ destekli): Gerçek hatalar ortaya çıkar

3.10. Gereksinim → Süreç → Sistem Zinciri

Bu zincir kırılırsa proje başarısız olur: Gereksinim → Süreç → Tasarım → Sistem → Kullanım

Kritik gerçek: Gereksinim hatası → sistem hatası → hasta riski

3.11. Klinik Önceliklendirme

Tüm gereksinimler eşit değildir.

Öncelik kriterleri:

- Hasta riski
- Sıklık
- Etki

Örnek:

Gereksinim	Öncelik
İlaç kontrolü	□ Çok yüksek
Rapor tasarımı	□ Orta

3.12. Yapay Zeka ile Gereksinim Üretimi

YZ şu sorulara cevap verir:

- En çok hata nerede?
- Hangi süreç riskli?
- Ne zaman problem artıyor?

Örnek çıktı: "Gece vardiyasında hata oranı artıyor"

Gereksinim: Gece vardiyası için ek kontrol sistemi

3.14. Kritik Mesaj

☞ Gereksinim analizi = teknik iş değil
☞ Bu bir **klinik güvenlik disiplinidir**

BÖLÜM-4

GEREKSİNİM ve TASARIM

EN ÇOK YAPILAN HATA

Gereksinim ve Tasarım Arasındaki Fark Nedir ?

- Gereksinim ve tasarım süreçleri birbirine çok yakın görünse de, iş analizinde biri **"Problem/İhtiyaç"** alanına, diğeri **"Çözüm"** alanına odaklanır.

•**Gereksinim:** Klinik bir ihtiyacı veya kuralı tarif eder (Sözleşme gibidir).
•**Tasarım:** O kuralın teknik ve görsel çözümünü belirler (Mavi kopya/Proje çizimi gibidir).

Gereksinim vs. Tasarım Karşılaştırması görsel koy

4.1. Temel Ayrım (Net ve Tartışmasız)

Bu iki kavram sürekli karıştırılır:

Kavram	Anlam
Gereksinim	Sistem ne yapmalı?
Tasarım	Sistem bunu nasıl yapacak?

Altın kural:

☞ **Analiz aşamasında "nasıl" sorusu sorulmaz**

☞ Sadece "ne" sorusu sorulur

Gereksinim ve Tasarım Arasındaki Fark Nedir ?

Özellik	Gereksinim (Requirement)	Tasarım (Design)
Temel Soru	**Ne?** (Sistem ne yapmalı?)	**Nasıl?** (Sistem bunu nasıl yapacak?)
Odak Noktası	İş ihtiyacı, problem ve kullanıcı beklentisi.	Teknik çözüm, mimari ve kullanıcı arayüzü.
Sorumluluk	Analist, paydaşların neye ihtiyacı olduğunu belirler.	Analist ve teknik ekip, ihtiyacın nasıl karşılanacağını kurgular.
Çıktı Tipi	İş kuralları, kullanıcı hikayeleri, fonksiyonel listeler.	Ekran tasarımları (prototipler), veri modelleri, sistem mimarisi.
Hedef Kitle	İş birimi, yöneticiler ve son kullanıcılar.	Yazılım geliştiriciler, veri tabanı uzmanları ve grafik tasarımcılar.
Örnek (Sağlık)	"Hemşire, hastanın yaşamsal bulgularını sisteme kaydedebilmelidir."	"Ateş değeri için bir metin kutusu olacak, değer 38°C üzerindeyse sayı kırmızı görünecek."

Gereksinim ve Tasarım Arasındaki Fark Nedir ?

- Örnek 1: Akıllı İlaç Yönetim Sistemi

Özellik	Gereksinim (Ne?)	Tasarım (Nasıl?)
Klinik Senaryo	Yanlış ilaç uygulama hatasını önlemek.	Karekod destekli doğrulama arayüzü.
Odak Noktası	Hemşirenin ilacı hastaya vermeden önce sistemin "doğru hasta, doğru ilaç" kontrolü yapması ihtiyacı.	İlaç arabasına entegre bir el terminali mimarisi ve okutma işlemi sonrası ekranda çıkacak onay animasyonu.
Örnek Cümle	"Sistem, ilaç barkodu ile hasta bilekliği eşleşmediğinde işlem yapılmasını engellemelidir."	"Ekranda 400x400 piksel boyutunda kırmızı bir 'X' ikonu çıkmalı ve sesli uyarı (0.5 saniye) vermelidir."

Gereksinim ve Tasarım Arasındaki Fark Nedir ?

Örnek 2: Tele-Tıp (Uzaktan Muayene) Uygulaması

Özellik	Gereksinim (Ne?)	Tasarım (Nasıl?)
Klinik Senaryo	Hastanın evden doktorla görüşebilmesi.	Görüntülü görüşme platformu ve veri akışı.
Odak Noktası	Görüşmenin kesintisiz olması ve hastanın şikayetlerini yükleyebilmesi beklentisi.	Görüntü aktarımı için WebRTC protokolünün kullanılması ve 'dosya ekle' butonunun ana ekrana yerleşimi.
Örnek Cümle	"Hasta, geçmiş tahlil sonuçlarını doktorla görüşme sırasında paylaşabilmelidir."	"Sohbet penceresinin yanında bir 'Ataş' ikonu bulunmalı; bu ikon tıklandığında telefonun galerisini açmalıdır."

Gereksinim ve Tasarım Arasındaki Fark Nedir ?

- Örnek 4: Bası Yarası (Dekübütis) Risk Değerlendirmesi

Özellik	Gereksinim (Requirement)	Tasarım (Design)
Temel Soru	**Ne?**: Sistem riskli hastayı tespit etmeli.	**Nasıl?**: Braden skalası puanlaması ve uyarı mekanizması.
Odak Noktası	Hemşirenin hastadaki bası yarası riskini standart bir ölçekle puanlaması ihtiyacı.	Puanlama ekranının yerleşimi, toplam puanın otomatik hesaplanma algoritması.
Sorumluluk	Analist, hemşirelerin Braden skalasını hangi sıklıkla doldurması gerektiğini belirler.	Teknik ekip, puan 12'nin altına düştüğünde sistemin hangi birimlere bildirim göndereceğini kurgular.
Örnek Cümle	"Sistem, yüksek riskli (puan <12) hastalar için hemşireye otomatik olarak 'Pozisyon Verme Planı' oluşturmalıdır."	"Riskli puan oluştuğunda ekranın sağında 'Pozisyon Hatırlatıcı' widget'ı aktifleşmeli ve her 2 saatte bir yanıp sönmelidir."

4.2. Neden Bu Kadar Kritik?

Çünkü bu hata:

❌ Yanlış sistem
❌ Kullanıcı memnuniyetsizliği
❌ Klinik risk üretir.

Gerçek saha hatası:

Analist sorar: "Nasıl bir sistem istiyorsunuz?"

Kullanıcı der: "Buton sağda olsun, mavi olsun"

☞ Bu **tasarımdır**, gereksinim değil.

4.3. Klinik Örnek (Çok Kritik)

❌ **Yanlış (tasarım odaklı):** "İlaç ekranında kırmızı buton olmalı"

✅ **Doğru (gereksinim):** "Yanlış ilaç seçildiğinde kullanıcı uyarılmalıdır"

Fark:

- İlki çözümü dayatır ❌
- İkincisi problemi tanımlar □

4.4. "Çözümle Başlama" Hatası

Sağlık projelerinde en büyük hata: **Problemi anlamadan çözüm üretmek**

Örnek:

Problem anlaşılmadan:

- Mobil uygulama yapılır
- Dashboard yapılır

Ama kimse şunu sormaz:"Gerçek sorun ne?"

4.5. Gereksinimden Tasarıma Geçiş

Doğru süreç:

Klinik Problem
↓
Gereksinim (NE)
↓
Tasarım (NASIL)
↓
Geliştirme

Kritik nokta:

☞ Tasarım, gereksinimden sonra gelir

☞ Önce değil

4.6. Klinik Senaryo ile Açıklama

Durum: Hastanede ilaç hataları artıyor

❌ **Hatalı yaklaşım:** "Barkod sistemi kuralım"

☞ Çözüm varsayımı

✅**Doğru yaklaşım:**

1. Problem analizi yapılır
2. Süreç incelenir
3. Gereksinim yazılır

Gereksinim - Örnek: "İlaç uygulaması öncesinde doğrulama yapılmalıdır"

Tasarım seçenekleri:

- Barkod ✓
- **RFID ✓**
- Manuel çift kontrol ✓

☞ Görüldüğü gibi: **1 gereksinim → birçok tasarım**

4.7. Analistin En Büyük Tuzağı

Analist çoğu zaman: ☞ **Erken çözüm üretir**

Neden?

- Teknik bilgi fazla
- Kullanıcı baskısı
- "Hızlı çözüm" beklentisi

Sonuç:

❌ Yanlış ürün
❌ Tekrar iş (rework)
❌ Proje gecikmesi

4.8. Gereksinim Kalitesini Test Etme

Bir gereksinimi test etmek için şu soruyu sor: "Bu bir çözüm mü, yoksa problem tanımı mı?"

Örnek:

İfade	Tür
Barkod sistemi kurulmalı	❌Tasarım
İlaç doğrulama yapılmalı	□ Gereksinim

4.9. Klinik Risk Perspektifi

Yanlış ayrım şu riski doğurur:

☞ Sistem doğru çalışır

☞ Ama yanlış şeyi yapar

Bu en tehlikeli durumdur.

4.10. Gereksinim → Tasarım İzlenebilirliği (Traceability)

Profesyonel projelerde: Her tasarım kararı bir gereksinime bağlı olmalıdır

Örnek:

Gereksinim	Tasarım
İlaç doğrulama	Barkod sistemi

Bu neden önemli?

- Denetlenebilirlik
- Klinik güvenlik
- Regülasyon uyumu

4.11. Sağlıkta Özel Durum

Sağlıkta tasarım:

☞ Sadece lüks/ tercih değil

☞ **Hasta güvenliği aracıdır**

Örnek:

- Buton rengi → hata riskini etkiler
- Uyarı tasarımı → karar kalitesini etkiler

4.12. Yapay Zeka Perspektifi

YZ de aynı hataya düşebilir: Yanlış problem → yanlış model

Örnek:

- Model: hızlı veri girişi optimize eder
- Ama: hata riskini artırır

4.13. Kritik Mesaj

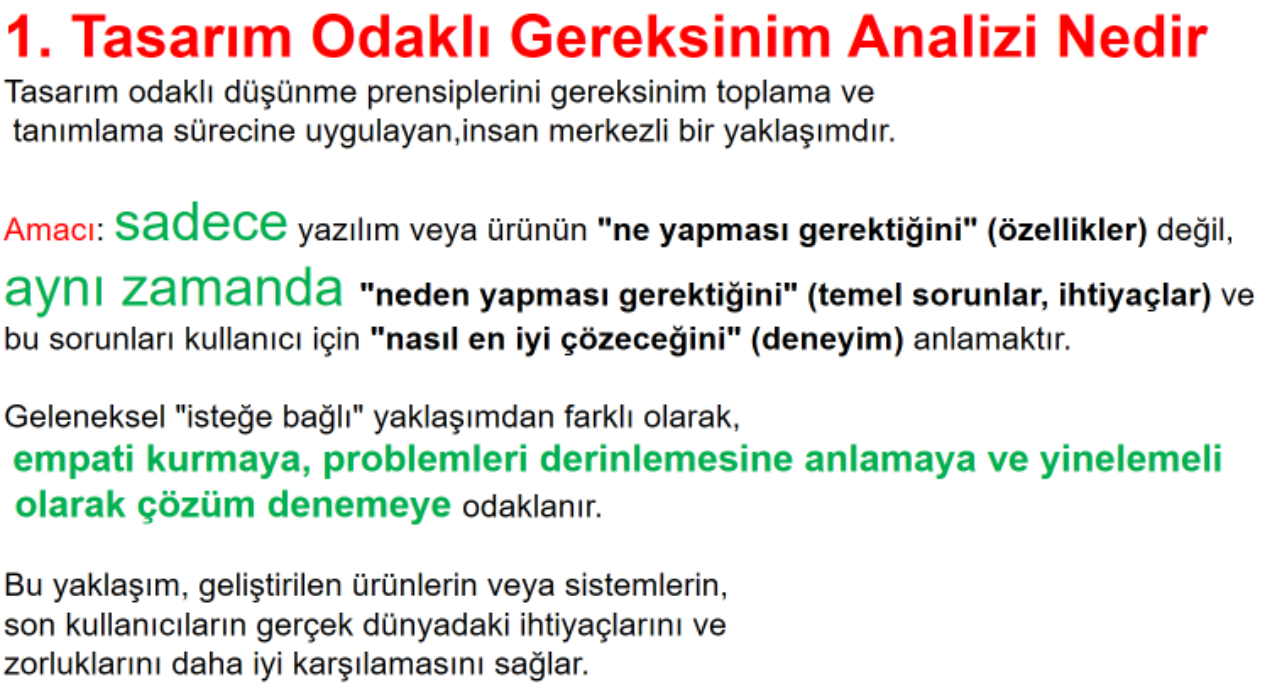
☞ **Problemi çöz, çözümü değil**

☞ Gereksinim doğruysa, tasarım seçenekleri çoğalır

BÖLÜM-5

TOGA: TASARIM ODAKLI GEREKSİNİM ANALİZİ (INSAN MERKEZLİ YAKLAŞIM)

5.1. TOGA Nedir?



TOGA (Tasarım Odaklı Gereksinim Analizi): **Kullanıcıyı merkeze alarak gereksinim üretme yaklaşımıdır**

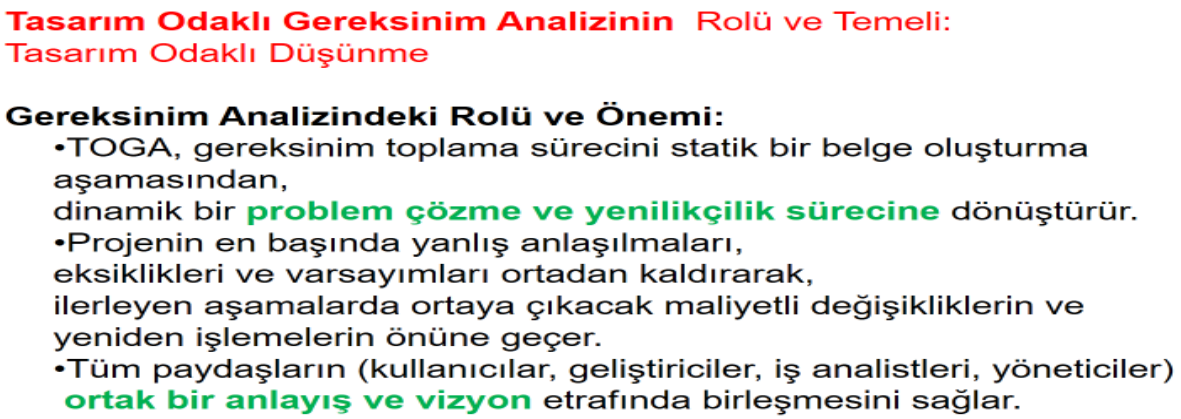
Bu model, klasik analizden farklı olarak:

- Sadece “ne gerekli?” sorusunu sormaz
- “kullanıcı bunu nasıl deneyimliyor?” sorusunu da sorar

Kritik fark:

Klasik Analiz	TOGA
Süreç odaklı	İnsan odaklı
Sistem bakışı	Deneyim bakışı
Teknik yaklaşım	Empati yaklaşımı

5.2. Neden TOGA Sağlıkta Kritik?



Sağlık sistemleri:

- Karmaşık
- Çok paydaşlı
- Yüksek riskli

En önemli gerçek: Sağlık hataları çoğunlukla **kötü tasarlanmış sistemlerden** kaynaklanır

Bu nedenle: TOGA = **Hata önleyici analiz yaklaşımı**

5.3. TOGA'nın 5 Aşaması

TOGA, 5 temel adımdan oluşur:

1. Empati : Kullanıcıyı anlamak

Nasıl yapılır?

- Gözlem (shadowing)
- Mülakat
- Günlük iş akışını izleme

Amaç: Kullanıcının yaşadığı gerçek problemi görmek

Klinik örnek:

Hemşire:

- 10 hastaya bakıyor
- Sürekli bölünüyor
- Sisteme veri girişi zor

☞ Problem: sistem değil, **iş yükü + süreç tasarımı**

2. Tanımlama

Problemi netleştirme

Yanlış: "Sistem kötü"

Doğru: "Yoğun bakımda ilaç uygulama sırasında doğrulama yapılmıyor"

Kritik çıktı: Problem cümlesi

3. Fikir Üretme

Çözüm alternatifleri oluşturma

Kurallar:

- Eleştiri yok
- Çok seçenek üret
- Yaratıcı düşün

Örnek çözümler:

- Barkod sistemi
- Sesli uyarı
- AI destekli kontrol
- Çift hemşire onayı

☞ Ama dikkat: Bu aşama hâlâ **tasarım değil, keşif aşamasıdır**

4. Prototip

Çözümü somutlaştırma

Örnek:

- Wireframe
- Mockup
- Basit demo

Amaç: Kullanıcıya göstermek

5. Test

Gerçek kullanıcı ile doğrulama

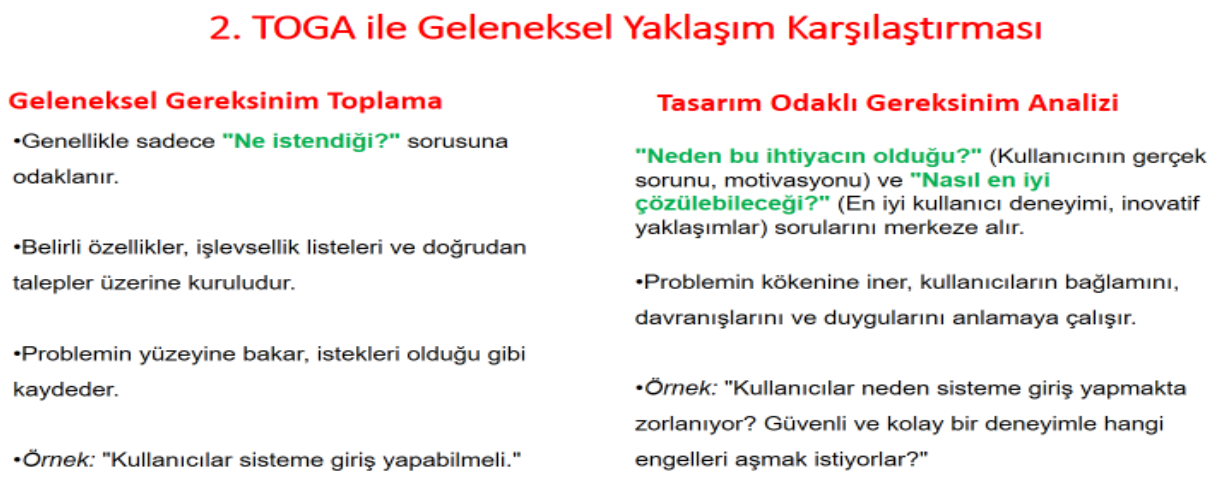
Sorular:

- Kullanıcı anlayabiliyor mu?
- Hata yapıyor mu?
- Süreç hızlandı mı?

Kritik: Test edilmemiş sistem = risk

5.4. TOGA Süreci (Bütünsel Akış)

Empati → Tanımlama → Fikir → Prototip → Test



5.5. Klinik Senaryo ile TOGA

Problem: İlaç hataları artıyor

1. Empati:

- Hemşire gözlemlendi
- Süreç karmaşık

2. Tanımlama: "İlaç doğrulama süreci yetersiz"

3. Fikir:

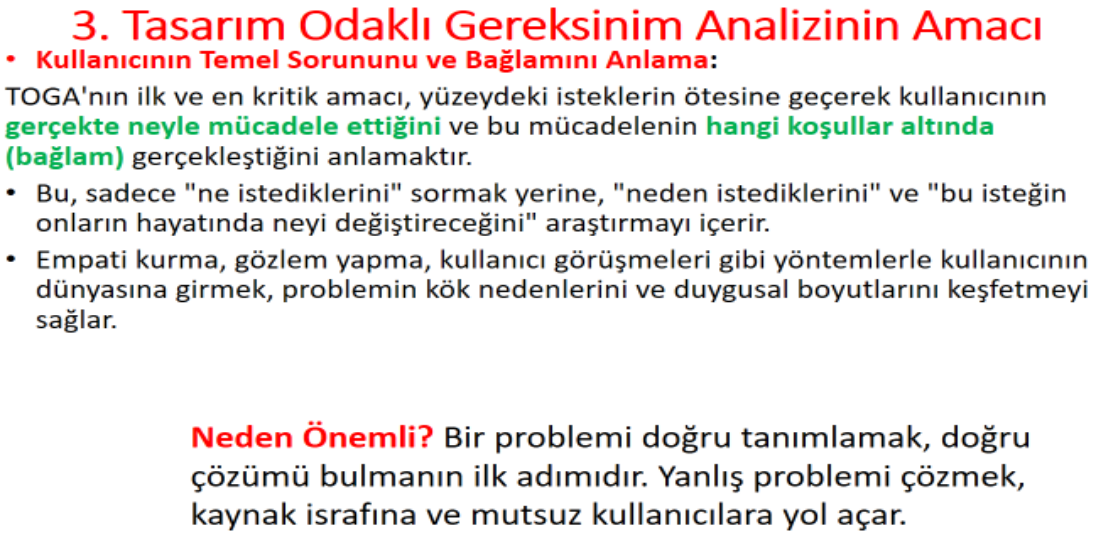
- Barkod
- AI kontrol
- Uyarı sistemi

4. Prototip: Basit ekran tasarımı

5. Test: Hemşire ile test edildi

Sonuç:
✓ Hata azaldı
✓ Süreç hızlandı

5.6. TOGA ve Gereksinim Analizi İlişkisi



TOGA: Gereksinim üretim yöntemidir

Süreç: Empati → Problem → Gereksinim → Tasarım

5.7. En Büyük Hata

TOGA'da yapılan en büyük hata:

✗ Empatiyi atlamak

Sonuç:

- Kullanıcı anlaşılmaz
- Sistem kullanılmaz
- Proje başarısız olur

5.8. Sağlıkta TOGA'nın Gücü

TOGA sayesinde:

✓ Kullanıcı hatası azalır
✓ Sistem kullanımı artar
✓ Klinik güvenlik yükselir

5.9. Yapay Zeka ile TOGA

YZ, TOGA'yı güçlendirir:

Empati:

- Log analizi
- Davranış verisi

Tanımlama: Risk analizi

Test: Simülasyon

5.10. TOGA vs Agile

TOGA ve Agile birlikte çalışır:

TOGA	Agile
Problem bulur	Çözüm üretir
İnsan odaklı	Ürün odaklı

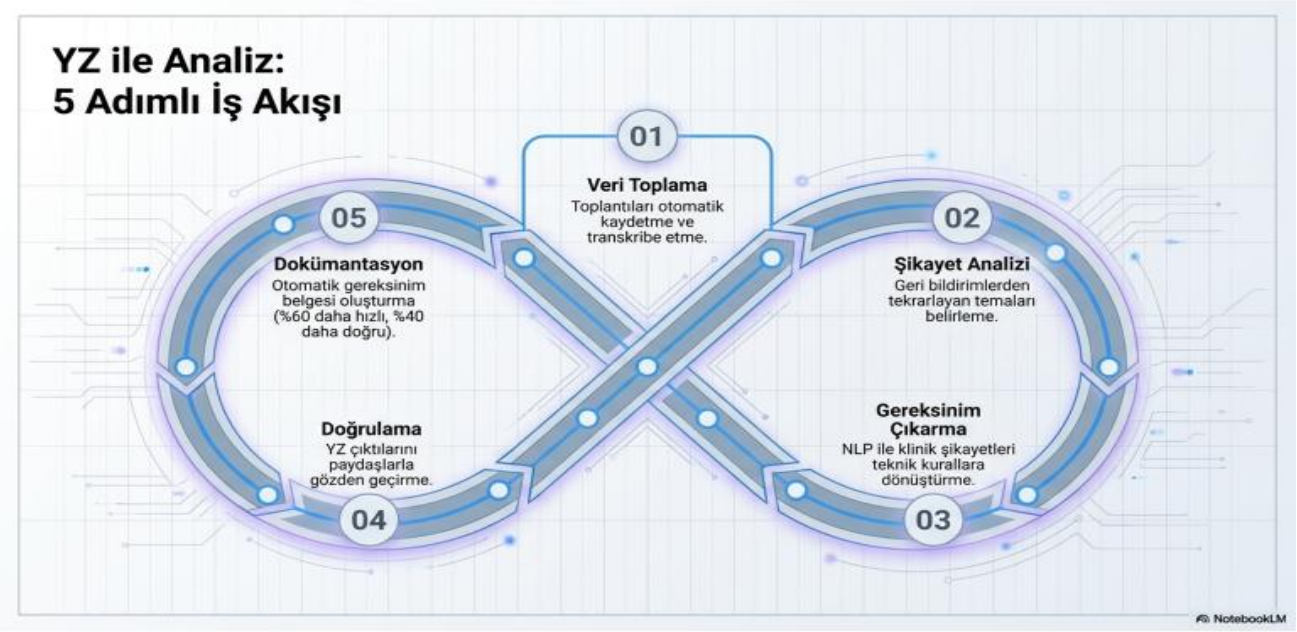
5.12. Kritik Mesaj

☞ **Kullanıcıyı anlamadan sistem tasarlanmaz**

☞ Empati = hasta güvenliği

BÖLÜM-6

VERİ TOPLAMA VE KAYNAKLAR: GEREKSİNİM ANALİZİNİN TEMELİ



6.1. Neden Veri Toplama Kritik?

Gereksinim analizi **varsayımla değil, veriyle yapılır**.

En büyük hata:

✕"Bence problem bu"

✓ "Veri gösteriyor ki problem bu"

Kritik ilke: **Veri yoksa analiz yoktur**

6.2. Sağlıkta Veri Türleri

Sağlık sistemleri çok farklı veri türleri içerir:

1. Klinik Veriler

- Vital bulgular
- Laboratuvar sonuçları
- Tanılar

2. Operasyonel Veriler

- Hasta akışı
- Yatak doluluk oranı
- Bekleme süreleri

3. Kullanıcı Davranış Verileri

- Sistem logları
- Tıklama verileri
- Kullanım süreleri

4. Güvenlik Verileri

- Hata kayıtları
- Olay bildirimleri
- Near-miss kayıtları

☞ **En kritik veri:**

⚠ **Hata verisi**

Çünkü gereksinim çoğunlukla buradan çıkar.

6.3. Veri Kaynakları

1 Gözlem (Shadowing) : Kullanıcıyı izlemek

Nasıl yapılır?

- Hemşire yanında dur
- Süreci baştan sona izle
- Not al

Avantaj:

✓ Gerçek davranış görülür

Dezavantaj:

✕ Zaman alır

2 Mülakat (Interview): Kullanıcıya sormak

Sorulacak doğru sorular:

- En zorlandığın süreç ne?
- En çok hata nerede oluyor?
- Sistem seni nerede yavaşlatıyor?

Kritik hata:

✕"Ne istiyorsun?" diye sormak

☞ "Ne problem yaşıyorsun?" diye sormak

3 **Workshop:** Tüm paydaşları bir araya getirmek

Katılımcılar:

- Hemşire
- Hekim
- **IT**
- Yönetim

Amaç: Ortak problem tanımı oluşturmak

4 **Doküman Analizi:** Mevcut sistemleri incelemek

Kaynaklar:

- Prosedürler
- Kılavuzlar
- Eski gereksinim dokümanları

Risk: ✕ Kağıt üstü süreç ≠ gerçek süreç

5 **Sistem Logları (EN GÜÇLÜ KAYNAK):** Gerçek davranış verisi

Örnek:

- Hangi ekran ne kadar kullanılıyor?
- Hangi işlem yarım kalıyor?
- Hata nerede oluşuyor?

Kritik içgörü: Kullanıcı ne dediği değil, ne yaptığı önemlidir

6 **Olay Bildirim Sistemleri:** Hasta güvenliği verisi

İçerik:

- İlaç hataları
- Yanlış hasta
- Gecikmeler

Altın veri: Near-miss (olay olmadan yakalanan hatalar)

Gereksinim Analizinde Hemşire İçin Veri Kaynakları

Veri Kaynağı Kategorisi	Yöntem Türü	Toplanacak Ham Veri (Ne?)	Hemşire Neden Bu Veriyi Toplar?	YZ/Dijital Destek Aracı
1. Gözlem (Davranışsal)	**Gölge Takibi (Shadowing)**	Hemşirelerin **Gerçekte Ne Yaptığı** (Sistem yavaşladığında el ile not alma, selobant kullanma, kısa yollar)	Kullanıcının size **anlattığı** ile **yaptığı** arasındaki farkı (Gizli Gereksinimi) tespit etmek.	**Kamera/Video Analizi** (Süreç Madenciliği yazılımları, ancak KVKK uyarısı gereklidir)
2. Eser Analizi (Artifacts)	**Doküman İncelemesi**	Post-it'ler, elle tutulan "kopya kağıtları", buruşuk hasta notları, elle tutulan kayıt defterleri	Yazılımın yetersiz kaldığı ve kullanıcıların kendi **"yamalarını"** yaptığı noktaları bulmak	**Claude/ChatGPT:** Ham metinleri (fotoğraflanmış notları) girip "Buradaki 3 ana sorun nedir?" diye sordurmak.
3. Veri Madenciliği (Loglar)	**Sistem Kayıt Analizi (Audit Logs)**	Kullanıcının bir ekranda **kaç saniye kaldığı**, hangi butona **kaç kere tıkladığı**, hataların sıklığı	İnsanların **subjektif** beyanları (Çok yoğunuz) ile sistemdeki **objektif** gerçeği (Sistem Bekleme Süresi) karşılaştırmak	**Gelişmiş LLM'ler (Gemini/GPT-4o Veri Analizi):** Log verisini analiz edip otomasyon gereksinimlerini belirlemek
4. Sözlü/Bağlamsal Sorgulama	**Mülakat / Bağlamsal Sorgulama**	**Kullanıcı Hikayeleri (User Stories)**, istekler ve duygusal yükler (Pain Points), iş akışlarının sözel anlatımı	**"Neden Yaptın?"** diye suçlamadan, sistemin neden işe yaramadığını dürüstçe öğrenmek	**Otter.ai / tldv.io:** Toplantı sesini metne döküp anında **Önemli Maddeleri/Aksiyonları** özetlemek
5. Kanıta Dayalı Literatür	**YZ Destekli Literatür Taraması**	Bir klinik sürecin uluslararası kabul görmüş **"Gold Standard"** veya bilimsel dayanağı olan makaleler	Çözümün "bizim hastanedeki kural" yerine, **kanıta dayalı** (Evidence-Based) olmasını sağlamak	**Perplexity.ai / Consensus:** Sorulara kaynakçalı, bilimsel makale özetleriyle cevap almak

6.4. Veri Toplama Hataları

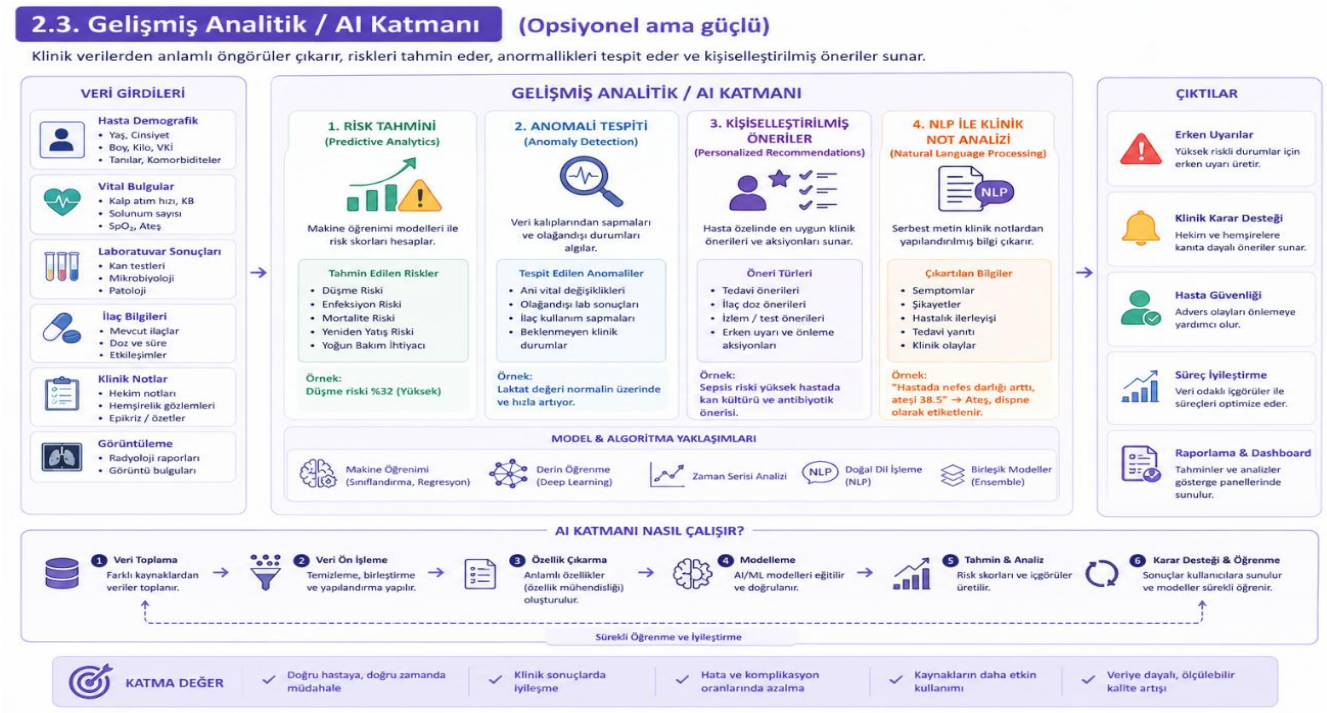
❌ **1. Tek kaynağa güvenmek:** Sadece mülakat yapmak

❌ **2. Kullanıcıya inanmak (tamamen):** Algı ≠ gerçek

❌ **3. Eski veriyi kullanmak:** Süreç değişmiş olabilir

❌ **4. Veri temizliği yapmamak**

6.5. Veri -Çoklu Doğrulama



En doğru yaklaşım: Birden fazla veri kaynağını birleştirmek

Örnek:

- Mülakat → "Sistem yavaş"
- Log → işlem süresi normal
- Gözlem → kullanıcı yanlış akış izliyor

☞ Gerçek problem: **UX tasarımı kötü**

6.6. Yapay Zeka ile Veri Analizi

YZ burada devreye girer:

Kullanım alanları:

- Hata örüntüsü bulma
- Risk tahmini
- Anomali tespiti

Örnek:

YZ analizi sonucu: "Gece 02:00–04:00 arası hata %40 artıyor"

➡ Bu bir gereksinim üretir:

- Gece vardiyası için destek sistemi

6.7. Veri → İçgörü → Gereksinim

Bu dönüşüm kritik:

Adımlar:

Veri
↓
Analiz
↓
İçgörü
↓
Gereksinim

Örnek:

- Veri: İlaç hatası kayıtları
- İçgörü: Yoğun saatlerde artıyor
- Gereksinim: Yoğunluk bazlı uyarı sistemi

6.8. Klinik Veri Güvenliği

Sağlık verisi: **En hassas veri türlerinden biridir**

Gereklilikler:

- KVKK uyumu
- Erişim kontrolü
- Anonimleştirme

Kritik: Analiz yapılırken bile veri korunmalıdır

6.9. Veri Kalitesi (Çok Kritik)

Problem:

- Eksik veri
- Yanlış veri
- Tutarsız veri

Sonuç: Yanlış analiz

Kural: "Garbage in → garbage out"

6.10. Kritik Mesaj

☞ **Gözlemle, doğrula, analiz et**
☞ Veri olmadan karar verme

BÖLÜM-7

SÜREÇ ANALİZİ: SAĞLIKTA GERÇEK AKIŞI ANLAMAK

7.1. Süreç Analizi Nedir?

Süreç analizi: Bir işin baştan sona nasıl gerçekleştiğini anlamaktır

Sağlıkta bu şu anlama gelir:

- Hasta nasıl kabul edilir?
- İlaç nasıl uygulanır?
- Taburculuk nasıl yapılır?

Kritik gerçek: Sistemler değil, **süreçler hata üretir**

7.2. Neden Süreç Analizi Şart?

Çünkü:

❌ Gereksinim → varsayımla yazılırsa
✓ Süreç → gerçek gözle görülür

En büyük hata: ☞ Süreci anlamadan sistem tasarlamak

7.3. Sağlıkta Süreçlerin Özelliği

Sağlık süreçleri:

- Çok adımlı
- Çok paydaşlı
- Dinamik
- Kesintiye açık

Örnek: İlaç uygulama süreci

- Hekim order girer
- Hemşire ilacı hazırlar
- Doğrulama yapılır
- Uygulama yapılır
- Kayıt girilir

☞ Bu zincirde **her adım risk noktasıdır**

7.4. AS-IS (Mevcut Durum Analizi)

AS-IS: ☞ “Şu an ne oluyor?”

Amaç:

- Gerçek süreci görmek
- Problemi ortaya çıkarmak

Nasıl yapılır?

- Gözlem
- Log analizi
- Süreç çizimi

⚠Kritik hata:

✕“Olması gereken süreci çizmek”

✓ “Gerçekte olanı çizmek”

7.5. TO-BE (Hedef Süreç)

TO-BE: ☞ “Nasıl olmalı?”

Amaç:

- Daha güvenli
- Daha hızlı
- Daha verimli süreç

Örnek:

AS-IS: Manuel kontrol

TO-BE: Otomatik doğrulama

7.6. GAP Analizi

GAP: ☞ Mevcut ile hedef arasındaki fark

Basit ifade: AS-IS → TO-BE arasındaki boşluk

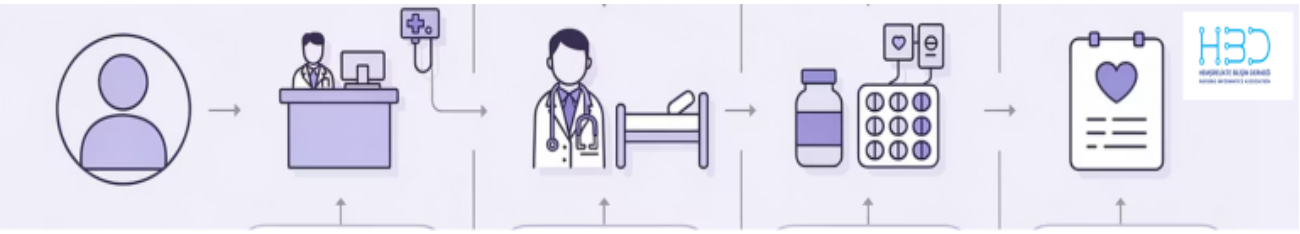
Örnek:

AS-IS	TO-BE	GAP
Kontrol yok	Otomatik kontrol	Sistem eksik

7.7. Süreç Akışı (Flow)

Süreçler genelde şu şekilde modellenir:

Başlangıç
↓
İşlem 1
↓
Karar noktası
↓
İşlem 2
↓
Bitiş



Sağlık örneği:

Hasta kabul
↓
Order girişi
↓
İlaç hazırlama
↓
Doğrulama
↓
Uygulama
↓
Kayıt

7.8. Süreçte Kritik Noktalar

Her süreçte: ☞ "Kırılma noktaları" vardır

Bunlar:

- Karar noktaları
- Manuel işlemler
- İnsan bağımlı adımlar

En riskli alan: ☞ Doğrulama yapılmayan noktalar

7.9. Klinik Süreç Analizi Örneği

Problem: İlaç hatası artıyor

AS-IS analizi:

- Doğrulama yok
- Yoğunluk yüksek
- Sistem uyarı vermiyor

GAP: Kontrol mekanizması eksik

TO-BE:

- Barkod doğrulama
- AI destekli uyarı

7.10. Süreç Analizi Hataları

❌ **1. Teorik süreç çizmek:** Gerçekten kopuk

❌ **2. Paydaşları dahil etmemek**

❌ **3. Detay atlamak**

❌ **4. Dinamikliği göz ardı etmek**

7.11. Yapay Zeka ile Süreç Analizi

Canlı Demo: YZ ile Süreç Şeması Oluşturma

Süreci YZ'ye Anlatın

"Hasta kabul sürecini Mermaid.js kodu olarak ver" gibi net bir prompt ile başlayın

Kod Üretimi

YZ, süreç adımlarını analiz ederek Mermaid.js formatında şema kodu üretir

Kodu Yapıştırın

Üretilen kodu Mermaid Live Editor'a aktarın

Anında Görselleştirme

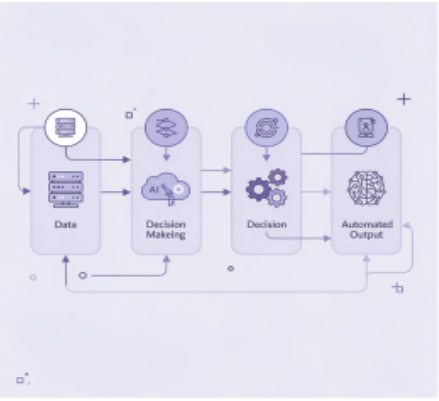
Kod otomatik olarak profesyonel akış şemasına dönüşür

Pratik İpucu: Mermaid Live Editor (mermaid.live) ücretsiz bir online araçtır. Kodunuzu kopyalayıp yapıştırmanız yeterli — hemen sonucu görürsünüz ve dilediğiniz formatta (PNG, SVG) indirebilirsiniz.

Bu yöntem, teknik bilgi gerektirmeden herkesin süreç diyagramları oluşturmasını mümkün kılar. Kalite ekipleri, hemşireler ve yöneticiler artık süreçlerini hızlıca dokümante edebilir.

Yapay Zeka ile Süreç Çizimi: Mermaid.js

Geçmişte saatlerce süren teknik çizim çalışmaları, artık yapay zeka sayesinde dakikalar içinde gerçekleşiyor. Sürecinizi anlatın, yapay zeka size hazır şema kodu üretsin.



https://www.mermaidchart.com/app/projects/7c0444e2-0acf-4f9d-8399-c07d0b984409/diagrams/ecfd0e56-423a-4572-9790-dfd6f5aa41ba/version/v0.1/edit

Avantajları:

- Hızlı ve kolay üretim
- Anında güncelleme imkanı
- Profesyonel görünüm
- YZ entegrasyonu ile otomasyon

"Bana bu süreci, tüm karar noktalarını göstererek 'Mermaid.js' formatında bir kod bloğu olarak ver."

YZ şu katkıları sağlar:

☞ **Süreç madenciliği (Process Mining)**

- Gerçek akışı ortaya çıkarır
- Gizli süreçleri bulur

☞ **Anomali tespiti**

- Normal dışı akışları yakalar

☞ **Performans analizi**

- Süre gecikmelerini gösterir.

7.12. Süreç → Gereksinim Dönüşümü

En kritik nokta: ☞ Süreç analizinin çıktısı gereksinimdir

Örnek:

Süreç: Doğrulama yok

☞ Gereksinim:

☞ "İlaç uygulaması öncesi doğrulama yapılmalıdır"

7.13. Klinik Güvenlik Perspektifi

Her süreç sorusu şu olmalı: ☞ "Burada hasta zarar görebilir mi?"

Eğer cevap "evet" ise: ☞ O nokta **kritik gereksinim üretir**

7.14. Kritik Mesaj

☞ Süreci anlamadan sistem tasarlama

☞ Gerçek akış = gerçek problem

Yapay Zeka ile Süreç Çizimi: Mermaid.js

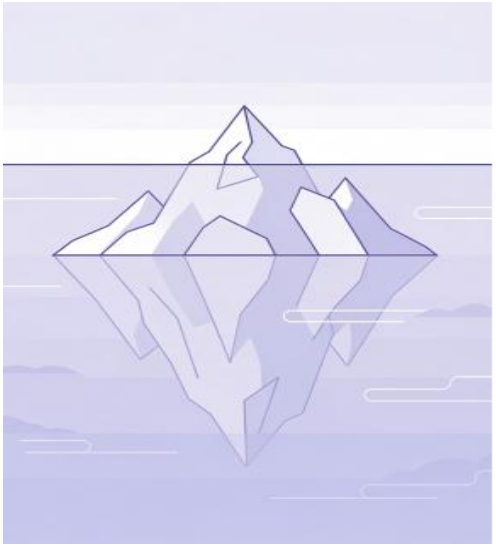
Program Adı	Özellik/Kullanım Alanı	YZ Entegrasyonu
PlantUML	Akış diyagramları, sıra (sequence) diyagramları ve sınıf diyagramları gibi daha geniş bir yelpazeyi destekler.	PlantUML sözdizimi, YZ araçları tarafından üretilebilir.
Graphviz	Açık kaynaklı bir grafik görselleştirme aracıdır. Büyük miktarda veriyi programatik olarak görselleştirmek için kullanılır.	Metin tabanlı giriş dosyası kullanır, bu da YZ tarafından kod üretimine uygundur.
Miro AI	Büyük bir sanal beyaz tahta uygulamasıdır. İçinde bulunan YZ, metin komutuyla zihin haritaları, akış şemaları ve görsel düzenlemeler oluşturabilir.	**Yüksek:** Miro Assist özelliği, basit metin komutlarıyla karmaşık diyagramları otomatik olarak çizer.
Whimsical AI	Akış şemaları, zihin haritaları (mind maps) ve bilgi mimarisi çizimleri için tasarlanmıştır. Hızlı ve sade bir arayüze sahiptir.	**Yüksek:** YZ ile şema ve zihin haritası oluşturma özelliklerine sahiptir.
Jotform Yapay Zeka İş Akışı Oluşturucusu	İş akışlarını saniyeler içinde tasarlamak için kullanılır.	**Yüksek:** YZ komutları ile iş akışı diyagramları oluşturur.

YZ Araçları ve Eğitim Uygulama Komutları (Promptlar)

Araç Adı	Uygulama Linki	Uygulama Örneği (Senaryo)	Bu Araç İçin Kullanılacak Prompt (Komut)
ChatGPT / Gemini	gemini.google.com	Klinik şikayeti teknik gereksinime dönüştürme.	"Bir kıdemli iş analisti rolünü üstlen. Bir hemşirenin 'Hasta kayıtlarına hızlı erişim istiyoruz, acil durumlarda dakikalar önemli' şikayetini , 'Kullanıcı Hikayesi' (User Story) formatında, kabul kriterleri ve teknik veritabanı gereksinimleri dahil olacak şekilde yaz."
	Mermaid Live Editor	mermaid.live	Manuel bir süreci dijital akış şemasına dönüştürme.
	Perplexity.ai	perplexity.ai	Kanıta dayalı klinik karar destek sistemleri (CDSS) literatürü.
	Miro AI	miro.com	AS-IS / TO-BE ve Gap Analizi tasarımı.
	Jotform AI	jotform.com	Akıllı dijital form ve bildirim akışı tasarımı.
	Claude (Artifacts)	claude.ai	Kök Neden Analizi (5 Neden / Balık Kılçığı).

BÖLÜM-8

KÖK NEDEN ANALİZİ: GERÇEK PROBLEMİ BULMA SANATI



8.1. Kök Neden Analizi Nedir?

Kök neden analizi (Root Cause Analysis – RCA): Bir problemin görünen değil, asıl nedenini bulma sürecidir

Kritik fark:

Yüzey Problem	Kök Neden
İlaç hatası	Doğrulama sistemi yok
Gecikme	Süreç darboğazı

Altın kural: Belirtiyi değil, nedeni çöz

8.2. Neden Bu Kadar Kritik?

Çünkü:

❌Belirti çözülür → problem tekrar eder
✓ Kök neden çözülür → problem ortadan kalkar

Klinik örnek:

Problem: "Yanlış ilaç verildi"

Yüzey çözüm:

- Hemşireyi uyar ❌

Kök neden:

- Sistem doğrulama yapmıyor ✓

8.3. Sağlıkta Kök Neden Perspektifi

Modern sağlık yaklaşımı:

✗"Kim yaptı?"

✓ "Sistem neden izin verdi?"

Bu yaklaşım:

☞ Hasta güvenliğini artırır

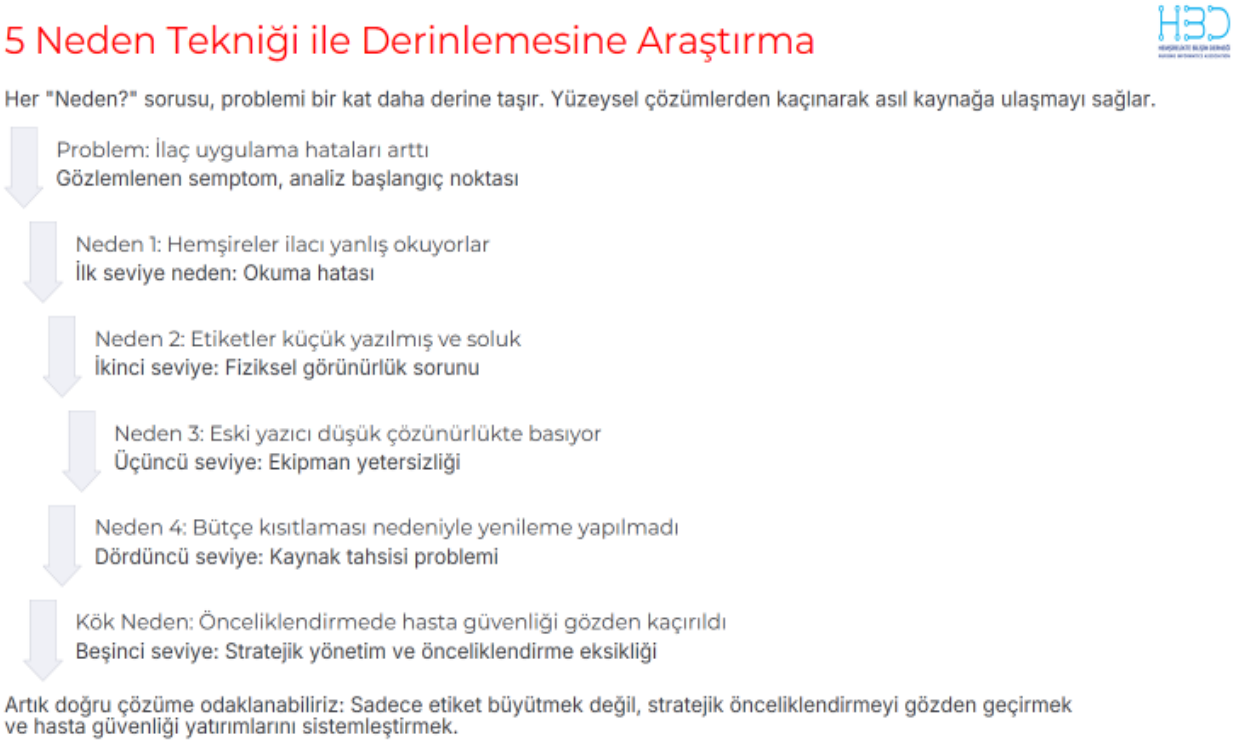
☞ Suçlama kültürünü azaltır

8.4. 5 Neden (5 Why) Tekniği

En basit ve güçlü yöntem: "Neden?" sorusunu tekrar sormak

Mantık: Her cevap → yeni bir "neden" sorusu doğurur

Sorun→N1→N2→N3→N4→Ko¨k Neden\text{Sorun} \rightarrow N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4 \rightarrow \text{Kök Neden}Sorun→N1→N2→N3→N4→Ko¨k Neden



Klinik örnek:

Problem: Yanlış ilaç verildi

1. Neden?
→ Yanlış ilaç seçildi

2. Neden?
→ Benzer isimli ilaçlar karıştırıldı

3. Neden?
→ Sistem uyarı vermedi

4. Neden?
→ Kural tanımlı değil

5. Neden?
→ Gereksinim analizinde eksik

☞ **Kök neden: Analiz hatası**

8.5. Ishikawa (Balık Kılçığı) Diyagramı

Daha kompleks analizler için kullanılır. Problemi kategorilere ayırır

Temel kategoriler:

- İnsan (People)
- Süreç (Process)
- Teknoloji (Technology)
- Ortam (Environment)
- Malzeme (Material)

Mantık:

```
          Problem
            ↑
 -------------------------
İnsan  Süreç  Teknoloji  Ortam
```

Klinik örnek: Problem: İlaç hatası

İnsan:

- Yorgunluk
- Deneyim eksikliği

Süreç: Standart yok

Teknoloji: Uyarı sistemi yok

Ortam: Yoğunluk

☞ **Bu analiz: çoklu kök nedenleri ortaya çıkarır**

8.6. Kök Neden vs Semptom

En kritik ayrım:

❌ Semptom:

- Hata arttı
- Sistem yavaş

✅ Kök neden:

- Süreç tasarımı hatalı
- Veri eksik
- Kontrol mekanizması yok

8.7. Kök Neden Analizi Hataları

❌ **1. İlk cevabı kabul etmek:** Derine inmemek

❌ **2. Kişiye odaklanmak:** Sistem göz ardı edilir

❌ **3. Veri kullanmamak**

❌ **4. Tek neden varsaymak**

8.8. Klinik Senaryo (Derin Analiz)

Problem: Yoğun bakımda ilaç hatası artıyor

5 Why + Ishikawa birleşimi:

- İnsan → yorgunluk
- Süreç → kontrol yok
- Teknoloji → uyarı yok
- Ortam → yoğunluk

☞ **Sonuç: Tek neden yok → sistemsel problem**

8.9. Yapay Zeka ile Kök Neden Analizi

YZ şu alanlarda destek sağlar:

☞ **Örüntü analizi:** Hata nerede yoğunlaşıyor?

☞ **Korelasyon analizi:** Hangi faktörler birlikte artıyor?

☞ **Anomali tespiti :** Normal dışı durumlar

Örnek: "Gece vardiyası + yeni personel = hata artışı"

➞Bu doğrudan kök neden ipucudur

8.10. Kök Nedenden Gereksinime

En kritik dönüşüm:

Kök neden: Doğrulama sistemi yok

Gereksinim: "Sistem ilaç uygulaması öncesi doğrulama yapmalıdır"

8.11. Klinik Güvenlik Perspektifi

Her analiz sonunda şu sorulmalı: "Bu tekrar olur mu?"

Eğer cevap "evet" ise: Kök neden çözülmemiştir

8.12. Bölüm Özeti

Bu bölümde:

- Kök neden analizi
- 5 Why
- Ishikawa
- Klinik uygulamalar ele alındı.

8.13. Kritik Mesaj

☞ **Sorunu değil, nedeni çöz**

☞ Derine inmeden sistem kurulmaz

BÖLÜM-9

KLİNİK KARAR DESTEK SİSTEMLERİ (CDSS): MİMARİ VE AKIŞ TASARIMI

9.1. CDSS Nedir?

Klinik Karar Destek Sistemi (CDSS): Sağlık profesyonellerine doğru zamanda, doğru bilgiyi sunarak karar sürecini destekleyen sistemdir

Kritik vurgu:

☞ CDSS **karar vermez**

☞ **karar destekler**

9.2. Neden CDSS Gereklidir?

Sağlık sistemlerinde:

- Bilgi çok fazla
- Zaman kısıtlı
- Hata toleransı düşük

Bu nedenle:

☞ İnsan tek başına yeterli değildir

☞ Sistem desteği gerekir

Amaç:

- Hata azaltmak
- Kaliteyi artırmak
- Standardizasyon sağlamak

9.3. CDSS Türleri

1 Kural Tabanlı CDSS

- IF–THEN mantığı
- Örnek: "Alerji varsa uyarı ver"

2 Yapay Zeka Tabanlı CDSS

- Veriden öğrenir
- Risk tahmini yapar

3 Hibrit CDSS: En güçlü model

- Kural + YZ birlikte çalışır

9.4. CDSS Mimarisi (Temel Yapı)

Bir CDSS sistemi aşağıdaki bileşenlerden oluşur:

1. Veri Kaynakları

- EHR (hasta kayıtları)
- Laboratuvar
- Vital bulgular
- Cihaz verileri

2. Entegrasyon Katmanı

- **HL7 / FHIR**
- API bağlantıları

3. Kural Motoru

- Klinik kurallar
- Protokoller

4. Yapay Zeka Katmanı

- Risk skorları
- Tahmin modelleri

5. Karar / Uyarı Motoru

- Alert
- Öneri

6. Kullanıcı Arayüzü

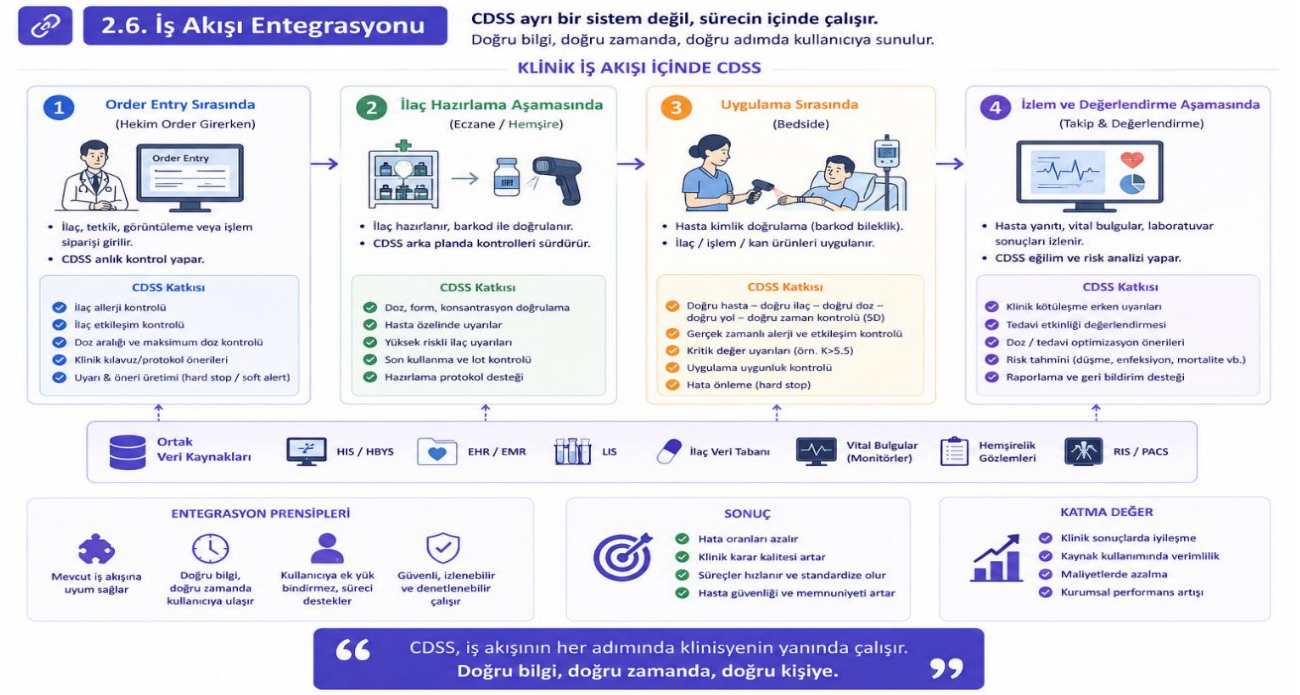
- Hekim
- Hemşire

7. Geri Bildirim Katmanı

- Öğrenme
- Sistem iyileştirme

9.5. CDSS Akış Modeli

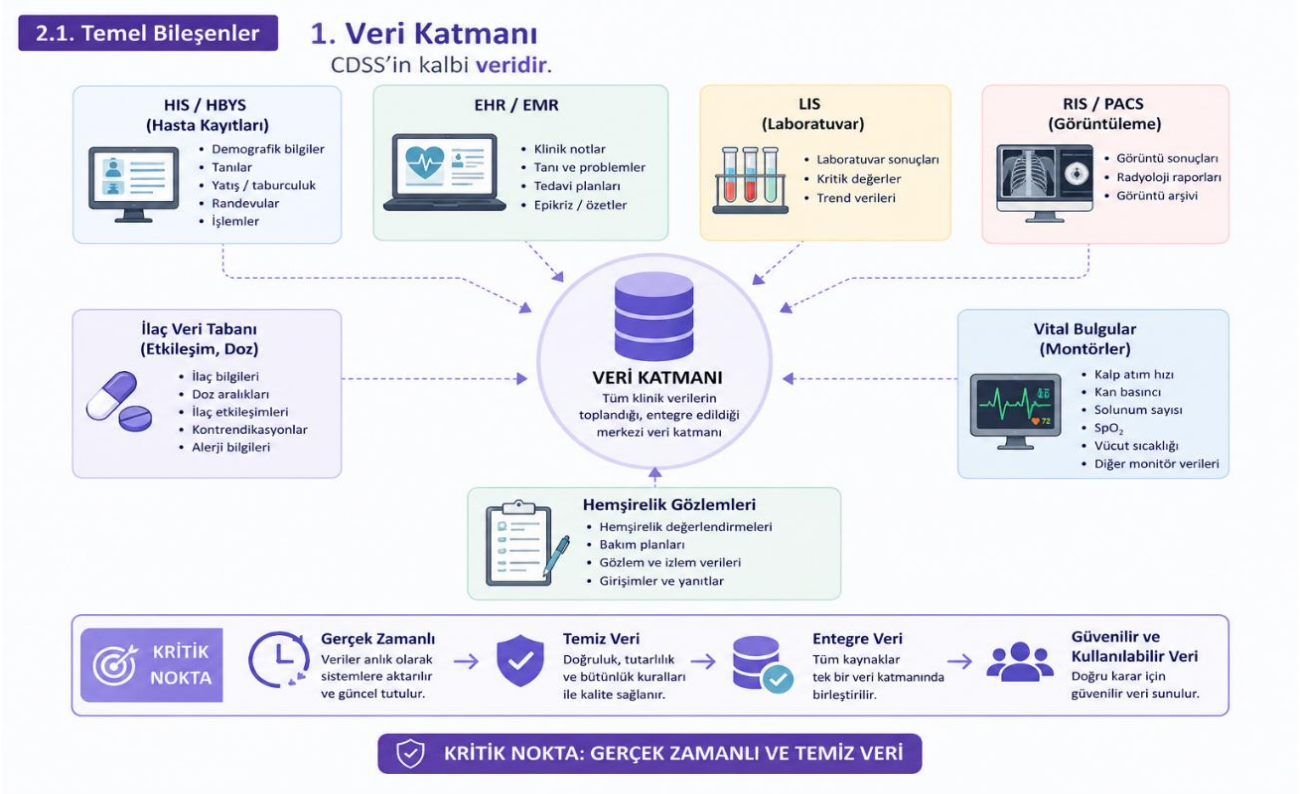
Veri → Entegrasyon → Kural Motoru → AI → Uyarı → Kullanıcı → Feedback

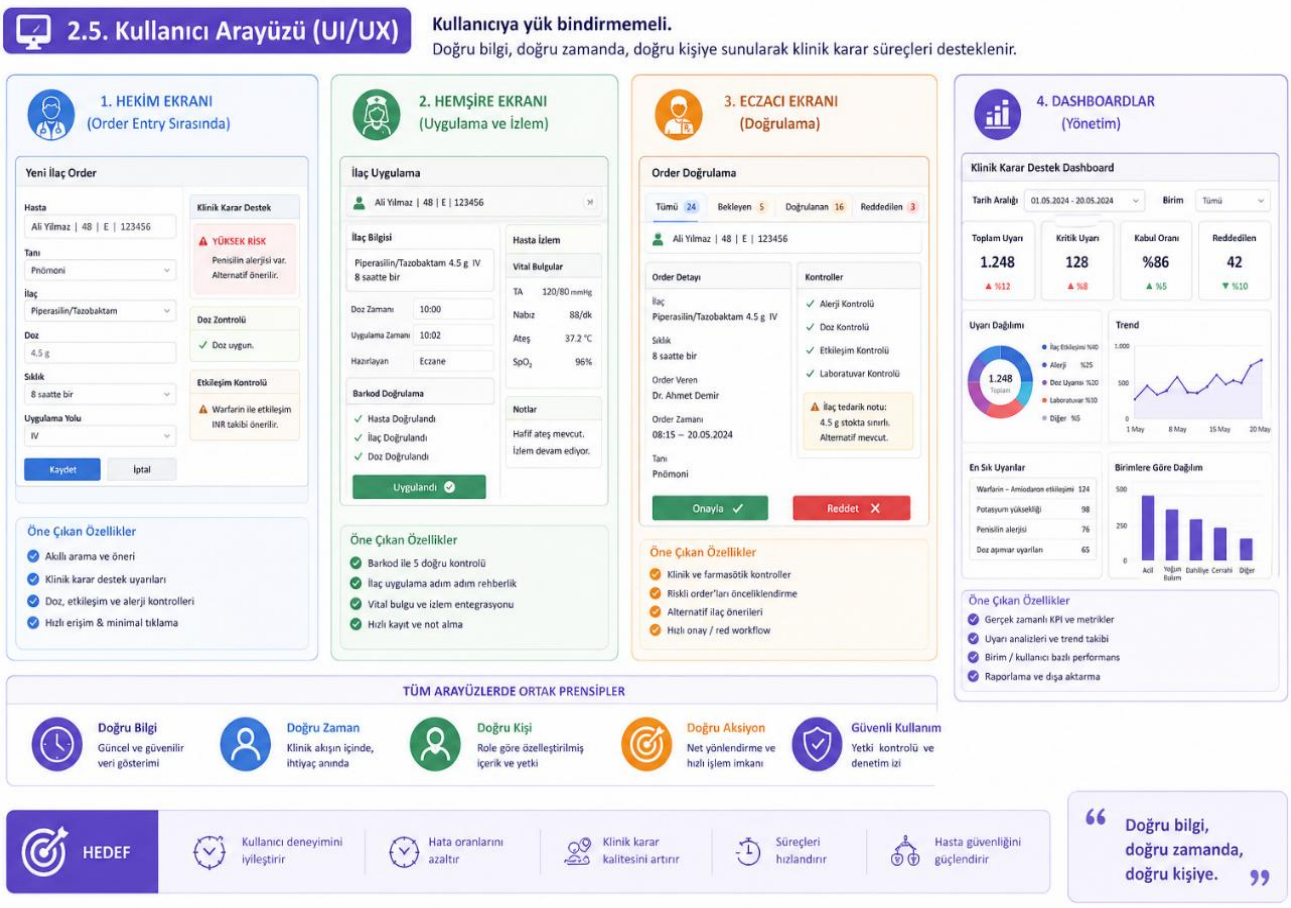


Açıklama:

1. Veri toplanır
2. Sistem analiz eder
3. Risk hesaplanır
4. Uyarı oluşturulur
5. Kullanıcı aksiyon alır
6. Sistem öğrenir

9.6. Klinik Senaryo (Gerçek Sistem)





Durum: Yoğun bakımda sepsis riski

Sistem:

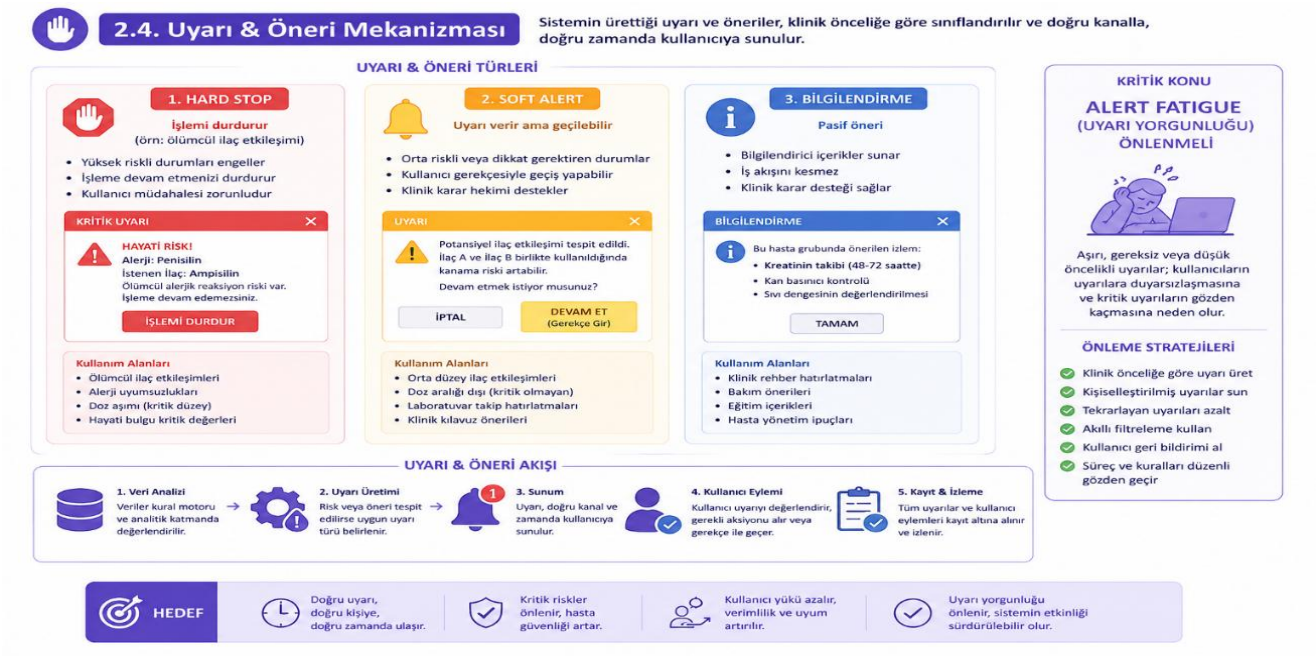
- Vital bulgular alınır
- YZ analiz eder
- Risk %85

Çıktı: "Sepsis riski yüksek" uyarısı

Sonuç:

- Erken müdahale
- Hayat kurtarma

9.7. Alert (Uyarı) Tasarımı – Kritik Konu



En önemli bileşen: **Uyarı sistemi**

İyi uyarı:

✓ Doğru zamanda
✓ Doğru kişiye
✓ Doğru içerikle

Kötü uyarı:

✗Çok fazla
✗Gereksiz
✗Sürekli tekrar

Sonuç: ⚠ Alert Fatigue (uyarı yorgunluğu)

9.8. Uyarı Önceliklendirme

Yüksek Risk

- Hayati risk
- Zorunlu müdahale

Orta Risk

- Dikkat gerektirir

Düşük Risk

- Bilgilendirme

9.9. CDSS Tasarım Hataları

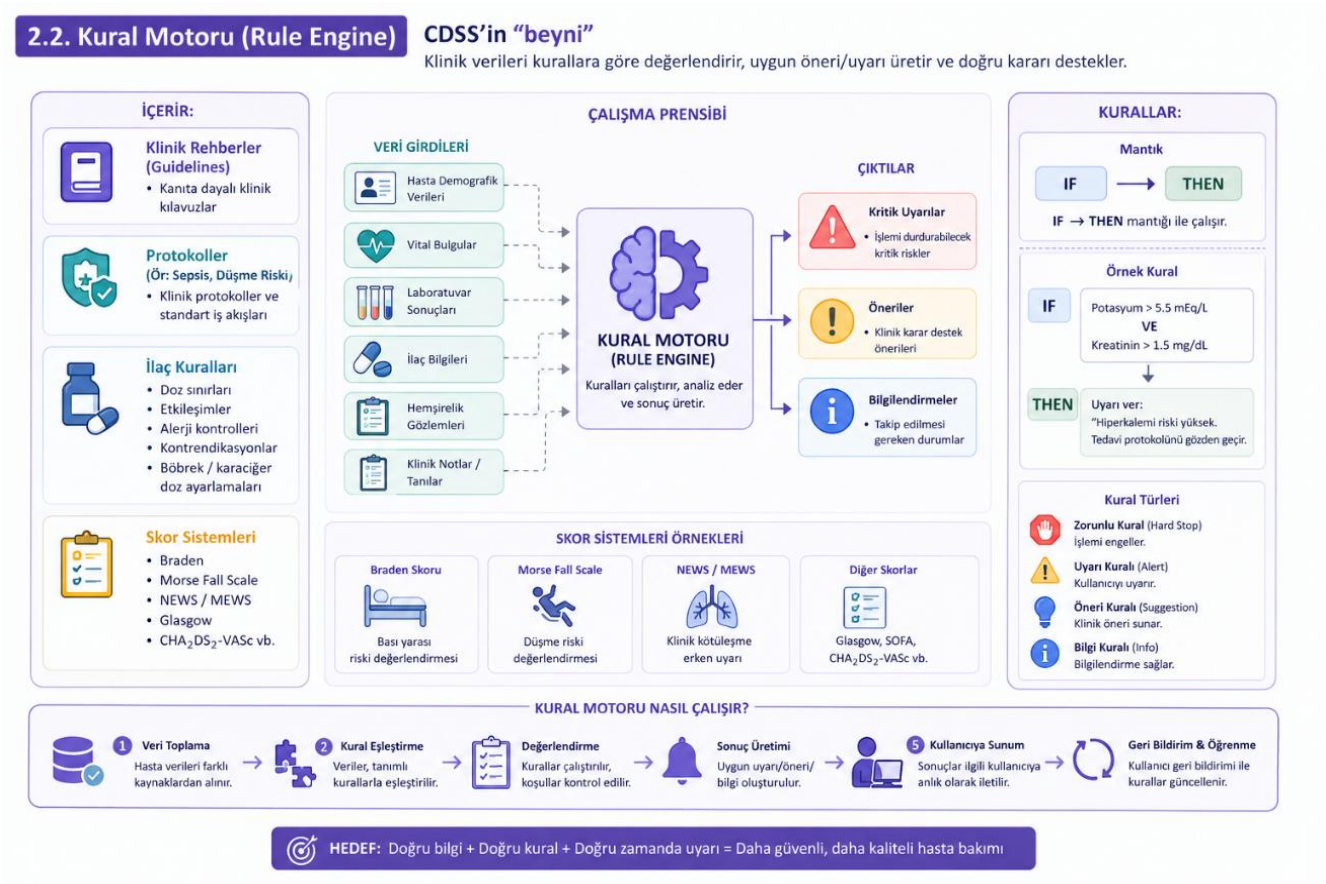
❌ **1. Fazla uyarı:** Kullanıcı ignore eder

❌ **2. Yanlış zamanlama:** Geç uyarı = değersiz

❌ **3. Klinik bağlam eksikliği**

❌ **4. Kullanıcıya yük bindirmek**

9.10. Yapay Zeka ile CDSS



YZ'nin rolü:

☞ **Risk tahmini:** Sepsis, düşme riski

☞ **Öneri üretme:** Tedavi seçenekleri

☞ **Öğrenme:** Sistem zamanla gelişir

9.11. Açıklanabilirlik (Explainability)

Kritik konu: "Sistem neden bu uyarıyı verdi?"

Gereklilik:

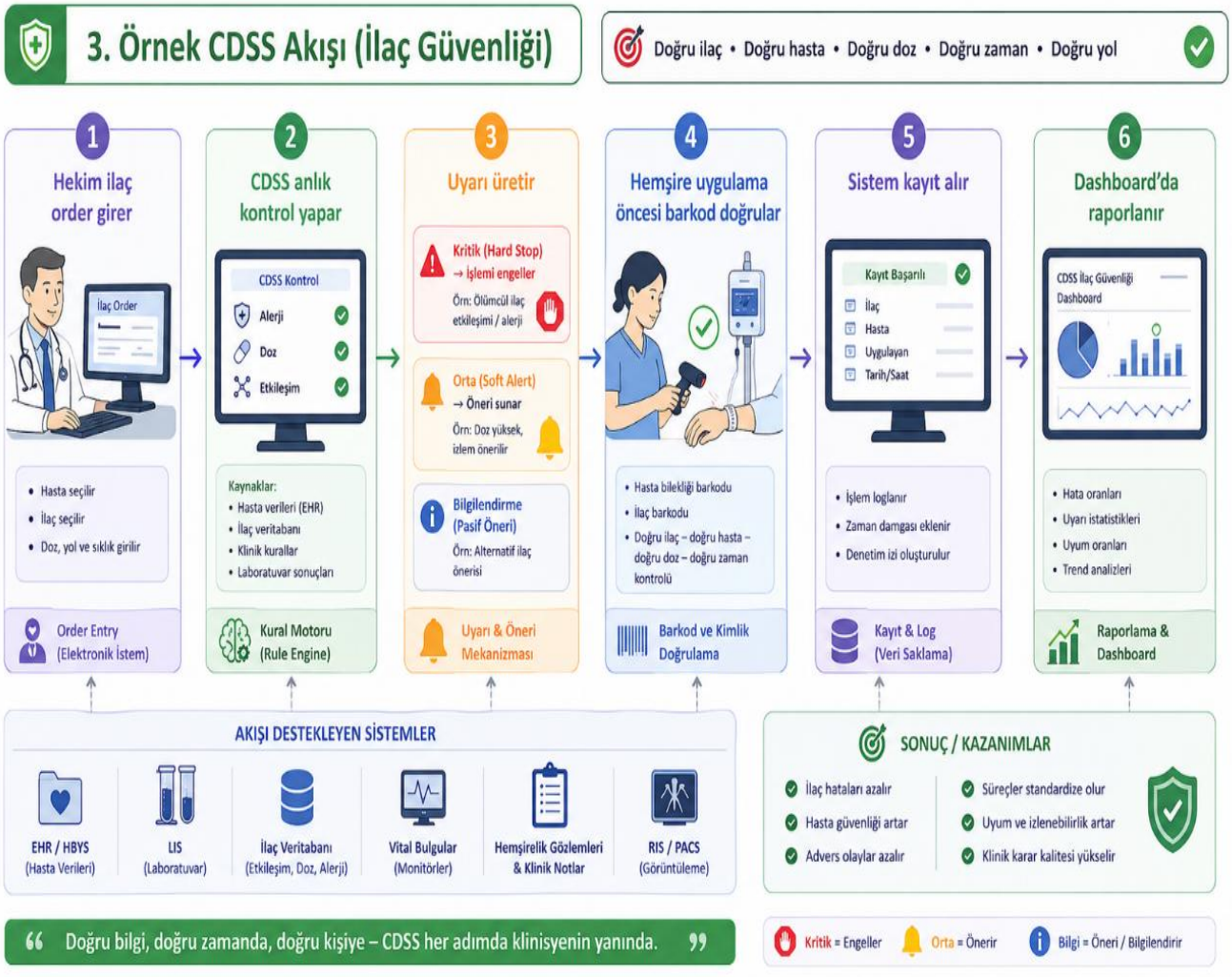
- Şeffaflık
- Güven
- Klinik kabul

9.12. CDSS ve Regülasyon

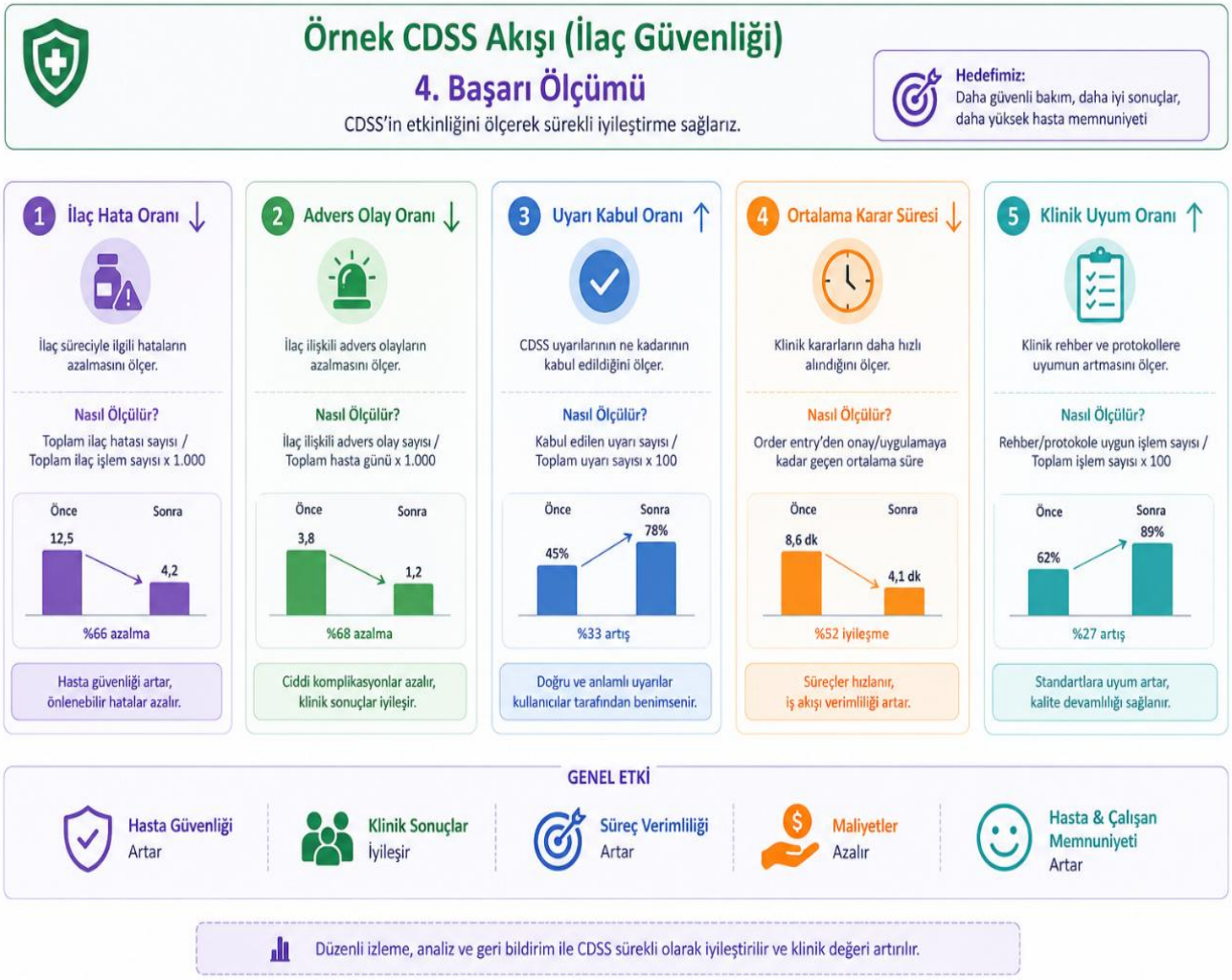
CDSS sistemleri: Tıbbi cihaz olarak değerlendirilebilir

Gereklilikler:

- Validasyon
- Klinik test
- Sürekli izleme



9.13. Başarılı CDSS İçin Altın Kurallar



- Doğru veri
- Doğru kural
- Doğru zaman
- Doğru kullanıcı

9.14. Bölüm Özeti

Bu bölümde:

- CDSS tanımı
- Mimari yapı
- Klinik kullanım
- Tasarım hataları ele alındı.

9.15. Kritik Mesaj

☞ CDSS = teknoloji değil, **klinik güvenlik sistemidir**
☞ Yanlış CDSS → sistematik hata üretir

BÖLÜM-10

VAKA ANALİZİ: İLAÇ HATALARI VE İTAKİ ÜZERİNDEN UÇTAN UCA GEREKSİNİM ANALİZİ

10.1. Vaka Tanımı

Bu bölümde iki kritik senaryo ele alınacaktır:

1. **İlaç uygulama hataları**
2. **İTAKİ (İlaç Takip) süreci**

Problem: ☞ Sağlık sistemlerinde ilaç hataları:

- En sık görülen hatalardandır
- Yüksek risk içerir
- Önlenebilir

10.2. Klinik Gerçeklik

Sahadaki durum:

- Hemşire yoğun
- Süreç hızlı
- Kontrol mekanizmaları zayıf

Sonuç:

☞ Yanlış ilaç
☞ Yanlış doz
☞ Yanlış hasta

10.3. AS-IS (Mevcut Süreç)

İlaç uygulama süreci:

Hekim order girer
↓
Hemşire ilacı hazırlar
↓
Hasta kontrolü (çoğu zaman manuel)
↓
İlaç uygulanır
↓
Kayıt girilir

Problemler:

- Doğrulama yok
- İnsan bağımlı
- Sistem uyarı vermiyor

10.4. Kök Neden Analizi

Problem: ☞ Yanlış ilaç uygulaması

5 Why:

- Neden? → Yanlış ilaç seçildi
- Neden? → Benzer isim
- Neden? → Uyarı yok
- Neden? → Sistem kuralı yok
- Neden? → Gereksinim eksik

☞ **Kök neden: sistem tasarım eksikliği**

10.5. Ishikawa Analizi

Kategori bazlı nedenler:

İnsan:

- Yorgunluk
- Dikkat dağınıklığı

Süreç:

- Standart yok
- Kontrol adımı eksik

Teknoloji:

- Uyarı sistemi yok
- Entegrasyon eksik

Ortam:

- Yoğunluk
- Gürültü

☞ Sonuç: Problem **çok faktörlü**

10.6. Gereksinim Tanımlama

Analiz sonrası üretilen gereksinimler:

Kritik gereksinimler:

- "Sistem ilaç uygulaması öncesinde doğrulama yapmalıdır"
- "Yanlış ilaç seçildiğinde uyarı vermelidir"
- "Hasta–ilaç eşleşmesi kontrol edilmelidir"

Destek gereksinimler:

- İşlem süresi kısa olmalı
- Kullanımı kolay olmalı

10.7. TO-BE (Hedef Süreç)

Yeni süreç:

Order girişi
↓
İlaç hazırlama
↓
Barkod / AI doğrulama
↓
Uyarı sistemi
↓
Uygulama
↓
Otomatik kayıt

10.8. CDSS Entegrasyonu

Yeni sistem:

- Hasta verisi alınır
- İlaç bilgisi kontrol edilir
- Risk hesaplanır
- Uyarı verilir

Örnek uyarı: "Bu ilaç bu hastaya uygun değil"

10.9. İTAKİ Süreci (Örnek Senaryo)

İTAKİ sisteminde:

AS-IS:

- Manuel kontrol
- Kağıt bağımlı süreç
- İzlenebilirlik düşük

Problemler:

- Gecikme
- Hata riski
- Veri kaybı

TO-BE:

- Dijital takip
- Otomatik uyarı
- Gerçek zamanlı izleme

10.10. Yapay Zeka Entegrasyonu

YZ sistemi:

- Hata örüntülerini öğrenir
- Riskli durumları tahmin eder

Örnek: "Bu hasta için doz riski yüksek"

10.11. Kazanımlar

Yeni sistem ile:

✓ **Klinik**

- Hata azalır
- Güvenlik artar

✓ **Operasyonel**

- Süre hızlanır
- İş yükü azalır

✓ **Stratejik**

- Veri oluşur
- Karar desteklenir

10.12. En Kritik Ders

Bu vaka şunu gösterir:

☞ Teknoloji problemi çözmez

☞ **Doğru analiz + doğru sistem çözer**

10.13. Kritik Mesaj

☞ **Her sistem bir problemden doğar**

☞ Problem doğruysa sistem değer üretir

BÖLÜM-11

YAPAY ZEKA İLE GEREKSİNİM ANALİZİ: VERİDEN İÇGÖRÜYE, İÇGÖRÜDEN SİSTEME

11.1. Paradigma Değişimi

Geleneksel gereksinim analizi:

- Gözlem
- Mülakat
- Workshop

Yapay zeka destekli analiz: **Veri → örüntü → öngörü → gereksinim**

Kritik fark:

Geleneksel	YZ Destekli
Subjektif	Veri odaklı
Sınırlı gözlem	Büyük veri
Reaktif	Proaktif

11.2. Yapay Zeka Gereksinim Analizini Nasıl Dönüştürür?

YZ üç ana katkı sağlar:

1 Görünmeyeni Görür

İnsan gözünden kaçan örüntüleri yakalar

Örnek: “Gece vardiyasında hata oranı %37 daha yüksek”

2 Tahmin Eder

Gelecekteki riskleri öngörür

Örnek: “Bu hasta 12 saat içinde sepsis riski taşıyor”

3 Öneri Üretir

Çözüm seçenekleri sunar

Örnek: “Ek doğrulama mekanizması gerekli”

11.3. YZ Destekli Analiz Süreci

Veri Toplama

↓

Veri Temizleme

↓

Analiz (ML / AI)

↓

İçgörü

↓

Gereksinim

↓

Sistem Tasarımı

11.4. Veri Kaynakları (YZ İçin)

YZ'nin gücü veriden gelir:

Klinik veri:

- Vital bulgular
- Lab sonuçları

Sistem logları:

- Kullanıcı davranışı
- İşlem süreleri

Olay verisi:

- Hatalar
- Near-miss

En değerli veri: **Hata verisi**

11.5. Kullanılan YZ Teknikleri

Makine Öğrenmesi

- Sınıflandırma
- Tahmin

Doğal Dil İşleme (NLP)

- Doküman analizi
- Klinik notlar
-

Anomali Tespiti

- Normal dışı durumlar

Süreç Madenciliği

- Gerçek akışı ortaya çıkarır

11.6. örnek Klinik Senaryo (YZ ile Analiz)

Veri:

- 6 aylık ilaç hatası kayıtları

YZ analizi:

- Gece vardiyası → %40 artış
- Yeni personel → risk yüksek
- Yoğun saat → hata artıyor

İçgörü:

Hata **zaman + deneyim + yoğunluk** ile ilişkili

Gereksinim:

- Gece vardiyası destek sistemi
- Yeni personel için kontrol mekanizması

11.7. Prompt Mühendisliği (Yeni Beceri)

YZ ile çalışmak için: Doğru soru sormak gerekir

❌ **Kötü prompt:** "Veriyi analiz et"

✅ **İyi prompt:** "İlaç hatalarının zaman, kullanıcı deneyimi ve yoğunluk ile ilişkisini analiz et"

11.8. İnsan + YZ İş Birliği

YZ güçlüdür ama tek başına yeterli değildir.

En doğru model: Human-in-the-loop

Rol dağılımı:

İnsan	YZ
Klinik yorum	Veri analizi
Etik karar	Örüntü bulma

11.9. Riskler ve Sınırlılıklar

❌ **Veri bias**

→ Yanlı sonuç

❌ **Aşırı güven**

→ Sorgulama azalır

❌ **Açıklanabilirlik**

→ "Neden?" bilinmez

11.10. Gereksinim Üretiminde YZ Kullanımı

YZ çıktısı doğrudan gereksinime dönüşebilir:

YZ çıktısı: "Yanlış hasta seçimi %12"

Gereksinim: "Hasta kimlik doğrulama zorunlu olmalıdır"

11.11. Gelecek Perspektifi

Yakın gelecekte:

- Gereksinim analizi → yarı otomatik
- Süreç analizi → gerçek zamanlı
- CDSS → sürekli öğrenen sistem

11.12. Kritik Mesaj

☞ **Veri konuşur, YZ yorumlar, insan karar verir**

BÖLÜM-12

TEKNİK GEREKSİNİMLER VE ENTEGRASYON: SAĞLIK SİSTEMLERİNİN OMURGASI

12.1. Teknik Gereksinim Nedir?

Teknik gereksinimler: **Sistemin çalışabilmesi için gerekli altyapı ve teknik koşullardır**

Önemli ayrım:

- Fonksiyonel: Sistem ne yapar
- Teknik: Sistem **nasıl çalışır**

Örnek:

Fonksiyonel: "İlaç doğrulama yapılmalı"

Teknik: "Barkod okuyucu ile hasta bilekliği okutulmalı"

12.2. Sağlıkta Teknik Karmaşıklık

Sağlık sistemleri:

- Birbirinden bağımsız sistemlerden oluşur
- Farklı veri formatları kullanır
- Gerçek zamanlı çalışır

Örnek sistemler:

- HBYS (Hastane Bilgi Yönetim Sistemi)
- LIS (Laboratuvar)
- RIS (Radyoloji)
- PACS (Görüntüleme)

Problem:

✗ Sistemler konuşamazsa → veri akmaz
✓ Entegrasyon şart

12.3. Entegrasyon Nedir?

Entegrasyon: **Farklı sistemlerin veri alışverişi yapabilmesi**

Amaç:

- Tek veri kaynağı
- Tutarlılık
- Gerçek zamanlı bilgi

12.4. HL7 Nedir?

HL7: Sağlık verisi paylaşım standardı

Özellik:

- Mesaj tabanlı
- Esnek
- Yaygın

Örnek mesaj:

- Hasta kabul
- Lab sonucu
- Order bilgisi

12.5. FHIR Nedir?

FHIR (Fast Healthcare Interoperability Resources): Modern sağlık veri standardı

Özellik:

- API tabanlı
- **JSON / REST**
- Web uyumlu

12.6. HL7 vs FHIR

HL7	FHIR
Eski	Modern
Mesaj	API
Karmaşık	Basit

☞ **Güncel yaklaşım: FHIR + API mimarisi**

12.7. Sistem Mimarisi (Katmanlı Yapı): Modern sağlık sistemleri katmanlıdır

1. Veri Katmanı

- Veritabanı
- Klinik veri

2. Entegrasyon Katmanı

- **API**
- **HL7 / FHIR**

3. İş Mantığı Katmanı

- Kurallar
- **CDSS**

4. Uygulama Katmanı

- Web
- Mobil

5. Sunum Katmanı

- Kullanıcı arayüzü

12.8. Gerçek Sistem Akışı

Hasta verisi (EHR)
↓
FHIR API
↓
CDSS Motoru
↓
AI Analizi
↓
Uyarı
↓
Kullanıcı

12.9. Performans Gereksinimleri

Sağlık sistemleri: **gerçek zamanlı çalışmalıdır**

Kritik kriterler:

- Yanıt süresi (<2 sn)
- Sistem uptime (%99.9)
- Yük altında performans

12.10. Güvenlik Gereksinimleri

Sağlık verisi: en kritik veri türüdür

Gereklilikler:

- Yetkilendirme
- Kimlik doğrulama
- Şifreleme

Türkiye için: KVKK uyumu zorunlu

12.11. Veri Gizliliği

Teknik çözümler:

- Anonimleştirme
- Maskeleme
- Log kayıtları

12.12. Ölçeklenebilirlik

Sistem: büyüyebilmelidir

Örnek: 100 kullanıcı → 10.000 kullanıcı

Çözüm:

- Cloud mimari
- Mikroservis

12.13. Kullanılabilirlik

Sağlıkta UX: hayat kurtarır

Kriterler:

- Az adım
- Net ekran
- Hata önleyici tasarım

12.14. Teknik Gereksinim Hataları

❌ **1. Entegrasyonu küçümsemek**

❌ **2. Performansı göz ardı etmek**

❌ **3. Güvenliği sonradan düşünmek**

12.15. Yapay Zeka Entegrasyonu

YZ entegrasyonu için:

Gerekenler:

- Veri pipeline
- Model servisleri
- API bağlantıları

12.16. Kritik Mesaj

☞ **İyi analiz + kötü teknik = başarısız sistem**
☞ Entegrasyon yoksa sistem yok

BÖLÜM-13

UYGULAMA YOL HARİTASI: FİKİRDEN CANLI SİSTEME

13.1. Neden Yol Haritası Şart?

Sağlık projelerinde en büyük hata:

✕"Hadi başlayalım" yaklaşımı

✓ Planlı ilerleme

Kritik gerçek:

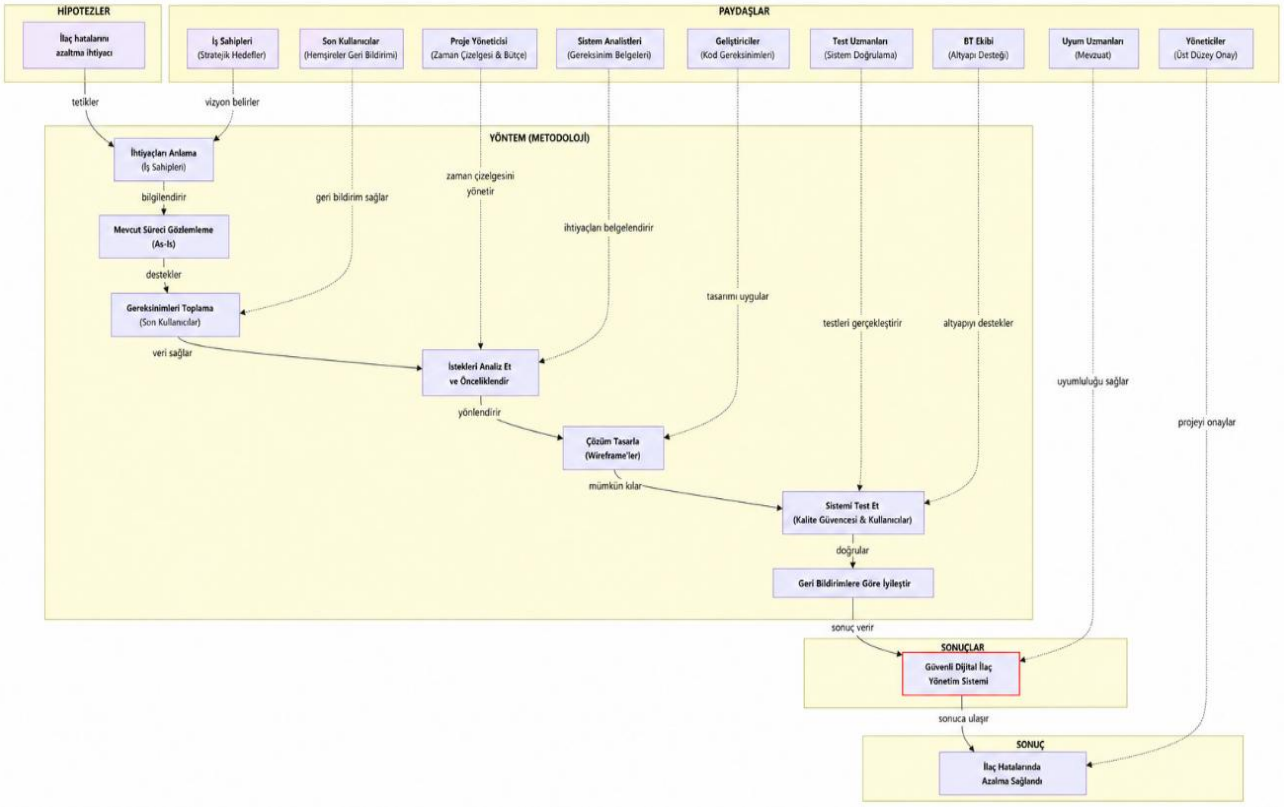
☞ Sağlık projeleri **IT projesi değildir**

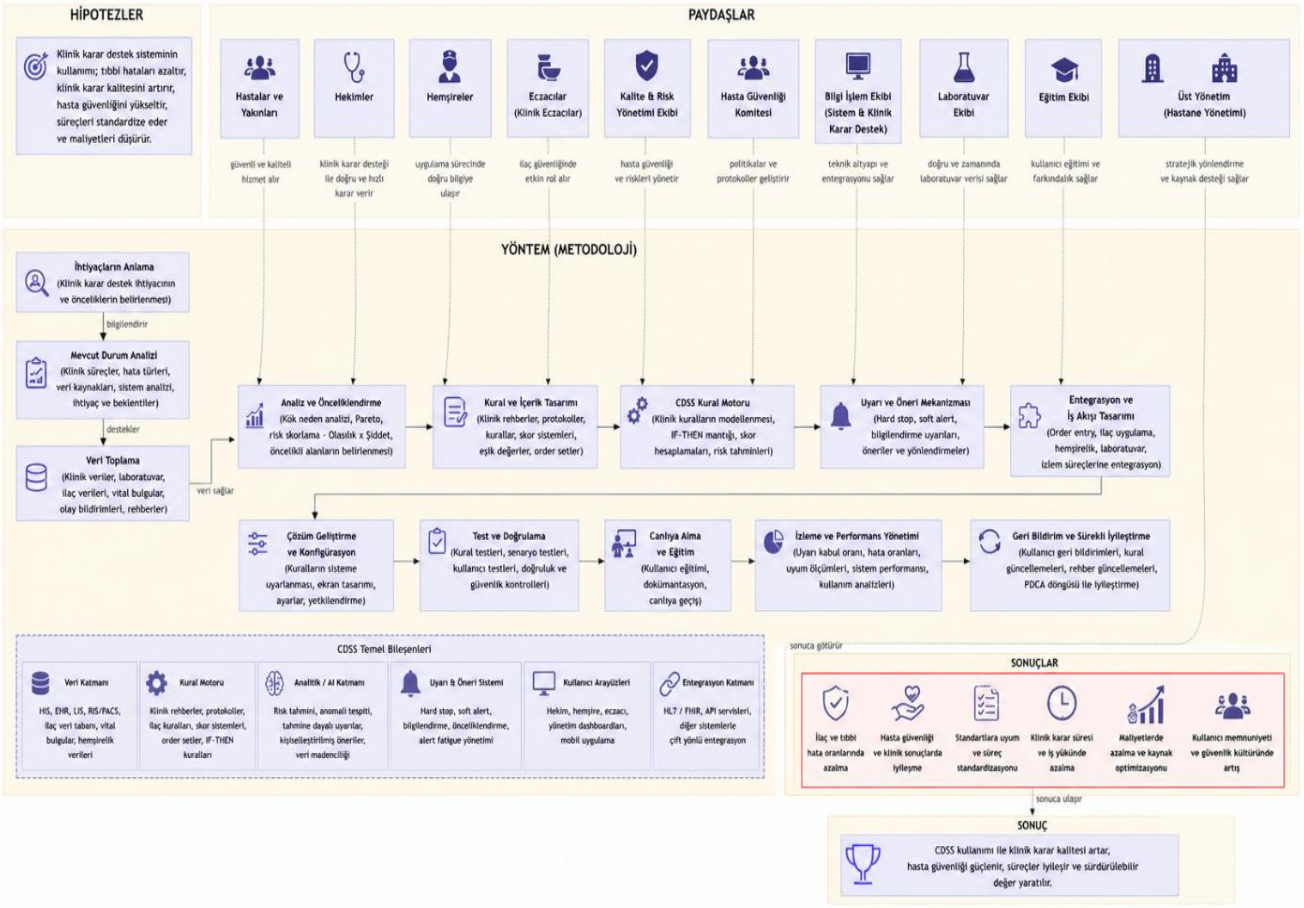
☞ **Klinik dönüşüm projesidir**

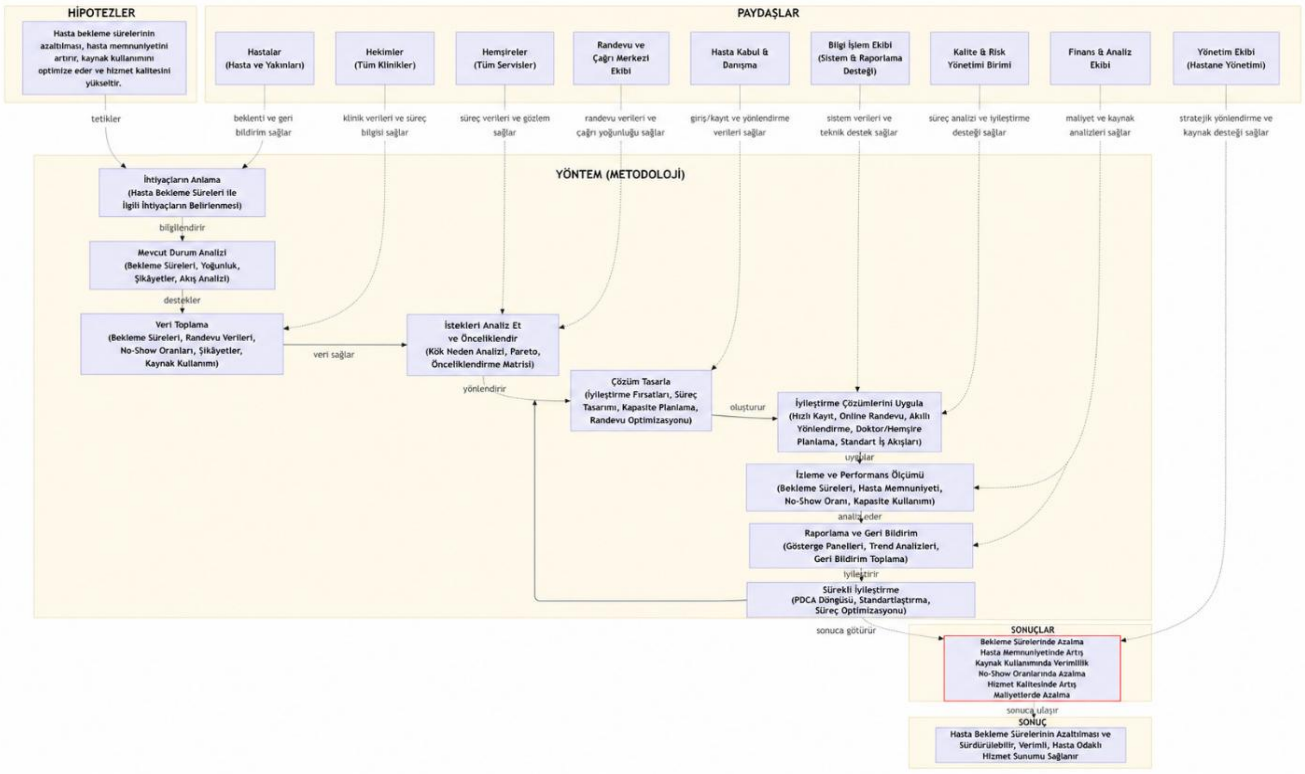
13.2. Genel Uygulama Aşamaları

Bir sağlık bilişimi projesi şu adımlarla ilerler:

Analiz
↓
Tasarım
↓
Geliştirme
↓
Test
↓
Eğitim
↓
Canlıya Geçiş
↓
İzleme


HİPOTEZLER
İlaç hatalarını azaltma ihtiyacı
PAYDAŞLAR
İş Sahipleri (Stratejik Hedefler)
Son Kullanıcılar (Hemşireler Geri Bildirimi)
Proje Yöneticisi (Zaman Çizelgesi & Bütçe)
Sistem Analistleri (Gereksinim Belgeleri)
Geliştiriciler (Kod Gereksinimleri)
Test Uzmanları (Sistem Doğrulama)
BT Ekibi (Altyapı Desteği)
Uyum Uzmanları (Mevzuat)
Yöneticiler (Üst Düzey Onay)
tetikler
vizyon belirler
YÖNTEM (METODOLOJİ)
İhtiyaçları Anlama (İş Sahipleri)
bilgilendirir
Mevcut Süreci Gözlemleme (As-Is)
destekler
Gereksinimleri Toplama (Son Kullanıcılar)
veri sağlar
geri bildirim sağlar
zaman çizelgesini yönetir
ihtiyaçları belgelendirir
tasarımı uygular
testleri gerçekleştirir
altyapıyı destekler
uyumluluğu sağlar
projeyi onaylar
İstekleri Analiz Et ve Önceliklendir
yönlendirir
Çözüm Tasarla (Wireframe'ler)
mümkün kılar
Sistemi Test Et (Kalite Güvencesi & Kullanıcılar)
doğrular
Geri Bildirimlere Göre İyileştir
sonuç verir
SONUÇLAR
Güvenli Dijital İlaç Yönetim Sistemi
sonuca ulaşır
SONUÇ
İlaç Hatalarında Azalma Sağlandı


HİPOTEZLER
Klinik karar destek sisteminin kullanımı; tıbbi hataları azaltır, klinik karar kalitesini artırır, hasta güvenliğini yükseltir, süreçleri standardize eder ve maliyetleri düşürür.
PAYDAŞLAR
Hastalar ve Yakınları
güvenli ve kaliteli hizmet alır
Hekimler
klinik karar desteği ile doğru ve hızlı karar verir
Hemşireler
uygulama sürecinde doğru bilgiye ulaşır
Eczacılar (Klinik Eczacılar)
ilaç güvenliğinde etkin rol alır
Kalite & Risk Yönetimi Ekibi
hasta güvenliği ve riskleri yönetir
Hasta Güvenliği Komitesi
politikalar ve protokoller geliştirir
Bilgi İşlem Ekibi (Sistem & Klinik Karar Destek)
teknik altyapı ve entegrasyonu sağlar
Laboratuvar Ekibi
doğru ve zamanında laboratuvar verisi sağlar
Eğitim Ekibi
kullanıcı eğitimi ve farkındalık sağlar
Üst Yönetim (Hastane Yönetimi)
stratejik yönlendirme ve kaynak desteği sağlar
YÖNTEM (METODOLOJİ)
İhtiyaçların Anlama (Klinik karar destek ihtiyacının ve önceliklerin belirlenmesi)
bilgilendirir
Mevcut Durum Analizi (Klinik süreçler, hata türleri, veri kaynakları, sistem analizi, ihtiyaç ve beklentiler)
destekler
Veri Toplama (Klinik veriler, laboratuvar, ilaç verileri, vital bulgular, olay bildirimleri, rehberler)
veri sağlar
Analiz ve Önceliklendirme (Kök neden analizi, Pareto, risk skorlama - Olasılık x Şiddet, öncelikli alanların belirlenmesi)
Kural ve İçerik Tasarımı (Klinik rehberler, protokoller, kurallar, skor sistemleri, eşik değerler, order setler)
CDSS Kural Motoru (Klinik kuralların modellenmesi, IF-THEN mantığı, skor hesaplamaları, risk tahminleri)
Uyarı ve Öneri Mekanizması (Hard stop, soft alert, bilgilendirme uyarıları, öneriler ve yönlendirmeler)
Entegrasyon ve İş Akışı Tasarımı (Order entry, ilaç uygulama, hemşirelik, laboratuvar, izlem süreçlerine entegrasyon)
Çözüm Geliştirme ve Konfigürasyon (Kuralların sisteme uyarlanması, ekran tasarımı, ayarlar, yetkilendirme)
Test ve Doğrulama (Kural testleri, senaryo testleri, kullanıcı testleri, doğruluk ve güvenlik kontrolleri)
Canlıya Alma ve Eğitim (Kullanıcı eğitimi, dokümantasyon, canlıya geçiş)
İzleme ve Performans Yönetimi (Uyarı kabul oranı, hata oranları, uyum ölçümleri, sistem performansı, kullanım analizleri)
Geri Bildirim ve Sürekli İyileştirme (Kullanıcı geri bildirimleri, kural güncellemeleri, rehber güncellemeleri, PDCA döngüsü ile iyileştirme)
CDSS Temel Bileşenleri
Veri Katmanı
HIS, EHR, LIS, RIS/PACS, ilaç veri tabanı, vital bulgular, hemşirelik verileri
Kural Motoru
Klinik rehberler, protokoller, ilaç kuralları, skor sistemleri, order setler, IF-THEN kuralları
Analitik / AI Katmanı
Risk tahmini, anomali tespiti, tahmine dayalı uyarılar, kişiselleştirilmiş öneriler, veri madenciliği
Uyarı & Öneri Sistemi
Hard stop, soft alert, bilgilendirme, önceliklendirme, alert fatigue yönetimi
Kullanıcı Arayüzleri
Hekim, hemşire, eczacı, yönetim dashboardları, mobil uygulama
Entegrasyon Katmanı
HL7 / FHIR, API servisleri, diğer sistemlerle çift yönlü entegrasyon
sonuca götürür
SONUÇLAR
İlaç ve tıbbi hata oranlarında azalma
Hasta güvenliği ve klinik sonuçlarda iyileşme
Standartlara uyum ve süreç standardizasyonu
Klinik karar süresi ve iş yükünde azalma
Maliyetlerde azalma ve kaynak optimizasyonu
Kullanıcı memnuniyeti ve güvenlik kültüründe artış
sonuca ulaşır
SONUÇ
CDSS kullanımı ile klinik karar kalitesi artar, hasta güvenliği güçlenir, süreçler iyileşir ve sürdürülebilir değer yaratılır.



13.3. Faz 1: Analiz (Foundation): **Bu aşama: Projenin %50'sidir**

Yapılacaklar:

- Gereksinim analizi
- Süreç analizi
- Kök neden analizi

Çıktılar:

- Gereksinim dokümanı
- Süreç haritaları

Risk:

❌ Eksik analiz → tüm proje hatalı

13.4. Faz 2: Tasarım

Amaç: Gereksinimleri sistem tasarımına dönüştürmek

İçerik:

- UI/UX tasarım
- Sistem mimarisi
- Veri akışı

Kritik: Kullanıcı bu aşamada dahil olmalı

13.5. Faz 3: Geliştirme

Yapılanlar:

- Yazılım geliştirme
- Entegrasyon
- API bağlantıları

Model:

- Agile (önerilir)
- Sprint bazlı ilerleme

13.6. Faz 4: Test (Çok Kritik)

Test türleri:

- Fonksiyonel test
- Klinik test
- Kullanıcı testi (UAT)

Kritik kural: Test edilmemiş sistem = risk

13.7. Faz 5: Eğitim : En çok ihmal edilen aşama

Gerçek: Kullanıcı öğrenmezse sistem başarısız olur

İçerik:

- Kullanıcı eğitimi
- Senaryo bazlı eğitim

13.8. Faz 6: Canlıya Geçiş (Go-Live)

Stratejiler:

- Big Bang
- Kademeli geçiş

Risk: Canlıya geçiş = en riskli an

13.9. Faz 7: İzleme ve İyileştirme

Yapılması gerekenler:

- Performans izleme
- Hata takibi
- Kullanıcı geri bildirimi

☞ Sistem canlıya geçince bitmez - **başlar**

13.10. Paydaş Yönetimi

Kimler var?

- Hekim
- Hemşire
- **IT**
- Yönetim

Kritik: Herkesin beklentisi farklı

13.11. Değişim Yönetimi

En büyük direnç:İnsan

Sorun:

- Alışkanlıklar
- Korku
- Güvensizlik

Çözüm:

- Eğitim
- İletişim
- Katılım

13.12. Risk Yönetimi

Tipik riskler:

- Teknik arıza
- Kullanıcı reddi
- Veri hatası

Yaklaşım: Riskleri önceden belirle

13.13. Başarı Kriterleri (KPI)

Klinik:

- Hata oranı
- Hasta güvenliği

Operasyonel:

- Süre
- Verimlilik

Teknik:

- Performans
- Uptime

13.14. Yapay Zeka ile İzleme

YZ sayesinde:

- Anomali tespiti
- Performans analizi
- Sürekli öğrenme

13.15. Gerçek Hayat Dersi

Projeler şu yüzden başarısız olur:

✗ Teknoloji kötü olduğu için değil
✗ İnsan yönetilemediği için

13.16. Kritik Mesaj

☞ **Sistem kurmak kolay, kullandırmak zordur**

☞ Başarı = teknoloji + insan + süreç

BÖLÜM-20

GELECEK PERSPEKTİFİ: YAPAY ZEKA, DİJİTAL İKİZ VE OTONOM SAĞLIK SİSTEMLERİ

14.1. Sağlıkta Dönüşümün Yönü

Sağlık sistemleri şu yönde evriliyor:

☞ Reaktif → Proaktif

☞ Manuel → Otomatik

☞ İnsan odaklı → İnsan + Yapay Zeka

Kritik gerçek: Gelecekte sağlık sistemleri **karar destek değil, karar ortağı** olacak

14.2. Yapay Zeka 2.0: Klinik Karar Ortağı

Bugün:

- Uyarı verir
- Risk gösterir

Gelecekte:

☞ Tedavi önerir

☞ Süreç optimize eder

☞ Klinik kararları şekillendirir

Örnek:

- Sepsis tahmini → erken müdahale
- Düşme riski → önleyici aksiyon

14.3. Dijital İkiz (Digital Twin)

En devrimsel kavramlardan biri: **Dijital ikiz = hastanın sanal kopyası**

Ne yapar?

- Hastayı simüle eder
- Tedavi sonuçlarını önceden gösterir

Örnek:

- “Bu ilaç verilirse ne olur?”
- “Bu ameliyat riski nedir?”

☞ Sistem cevap verir

14.4. Otonom Sağlık Sistemleri

Gelecekte: Sistemler bazı kararları kendisi alacak

Örnek:

- Otomatik ilaç ayarlama
- Akıllı ICU yönetimi

Ama: İnsan her zaman kontrol noktasında olacak

14.5. Sürekli Öğrenen Sistemler

Yeni nesil CDSS: **statik değil, öğrenen sistemler**

Özellik:

- Her veriden öğrenir
- Sürekli gelişir

14.6. Hyper-Personalization

Gelecek: H**er hasta için özel sistem**

Örnek:

- Kişiye özel ilaç
- Kişiye özel risk modeli

14.7. Veri Ekosistemi

Sağlık verisi artık:

- Hastane
- Ev
- Giyilebilir cihaz
- Mobil uygulama

☞ Hepsi birleşir

Sonuç: 360° hasta görünümü

14.8. Yapay Zeka + IoT + CDSS

Bu üçlü birleşiyor:

- IoT → veri üretir
- AI → analiz eder
- CDSS → karar destekler

☞ Ortaya çıkan: **Akıllı sağlık ekosistemi**

14.9. Etik ve Riskler

Gelecek sadece fırsat değil:

⚠ Riskler:

- Veri gizliliği
- AI bias
- Aşırı otomasyon

Kritik soru: "Kararı kim veriyor?"

Cevap: İnsan + sistem birlikte

14.10. Sağlık Profesyonelinin Geleceği

Rol değişiyor:

Bugün:

- Manuel işlem
- Veri girişi

Gelecek:

☞ Analitik düşünme

☞ Sistem yönetimi

☞ Klinik karar

14.11. Yeni Yetkinlikler

Geleceğin uzmanı:

- Veri okuryazarlığı
- AI anlayışı

- Sistem düşüncesi

14.12. Stratejik Perspektif

Sağlık kurumları için: Dijital dönüşüm zorunludur

Rekabet artık:

- Teknoloji
- Veri
- hız

14.13. Büyük Resim

Bu kitap boyunca öğrendiğin her şey: Bu geleceğe hazırlık

Zincir: Gereksinim → Sistem → CDSS → AI → Otonom sistem

14.14. Final Mesajı

☞ **Sağlık bilişimi artık bir seçenek değil**

☞ Bu, geleceğin temelidir

KAPANIŞ

Bu noktada:

✓ Gereksinim analizi
✓ Süreç analizi
✓ CDSS
✓ Yapay zeka
✓ Sistem tasarımı tamamlandı.

Teşekkürler

Katılımınız için teşekkür ederiz.

HEMŞİRELİKTE BİLİŞİM DERNEĞİ
NURSING INFORMATICS ASSOCIATION