Yazılım Proje Yönetimi: Vizyondan Yapı Taşlarına

Gereksinim Mühendisliği ve Kapsam Yönetimi Mimarisi (Hafta 5)



Personalar | MVP | NFR | WBS | INVEST | Scope Creep

Bağlam ve Sorun: Neden Kapsam Yönetimi?

Haftanın Amacı: Görünmez olan yazılımı, ölçülebilir, test edilebilir ve %100 kapsanan mimari parçalara ayırmak.

Standish Group CHAOS 2023 Raporu



Büyük projelerin
%80-85'i
başarısız oluyor veya sorunlu teslim ediliyor.

Küresel Yıllık Zarar:
$450 Milyar

Kök Neden Kötü Kod Değildir: Projeler genellikle kötü mühendislikten değil, belirsiz gereksinimlerden ve yönetilemeyen kapsamdan çöker.

Yazılımın Doğası (Fred Brooks)



Kapsamın Temeli: Ürün Vizyonu ve Hedef Kullanıcı



Ürün Vizyonu (Kuzey Yıldızı)

"Neden bu ürünü yapıyoruz?" Tüm gereksinim kararlarının (kabul/ret) geçirildiği nihai stratejik filtre.

Kullanıcı Personası



Demografik Veriler

- Yaş:
- Rol:
- Konum:
- Teknoloji Kullanımı:

Acı Noktaları (Pain Points)

- ⚠ Zaman Kaybı
- ⚠ Verimsiz İş Akışı
- ⚠ Hatalı Veri Girişi
- ⚠ Kötü Kullanıcı Deneyimi

İhtiyaçlar (Needs)

- ✓ Daha Hızlı Süreçler
- ✓ Otomasyon
- ✓ Doğru Raporlama
- ✓ Sezgisel Arayüz

Personalar: Hedef kitleyi somutlaştıran, veriye dayalı empati araçları.

Kritik Kural: Vizyon ile hizalanmayan veya tanımlı bir personanın problemini çözmeyen hiçbir gereksinim kapsama giremez.

Değerin Özeti: MVP (Minimum Viable Product)

Yanlış Yaklaşım: Sadece Proje Odaklı



Kullanıcı Değeri: Sıfır

Kullanıcı Değeri: Sıfır

Kullanıcı Değeri: Sıfır

Doğru Yaklaşım: MVP Odaklı



MVP'nin Doğası

Minimum özellik setiyle maksimum doğrulanmış öğrenme (validated learning) elde etmektir.

Risk Yönetimi

Yanlış ürünü inşa etme riskini en erken aşamada teşhis eder.

Kapsam Filtresi

'Olmazsa Olmaz' (Must-have) ile 'Olsa İyi Olur' (Nice-to-have) ayrımının matematiğidir.

Terminolojik Ayrım: Ürün Kapsamı vs. Proje Kapsamı

Ürün Kapsamı (Product Scope)

Odak:	Özellikler, fonksiyonlar, nihai değer (Ne üretilecek?)
Ölçüt:	Gereksinimlerin başarıyla karşılanması.
Ana Soru:	Sistem hangi problemi çözüyor?

Başarılı bir proje, Ürün Kapsamını, onaylanmış Proje Kapsamı sınırları içinde teslim eden girişimdir.

Proje Kapsamı (Project Scope)

Odak:	Bütçe, zaman, kaynaklar, kilometre taşları (Nasıl üretilecek?)
Ölçüt:	Proje planının (bütçe/zaman) tutturulması.
Ana Soru:	Bu işi bitirmek için hangi süreçler gerekiyor?

Gereksinimlerin Anatomisi: Fonksiyonel ve NFR

Fonksiyonel Gereksinimler (Sistem Ne Yapmalı?)



Kullanıcı giriş yapabilmeli

Sepete ürün ekleyebilmeli

Ödeme ekranını görebilmeli.

Fonksiyonel Olmayan Gereksinimler - NFR (Sistem Nasıl Davranmalı?)

Performans

Yanıt süresi < 3 saniye

Güvenlik

Uçtan uca şifreleme, KVKK uyumu

Ölçeklenebilirlik

Eşzamanlı 10.000 kullanıcı desteği

Uyarı: Büyük ölçekli yazılım çöküşlerinin %70'i göz ardı edilen görünmez kalite niteliklerinden (NFR) kaynaklanır.

Parçala ve Yönet: İş Kırılım Yapısı (WBS) ve İlkeler

1

WBS (Work Breakdown Structure):

Projenin teslimat odaklı (deliverable-oriented), hiyerarşik olarak daha yönetilebilir parçalara ayrıştırılmasıdır.





2

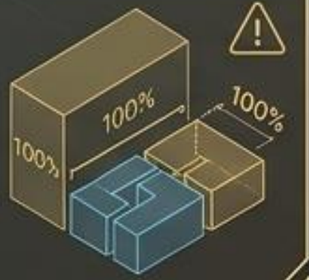
MECE Prensibi:

Alt parçalar birbiriyle asla örtüşmemeli (Mutually Exclusive) ve bütünde hiçbir boşluk/eksik bırakmamalıdır (Collectively Exhaustive).

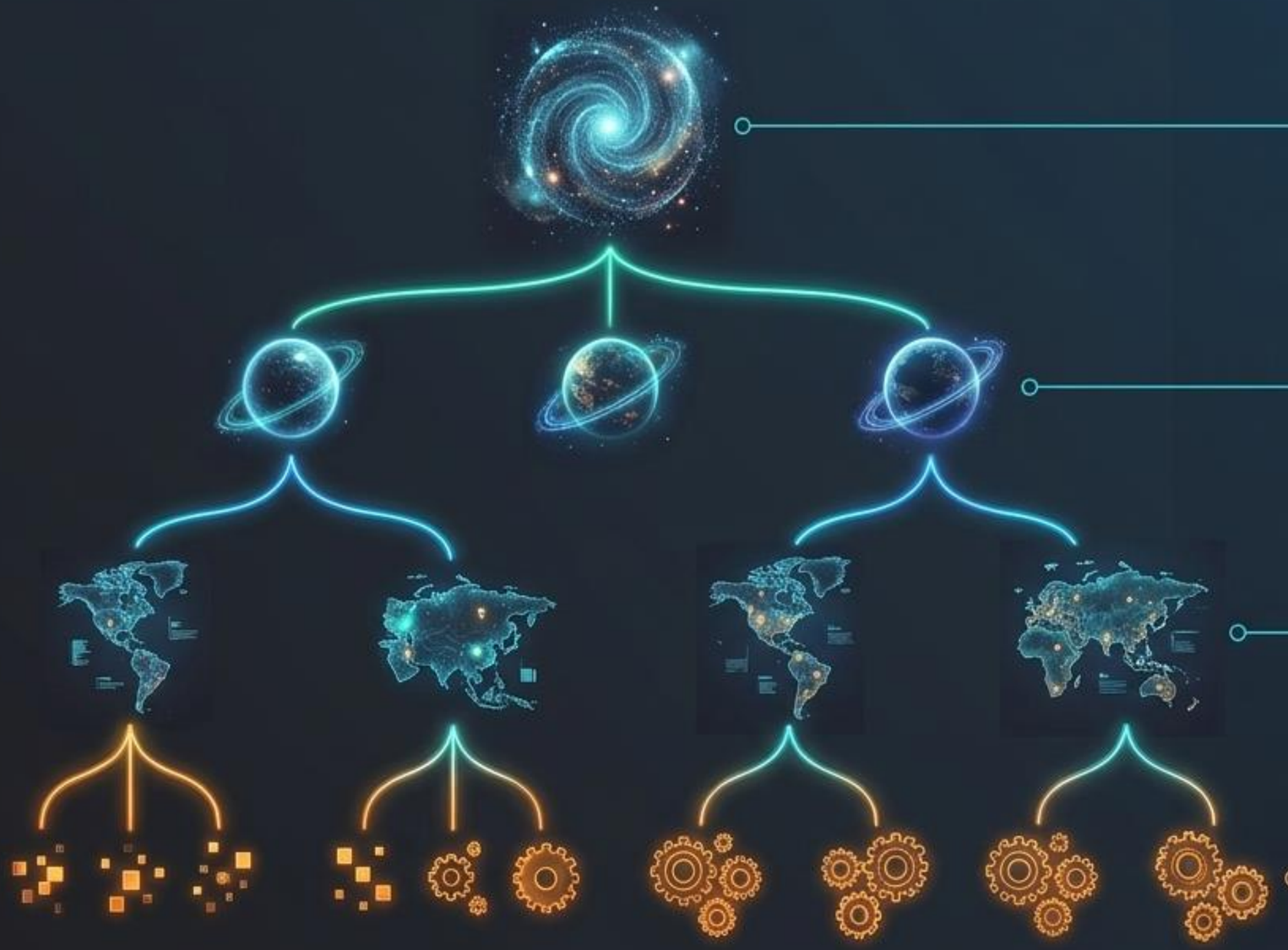


%100 Kuralı:

Bir WBS'teki alt seviye elemanların toplam iş yükü, her zaman ebeveyn (üst) elemanın tam olarak %100'üne eşit olmalıdır. Kapsam dışı hiçbir şey WBS'e giremez; giren her şey %100'ün içinde olmalıdır.



Hiyerarşik Ayrıştırma: Agile Kapsam Ağacı



Epic

Tek bir Sprint'e sığmayacak kadar büyük iş hedefleri.
(Örn: Kullanıcı Yetkilendirme Altyapısı)

Feature

Epic'in mantıksal bir alt kümesi, sürümlenebilir iş paketi.
(Örn: Google ile Giriş Yapma)

User Story

Kullanıcı değerine odaklanan, test edilebilir küçük birim. (Örn: Bir kullanıcı olarak, tek tıkla giriş yapmak istiyorum.)

Task (Sub-task)

Geliştiricinin yazacağı teknik kod/veritabanı işi. (Örn: OAuth2.0 API entegrasyonunun kodlanması.)

Kaliteli Hikaye Yazımı ve INVEST Kriterleri

`As a [role], I want [action] so that [benefit].`

Bir [Kullanıcı Tipi] olarak, [Eylem] yapmak istiyorum,
böylece [Fayda] elde edebilirim.



Independent: Diğer hikayelerden bağımsızdır, bloke edici bağımlılık içermez.

Negotiable: Esnektir, geliştirme ekibiyle müzakereye açıktır (katı sözleşme değildir).

Small: Tek bir Sprint'te tamamlanacak kadar küçük boyutludur.

Valuable: Doğrudan kullanıcıya veya iş modeline somut bir değer katar.

Testable: Hikayenin başarıyla tamamlanıp tamamlanmadığını kesin olarak doğrulayacak net kabul ölçütlerine sahip olmalıdır.

Estimable: Eforu ve karmaşıklığı tahmin edilebilir düzeydedir (Story Points).

Belirsizliği Ortadan Kaldırmak: Kabul Ölçütleri (BDD)

(Behavior-Driven Development)



Bu yapı, hikayenin test edilebilirliğini (T in INVEST) garanti altına alır ve kalite güvencesi (QA) otomasyonunun temelini oluşturur.

Gherkin Dili ile BDD Test Senaryoları

Doğal dil yapısını kullanan, iş analistleri ve yazılımcıların ortak anladığı test senaryosu sözdizimi (syntax).

Temel Anahtar Kelimeler

- **Feature (Özellik):** Test edilen ana modül.
- **Scenario (Senaryo):** Test edilecek spesifik durum.
- **Given (Diyelim ki):** Sistemin başlangıç durumu ve önkoşul.
- **When (Eğer ki):** Kullanıcı veya sistem tarafından yapılan eylem.
- **Then (O zaman):** Eylemin beklenen, ölçülebilir sonucu.
- **And / But (Ve / Fakat):** Çoklu koşulları birbirine bağlayan ek ifadeler.

Dijital Kütüphane Sistemi Örneği

```
Feature: Kitap Rezervasyonu
  Scenario: Başarılı bir şekilde kitap ayırtma işlemi
    Given (Diyelim ki) öğrenci sisteme geçerli
    bilgileriyle giriş yapmış olsun,
    And (Ve) aranılan "Yazılım Proje Yönetimi"
    kitabı kütüphane stoklarında mevcut olsun,
    When (Eğer ki) öğrenci "Rezerve Et" butonuna
    tıklarsa,
    Then (O zaman) sistem öğrenciye bir
    rezervasyon onay numarası göstermelidir,
    And (Ve) ilgili kitabın genel stok sayısını
    anında 1 düşürmelidir.
```

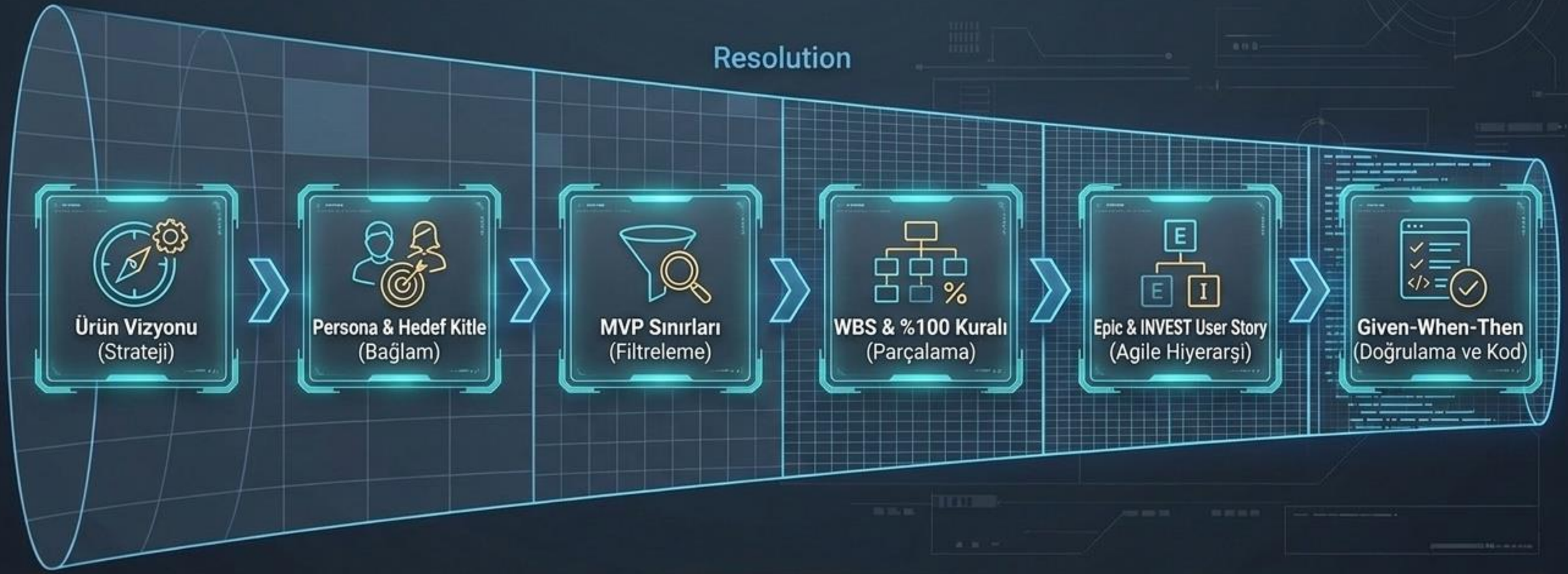
Otomasyon Entegrasyonu

Metin tabanlı bu senaryolar, Cucumber veya SpecFlow gibi araçlar tarafından doğrudan **otomatik test koduna** dönüştürülür.

Cucumber SpecFlow Behave

Kritik Değer: Gherkin, iş birimi (müşteri) ile geliştiriciler/testçiler arasındaki iletişim kopukluğunu **sıfırlayarak** canlıya çıkış öncesi hataları önler.

BÜYÜK SENTEZ: Vizyondan Koda Kapsam Akışı



Kapsam yönetimi, soyut stratejinin sıfır hata toleranslı, ölçülebilir matematiksel bir algoritmaya dönüştürülme sanatıdır.

Kapsam Riskleri: Scope Creep vs. Gold Plating

Kapsam Kayması (Scope Creep)



Kaynak:	Dışarıdan (Müşteri, Paydaş, Yönetim).
Durum:	Sürekli ve kontrolsüz şekilde yeni "küçük" isteklerin projenin sınırlarını esnetmesi.
Risk:	Bütçe tükenmesi ve takvim aşımı.
Önlem:	Sıkı Değişim Kontrol (Change Control) süreçleri.

Altın Kaplama (Gold Plating)



Kaynak:	İçeriden (Geliştirici veya Tasarımcı ekibi).
Durum:	Müşteri istemediği halde "daha havalı" olsun diye ürüne fazladan özellik veya cila eklenmesi.
Risk:	Boşa harcanan kaynak, artan bakım ve test maliyeti.
Önlem:	"Sadece isteneni yap" (MVP) disiplini.

Haftalık Ürün Kılavuzu (Survival Checklist)

Gelecek Projeler İçin Hayatta Kalma Kılavuzu

- [x] **1. Empatiyi Veriyle Sabitleyin:** Personalarınızı varsayımlar üzerine değil, gerçek kullanıcı verileri üzerine kurun.



- [x] **2. MVP Disiplinini Koruyun:** Ürün Vizyonuna uymayan her fantastik özelliğe şüphesiz 'Hayır' deyin.



- [x] **3. Çatlak Bırakmayın:** WBS'i %100 kuralına ve MECE prensibine sadık kalarak, atlanmış hiçbir iş bırakmadan oluşturun.



- [x] **4. Değer Odaklı İletişim Kurun:** User Story'leri teknik jargonla değil, kullanıcının elde edeceği iş değeriyle yazın (INVEST uyumlu).



- [x] **5. Sınırları Savunun:** Stakeholder analizini güncel tutarak Kapsam Kaymasına (Creep) ve geliştirici egosu kaynaklı Altın Kaplamaya (Gold Plating) karşı uyanık olun.





"Başarılı bir yazılım projesi **vizyonun gücünden** beslenir; ancak **kapsamın netliği, gereksinimlerin titizliği** ve **sınırların korunmasıyla** inşa edilir."

Hatırlatma: Mimariyi parçalara doğru ayıramazsanız, görünmez olan yazılım eninde sonunda bütçenizi ve takviminizi yutacaktır.



Gelecek hafta görüşmek üzere.