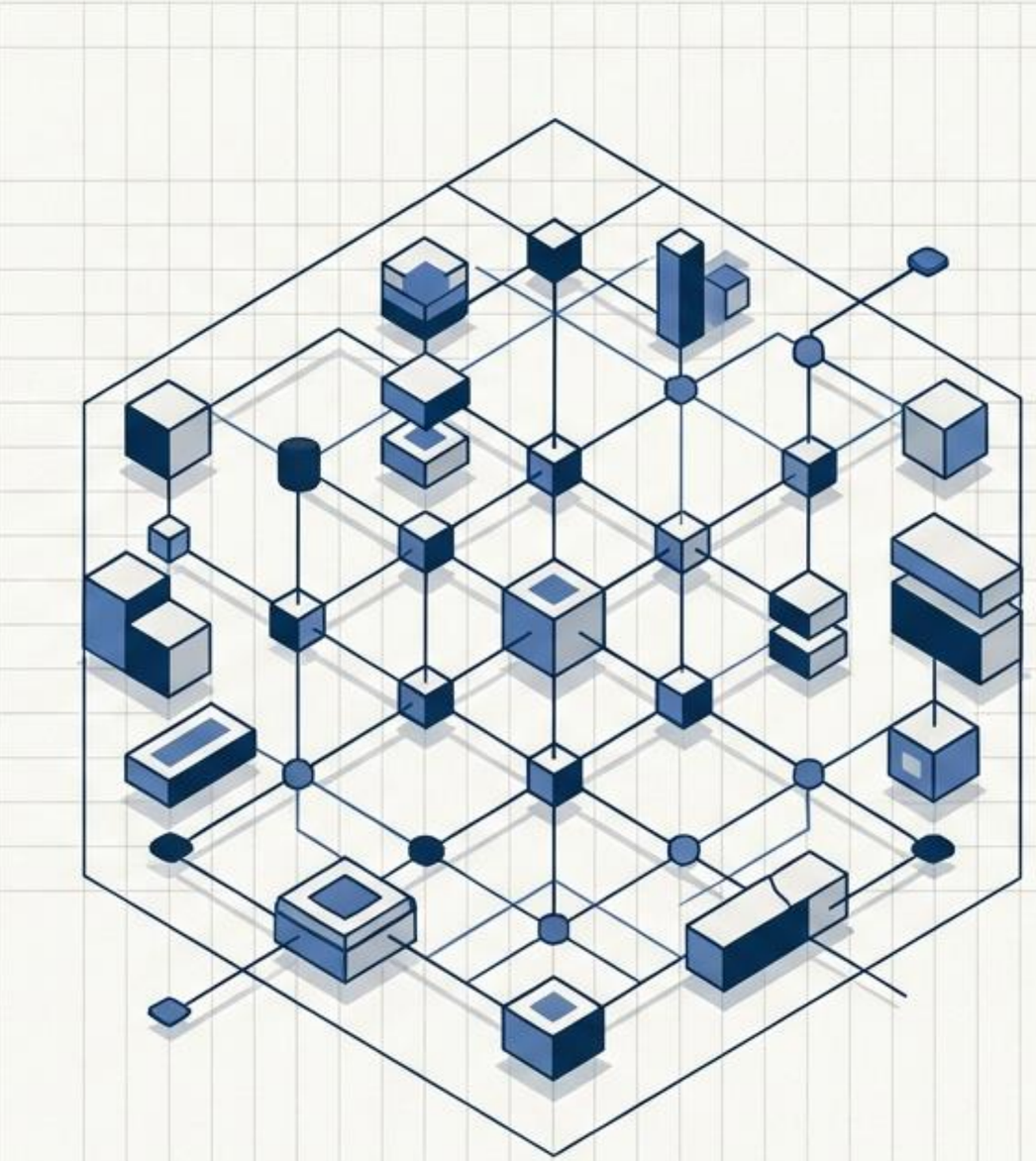
# Yazılım Proje Yönetimi: 1. Hafta

Giriş ve Temel Kavramlar

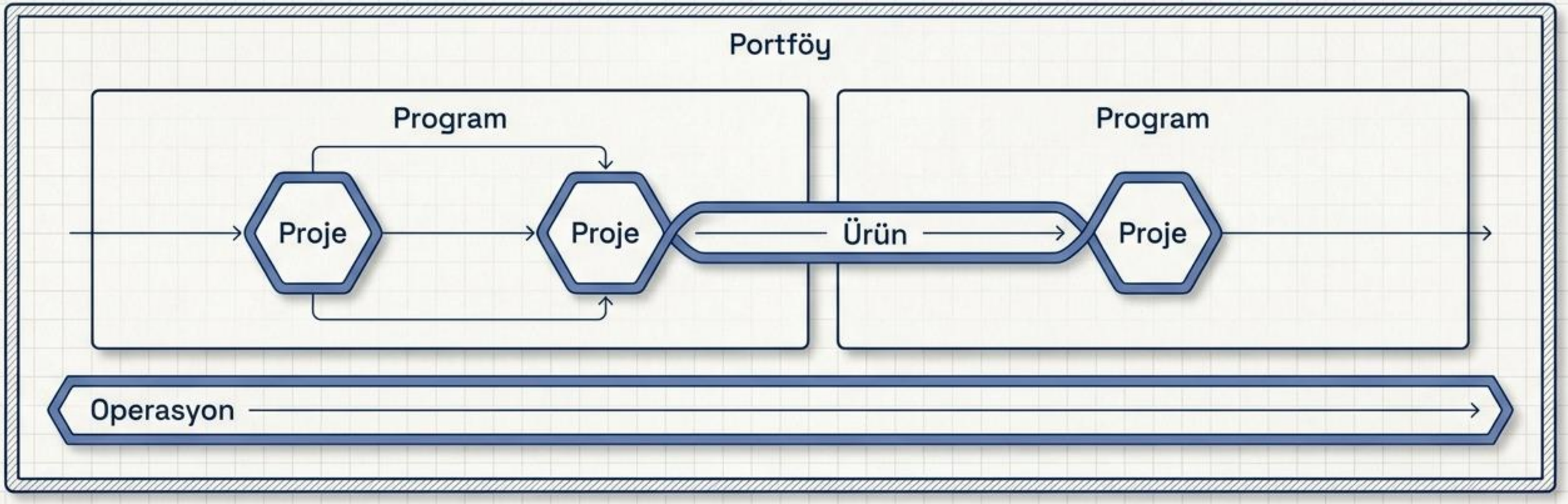
Proje, Portföy, Riskler ve Paydaş Analizine Detaylı Bir Bakış



# Yönetimin Temel Varlıkları (The 4 P's of Software Project Management)

- **İnsan (People):** Projenin en kritik varlığıdır. Yazılım, yaratıcı ve zihinsel bir süreç olduğu için geliştirici ekibin yetkinliği, motivasyonu, ekip içi iletişimi ve paydaşların (müşteri, son kullanıcı, sponsor) yönetimi teknik süreçlerden bile önce gelir.
- **Ürün (Product):** Geliştirilmekte olan yazılımın kendisi ve ulaşılmak istenen hedeftir. Ürünün kapsamı (scope), gereksinimleri ve sınırları net bir şekilde belirlenmelidir.
- **Süreç (Process):** İnsanların ürünü ortaya çıkarırken izledikleri yoldur (Çevik, Şelale, Spiral vb.). Süreç, karmaşıklığı yönetilebilir adımlara böler ve ekibin senkronize çalışmasını sağlayan bir çerçeve sunar.
- **Proje (Project):** Belirli bir bütçe, zaman ve kaynak kısıtı içinde ürünün ortaya çıkarılması için yapılan planlı faaliyetlerin bütünüdür. Risk yönetimi, maliyet tahmini ve takvim oluşturma bu varlığın yönetimiyle ilgilidir.

# Proje Yönetiminin 5 Temel Taşı




## Proje

Belirli bir başlangıcı ve sonu olan, özgün bir ürün/hizmet üretmek için üstlenilen geçici girişim.

## Program

Tek başına yönetildiğinde elde edilemeyecek ekstra faydalar sağlamak için koordine edilen ilişkili projeler grubu.

## Portföy

Kurumun stratejik iş hedeflerine ulaşmak için yönetilen tüm projeler, programlar ve operasyonlar bütünü.

## Ürün

Belirli bir bitiş tarihi olmayan, yaşam döngüsü boyunca sürekli iyileştirilen değer odağı.

## Operasyon

Kurumun sürekliliğini sağlayan, tekrarlayan ve kalıcı iş süreçleri.

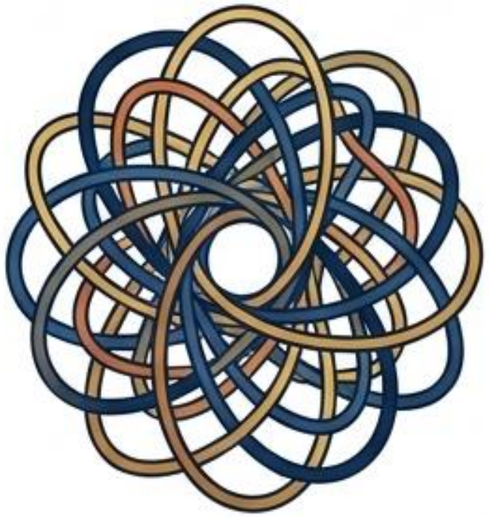
Gemini Notebook

# Matris: Proje, Program ve Portföy Karşılaştırması

| | Proje | Program | Portföy |
|---|---|---|---|
| Odak (Focus) | Kapsam ve Teslimat | Karşılıklı Bağımlılıklar ve Fayda | Stratejik Uyum ve Yatırım Getirisi (ROI) |
| Başarı Kriteri (Metric) | Bütçe, Zaman ve Kalite hedeflerine ulaşmak | İstenen iş faydasını ve verimliliği sağlamak | Kurumsal hedeflerin gerçekleşmesi |
| Kapsam (Scope) | Tanımlı ve sınırlandırılmış | Projelerin toplamından daha geniş | Organizasyonun tüm vizyonu |

Proje yöneticisi işi doğru yapar. Portföy yöneticisi doğru işi yapar.

# ‘No Silver Bullet’: Fred Brooks’un 4 Temel İlkesi



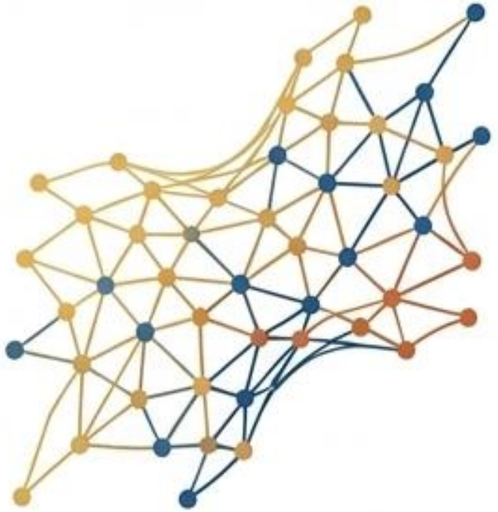
## Karmaşıklık

Boyutuna oranla çok sayıda benzersiz, doğrusal olmayan durum ve etkileşim içerir.



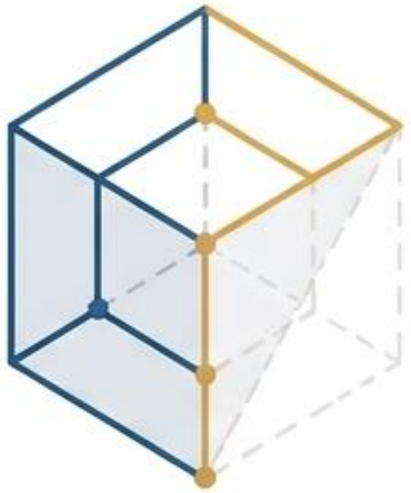
## Uyumluluk

Mevcut donanım, eski işletim sistemleri ve keyfi insan süreçleri gibi dış kurallara uymak zorundadır.



## Değişebilirlik

Somut bir kütlesi olmadığı için sürekli değişime, yeni taleplere ve esnemeye açıktır.



## Görünmezlik

Geometrik veya uzamsal bir varlığı yoktur; yazılımı tam anlamıyla görselleştirmek imkansızdır.

# Değer Hiyerarşisi

- **Stratejik Değer (Vizyon):** Kurumun veya müşterinin nihai amacı (Örn: Pazar payını artırmak, operasyonel maliyetleri %20 düşürmek).
- **Proje/Ürün Değeri:** Stratejik vizyona hizmet edecek yazılımın sağlayacağı doğrudan fayda (Örn: Müşterilerin kendi işlemlerini yapabileceği bir portal sunmak).
- **Özellik Değeri (Feature Value):** Sistemdeki belirli bir modülün kullanıcıya sunduğu yetenek (Örn: Sepette unutulan ürünler için otomatik e-posta hatırlatıcısı).
- **Teknik Değer:** Sürdürülebilirlik, ölçeklenebilirlik, temiz kod mimarisi ve düşük teknik borç gibi geliştirme ekibi için anlam ifade eden ancak uzun vadede ürünün yaşaması için şart olan altyapısal değer.

# Projelerin Başarısızlık Nedenleri

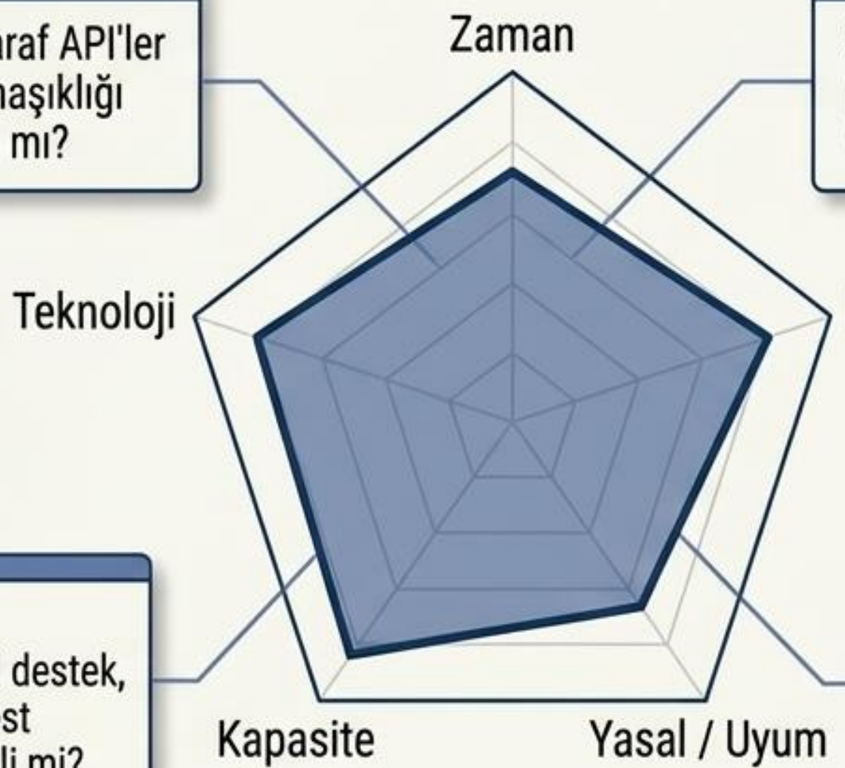
| | |
|---|---|
| Kötü Tanımlanmış Gereksinimler | Müşterinin ne istediğini tam bilmemesi veya ekibin müşteriyi yanlış anlaması. En yaygın başarısızlık nedenidir. |
| **Kapsam Kayması (Scope Creep)** | Proje başladıktan sonra, bütçe ve takvim güncellenmeden sürekli yeni özelliklerin (feature) projeye dahil edilmesi. |
| **Gerçekçi Olmayan Planlama** | Aşırı iyimser zaman ve maliyet tahminleri yapılması (Planning Fallacy). Geliştiricilerin işin zorluğunu hafife alması. |
| **Zayıf Paydaş İletişimi** | Geliştirme ekibi ile iş birimleri (domain uzmanları) arasında ortak bir dilin olmaması. Son kullanıcıların sürece çok geç dahil edilmesi. |
| **Teknik Borcun Göz Ardı Edilmesi** | Projeyi hızlı teslim etmek adına mimari standartlardan taviz verilmesi (Örn: SOLID prensiplerinin ezilmesi). Bu durum ilerleyen safhalarda projeyi kilitler. |
| **Risk Yönetiminin Olmaması** | Kilit bir personelin işten ayrılması, seçilen 3. parti bir kütüphanenin yetersiz kalması gibi risklerin öngörülüp B planlarının hazırlanmaması. |

# Yazılım Fizibilitesi: Başarısızlığı Kod Yazmadan Önce Yakalamak

**Teknoloji:** 3. taraf API'ler ve sistem karmaşıklığı doğru anlaşıldı mı?

**Zaman:** Pazar baskısı ile mühendislik gerçekliği uyuşuyor mu?

**Bütçe:** Kapsam genişlemesine karşı görünmez maliyetler bütçelendi mi?




**Kapasite:** Organizasyonel destek, geliştirme ve test uzmanlığı yeterli mi?

**Yasal / Uyum:** KVKK, GDPR ve veri gizliliği standartları karşılanıyor mu?

Temel Çıkarım: Fizibilite tek bir 'yapabilir miyiz?' sorusu değil, çok boyutlu bir risk ve kısıt değerlendirmesidir.

# Doğru Projeyi Seçmek İçin Puanlama Modelleri

- Organizasyonların kısıtlı kaynaklarını (zaman, bütçe, mühendislik eforu) en yüksek iş değerini üretecek projelere yönlendirmesini sağlayan analitik karar destek araçlarıdır. Subjektif tercihleri ortadan kaldırarak şeffaf ve matematiksel bir seçim süreci sunarlar.
- Yazılım projelerinin doğası gereği barındırdığı soyutluk ve teknik karmaşıklık, bu değerlendirmelerin çok boyutlu yapılmasını zorunlu kılar.
  - **Ağırlıklı Puanlama Modeli (Weighted Scoring Model)**
  - **Finansal Puanlama Modelleri**
  - **Analitik Hiyerarşi Süreci (AHP - Analytic Hierarchy Process)**
  - **Risk / Değer Matrisi (Risk/Value Matrix)**

# Ağırlıklı Puanlama Modeli (Weighted Scoring Model)

- En yaygın ve esnek yaklaşımdır. Her projenin stratejik hedeflere, finansal beklentilere ve teknik kapasiteye ne kadar uygun olduğu belirlenen kriterlere göre puanlanır.

# Finansal Puanlama Modeli

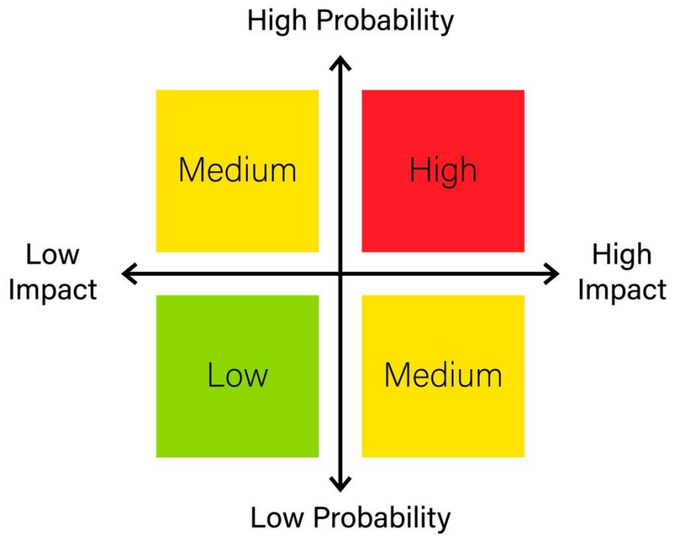
- Projelerin ekonomik verimliliğine odaklanan nicel modellerdir. Genellikle tek başlarına değil, ağırlıklı puanlama modelinin alt kriterleri olarak kullanılırlar.
  - **Net Bugünkü Değer (NPV):** Projenin gelecekteki tüm nakit akışlarının (sunucu maliyetleri, lisans giderleri, adam-ay eforları vs.) bugünkü değerini hesaplar. NPV'si sıfırdan büyük olan ve en yüksek değere sahip proje seçilir.
  - **Yatırım Getirisi Oranı (ROI):** (Net Kar / Toplam Yatırım Maliyeti) x 100 formülüyle hesaplanır.
  - **Geri Ödeme Süresi (Payback Period):** Yatırımın kendini ne kadar sürede amorti edeceğini ölçer. Yazılım ve teknoloji dünyasında sistemlerin hızlı eskimesi (obsolescence) nedeniyle, geri ödeme süresinin kısa olması (örneğin 12-18 ay) kritik bir eleyici kriterdir.

# Analitik Hiyerarşi Süreci (AHP - Analytic Hierarchy Process)

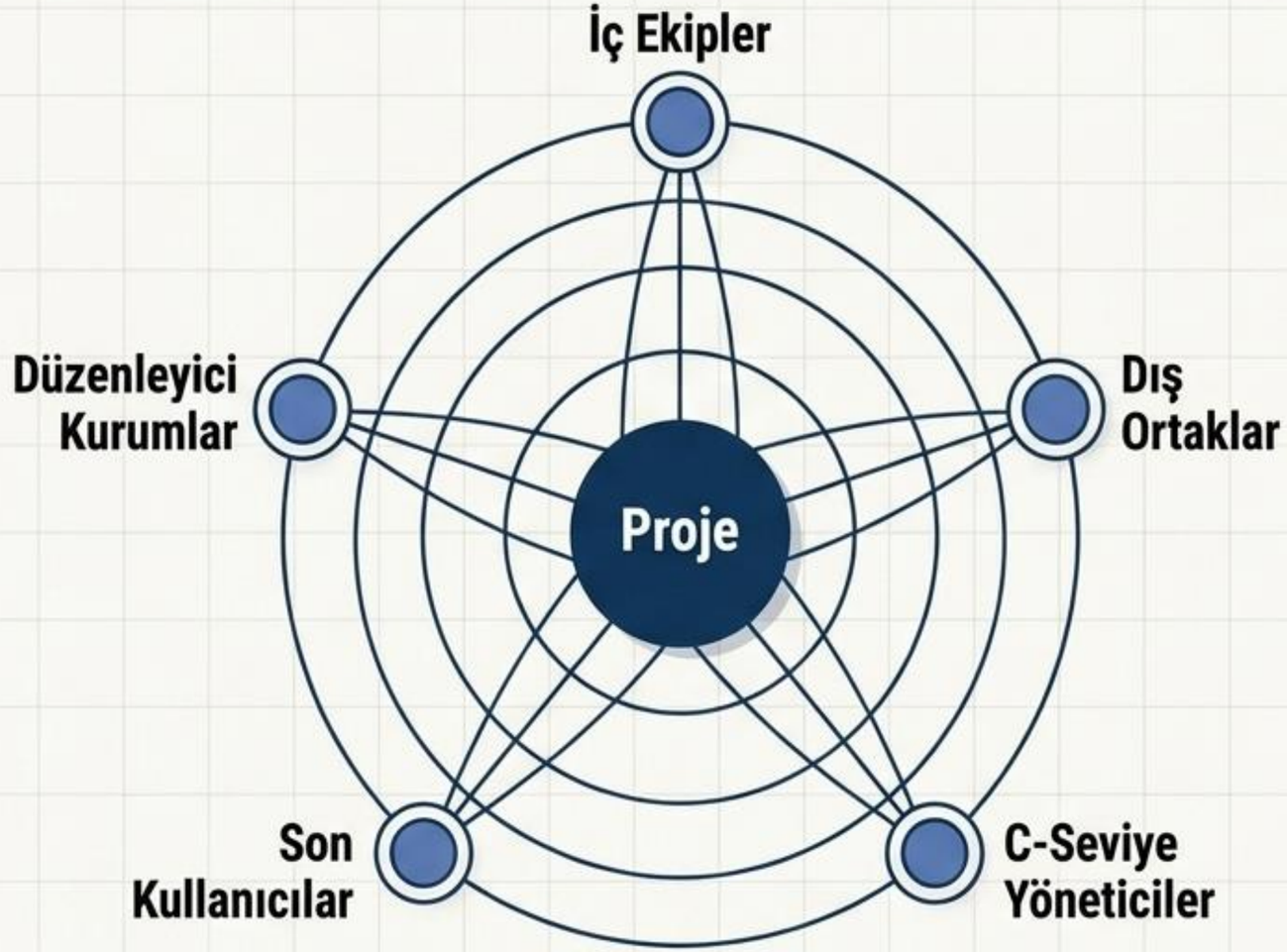
- Thomas Saaty tarafından geliştirilen bu model, birbirine zıt veya ölçülmesi zor kriterlerin (örneğin, "Kullanıcı Deneyimi" ile "Güvenlik Altyapısı" arasındaki denge) bulunduğu karmaşık kararlar için kullanılan matematiksel olarak titiz bir yöntemdir.
  - Kriterler kendi aralarında ikili karşılaştırmalara (pairwise comparison) tabi tutulur. Karar vericiler "Güvenlik, performanstan ne kadar daha önemlidir?" gibi soruları 1-9 ölçeğinde yanıtlar.
  - Matris hesaplamaları (özdeğer ve özvektörler) kullanılarak kriterlerin ağırlıkları tamamen matematiksel bir tutarlılıkla (Consistency Ratio) çıkarılır.
  - Büyük ölçekli kurumsal mimari dönüşümlerinde (örneğin, monolitik bir yapıdan mikroservis mimarisine geçiş projelerinin değerlendirilmesinde) son derece etkilidir.

# Risk / Değer Matrisi (Risk/Value Matrix)

- Projeleri görselleştirmek, portföy dengesi kurmak ve üst yönetime sunmak için kullanılan iki boyutlu bir modeldir. Yatay eksene "Risk", dikey eksene ise "İş Değeri" yerleştirilir.



# İnsan Faktörü: Paydaş Analizi Nedir?




Projeler kötü koddan dolayı değil, **gücü elinde bulunduran** ve projenin ona **zarar vereceğini düşünen** biri yüzünden iptal edilir.

## Analiz Kuralları

1. Unvana göre değil, gerçek etki ve davranışa göre analiz yapın.
2. Sadece destekçilere değil, direnç gösterebilecek gizli engellere odaklanın.

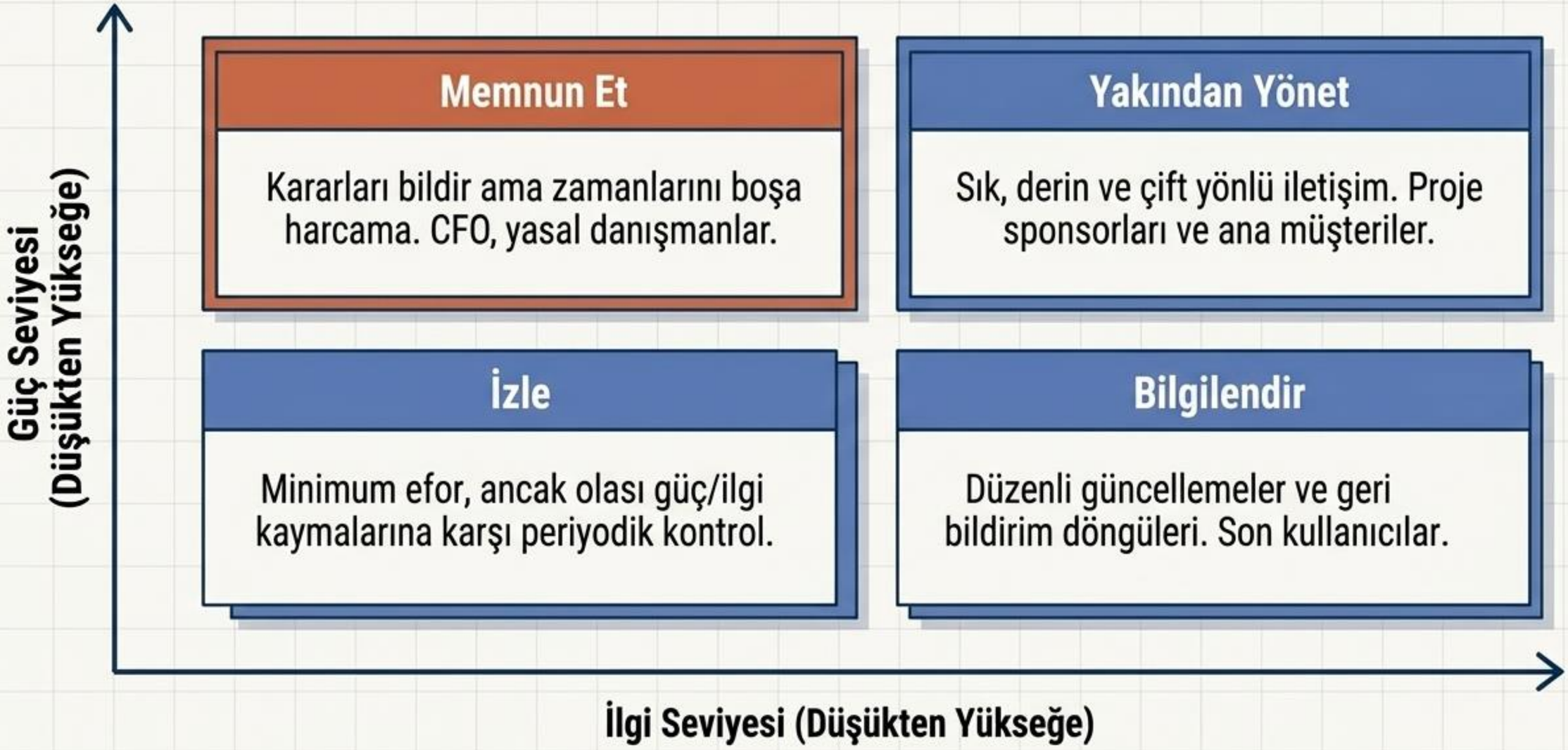
# Mendelow Matrisi (Güç–İlgi Izgarası / Power-Interest Grid)

- 1991 yılında Aubrey Mendelow tarafından geliştirilen ve projelerdeki paydaşları **Etki Gücü (Power)** ile **İlgi Düzeyi (Interest)** eksenlerinde sınıflandıran stratejik bir haritalama aracıdır. Yazılım projeleri sadece kod kalitesi veya bütçe yetersizliği nedeniyle başarısız olmaz; sıklıkla karar yetkisine sahip paydaşların beklentilerinin yönetilememesi veya görünmeyen direnç noktaları nedeniyle durdurulur.

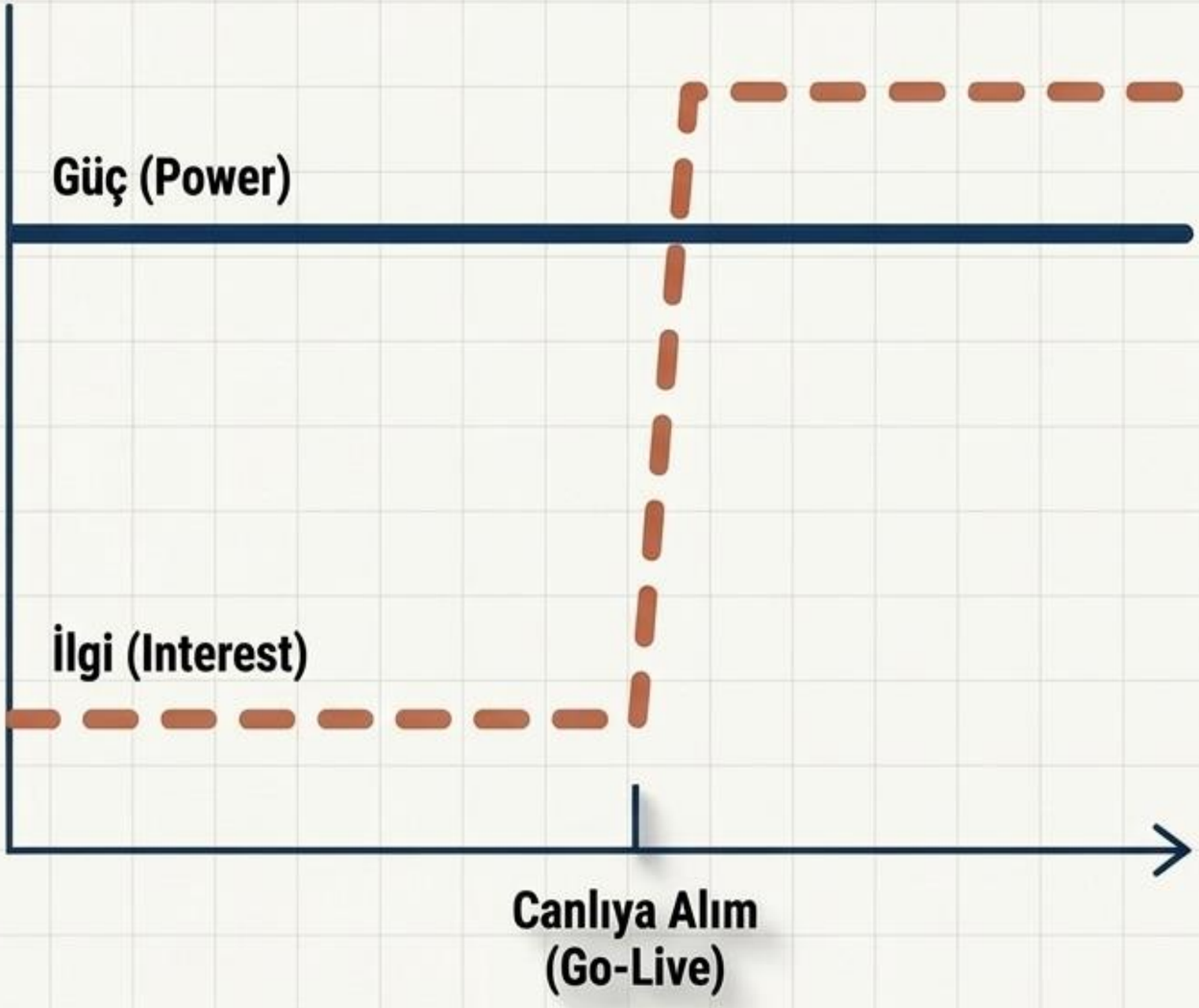
# Adım Adım Mendelow Matrisi Uygulama Metodolojisi

- **1. Paydaşların Belirlenmesi:** Proje Başlatma Belgesi (Project Charter), organizasyon şeması ve sponsor görüşmelerinden yararlanılarak projeden etkilenen tüm kişiler listelenir (Orta ölçekli projeler için 15-40 kişi optimumdur).
- **2. Güç ve İlgi Seviyelerinin Değerlendirilmesi:** Her paydaş için bağımsız iki puanlama yapılır. Göreceli sıralama için 0-10 ölçeği veya Düşük / Orta / Yüksek derecelendirmesi kullanılır.
- **3. Matrise Yerleştirme ve Tutum Seviyesinin Eklenmesi:** Paydaşlar matristeki çeyreğe yerleştirilir. Derinlik katmak için paydaşın projeye yaklaşımı olumlu ise (+) işareti, olumsuz/dirençli ise (-) işareti eklenir.
- **4. İletişim Eylemlerinin Tanımlanması:** Her çeyrek için iletişim kanalı (Slack, e-posta, 1:1, demo), güncelleme sıklığı ve rapor tipi belirlenir.
- **5. İzleme ve Faz Kapılarında (Stage Gates) Gözden Geçirme:** Matris statik değildir. Her proje fazı sonunda paydaşların güç ve ilgi değişimleri gözden geçirilerek matris güncellenir.

# Mendelow’un Güç-İlgi Matrisi



# Tehlike Bölgesi: Neden ‘Memnun Et’ Çeyreği Çeyreği Projeleri Öldürür?




## Senaryo

Sessiz projelerin öldüğü yer burasıdır. Yüksek güçlü, düşük ilgisi olan bir paydaş (örn. BT Güvenlik Direktörü) zararsız görünür. Ancak proje aniden onların umursadığı bir protokole dokunduğunda ilgileri kriz anında zirveye fırlar.

## Çıkarım

Paydaş analizi tek seferlik bir belge değildir. Sürekli analiz, bu “ilgi sıçramalarını” krizden önce yakalamayı sağlar.

# RACI Matrisi

- Bir yazılım projesinde "Herkes birinin yapacağını düşündü ama kimse yapmadı" sendromunu ortadan kaldıran, yetki ve sorumluluk sınırlarını kesin çizgilerle çizen bir görev dağılım çerçevesidir.
  - **R - Responsible (Sorumlu / İşi Yapan):** İşi fiilen yerine getiren, kodu yazan, testi koşan veya belgeyi hazırlayan kişi ya da ekiptir. Bir görevin birden fazla "R" rolü olabilir.
  - **A - Accountable (Hesap Verebilir / Onay Merci):** İşin doğru, standartlara uygun ve zamanında yapılmasından nihai olarak sorumlu olan kişidir. Kararı o verir, "iş bitti" onayını o imzalar. RACI'nin Altın Kuralı: Bir görevin sadece ve sadece BİR tane 'A'sı olabilir. İki 'A' olursa karar süreçleri tıkanır ve çatışma çıkar.
  - **C - Consulted (Danışılan):** İş yapılmadan önce fikrine, deneyimine veya teknik uzmanlığına başvurulan kişidir. Karşılıklı (çift yönlü) bir iletişim söz konusudur. (Örn: Yeni bir modül tasarlanırken siber güvenlik uzmanına danışılması).
  - **I - Informed (Bilgilendirilen):** Karar alındıktan veya görev tamamlandıktan sonra sonuçtan haberdar edilen kişidir. Fikri sorulmaz, tek yönlü bir bildirim yapılır. (Örn: Sistem canlıya alındığında son kullanıcılara e-posta gönderilmesi).

# Akıllı Yemekhane Sistemi Projesi

| Proje Görevi | Proje Yöneticisi | Yazılım Geliştirici | Veritabanı Uzmanı (DBA) | Yemekhane Müdürü (Müşteri) |
|---|---|---|---|---|
| **IEEE SRS (Gereksinim) Hazırlığı** | A | C | I | R, C |
| **Veritabanı Mimarisinin Kurulması** | I | C | A, R | I |
| **Ödeme API'sinin Entegrasyonu** | A | R | C | I |
| **Kullanıcı Kabul Testleri (UAT)** | A | I | I | R |

# Modellerin Karşılaştırması: Güç-İlgi Matrisi vs. RACI

## Panel A: Güç-İlgi Matrisi

- **Odak:** Stratejik Katılım Planlaması (Kimi, nasıl yönetmeliyim?)
- **Soru:** İletişim stratejim ne olmalı?
- **Zayıflık:** Proje içi günlük görev atamalarını belirlemez.

## Panel B: RACI Matrisi

- **Odak:** Rol ve Sorumluluk Netliği (Kim, ne yapıyor?)
- **Soru:** Bu görevin Sorumlusu (R), Hesap Verebiliri (A), Danışılanı (C) ve Bilgilendirileni (I) kim?
- **Zayıflık:** Politik ve dış paydaşları kapsamaz.

**Özet:** Bir sponsor Güç-İlgi matrisinde ‘Yakından Yönet’ statüsünde olabilir, ancak RACI’de sadece ‘Hesap Verebilir (A)’ rolündedir. İkisi birbirini tamamlar, yerine geçmez.

# Dağıtık Ekiplerde İletişim Planı

| Paydaş | İhtiyaç Duyulan Bilgi | Kanal | Sıklık | Sorumlu |
|---|---|---|---|---|
| Geliştiriciler | Günlük durum, engeller | Slack, Jira | Günlük | Proje Yöneticisi |
| Sponsor | Bütçe, aşama sağlığı | E-posta, Zoom | İki Haftada Bir | Portföy Yöneticisi |
| Son Kullanıcılar | Yeni özellikler, kesintiler | Bülten, Portal | Aylık | Ürün Yöneticisi |

**Altın Kural:** Toplantıları varsayılan olmaktan çıkarın! Senkron (gerçek zamanlı) iletişim sadece kararlar ve acil durumlar içindir. Asenkron iletişim, durum güncellemeleri ve odak gerektiren işler için kullanılmalıdır.

# Jira ve Slack

- **Jira:** Bir proje yönetimi ve iş/hata takibi yazılımıdır. Ekipler, yapılacak işleri, yazılım hatalarını ve müşteri taleplerini Jira üzerinde dijital görev kartları halinde oluşturur. Çevik yönetimdeki Scrum ve Kanban tahtaları burada barındırılır. Bir görevin kime atandığı, teknik gereksinimleri, teslim tarihi ve mevcut durumu (Yapılacak, Devam Ediyor, Test Aşamasında, Bitti) Jira üzerinden ölçülür ve takip edilir.
- **Slack:** İş odaklı bir anlık mesajlaşma ve asenkron iletişim platformudur. İletişim, karmaşık e-posta zincirleri yerine belirli projelere, departmanlara veya konulara göre ayrılmış "Kanallar" üzerinden yürütülür. Ekip üyeleri bu kanallarda dosya paylaşabilir, kod parçacıkları gönderebilir ve eşzamanlı veya eşzamansız olarak haberleşebilir.
- Genel kullanımda Jira işin kendisinin, aşamalarının ve resmi kaydının tutulduğu sistemken; Slack o iş hakkında ekibin koordine olduğu ve tartıştığı iletişim alanıdır. İki platform genellikle birbirine entegre çalışır; örneğin Jira'da yeni ve acil bir görev açıldığında veya bir görev tamamlandığında, ilgili Slack kanalına otomatik olarak uyarı mesajı düşer.

# 1. Hafta Sentezi: Proje Başlatma Araç Kutusu

## 1. Ekosistemi Tanı

Projenin, Portföy ve Ürün stratejisi içindeki yerini net olarak haritalandırın. Doğru işi yaptığınızdan emin olun.

## 2. Fizibilite ve Risk

'Neden başarısız oluruz?' kök nedenlerini analiz edin ve 2x2 Değer/Risk puanlama matrisiyle yatırımı doğrulayın.

## 3. Süreç Geçitlerini Kur

Aşama-Geçit (Stage-Gate) metodolojisiyle duygusal kararları engelleyin. Her aşama sonunda net bir Devam/Tamam kontrolü yapın.

## 4. İnsanları Yönet

Mendelow matrisi ile paydaşları haritalayın, güç/ilgi değişimlerini izleyin ve dağıtık ekipler için net bir asenkron iletişim planı kurgulayın.

# Kaynaklar


- Brooks, F. P. (1987) & SEBoK Wiki: *No Silver Bullet: Essence and Accidents of Software Engineering* / *The Nature of Software* (IEEE Computer & IEEE/SEBoK Wiki).
- Lehtinen, T. O. A., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014): *Perceived Causes of Software Project Failures – An Analysis of Their Relationships*, Information and Software Technology (Elsevier), 56(6), 623–643.
- Joubert, S. / Northeastern University (2024): *Project, Program, and Portfolio Management: Key Differences*, Northeastern University Graduate Knowledge Hub.
- Firmbee: *4Ps of Management: Project, Product, Program and Portfolio*.
- Requiment Analysis: *Why Do Software Development Projects Fail?