

B Ö L Ü M

6

Yazılım Geliştirme Yaşam Döngüsü

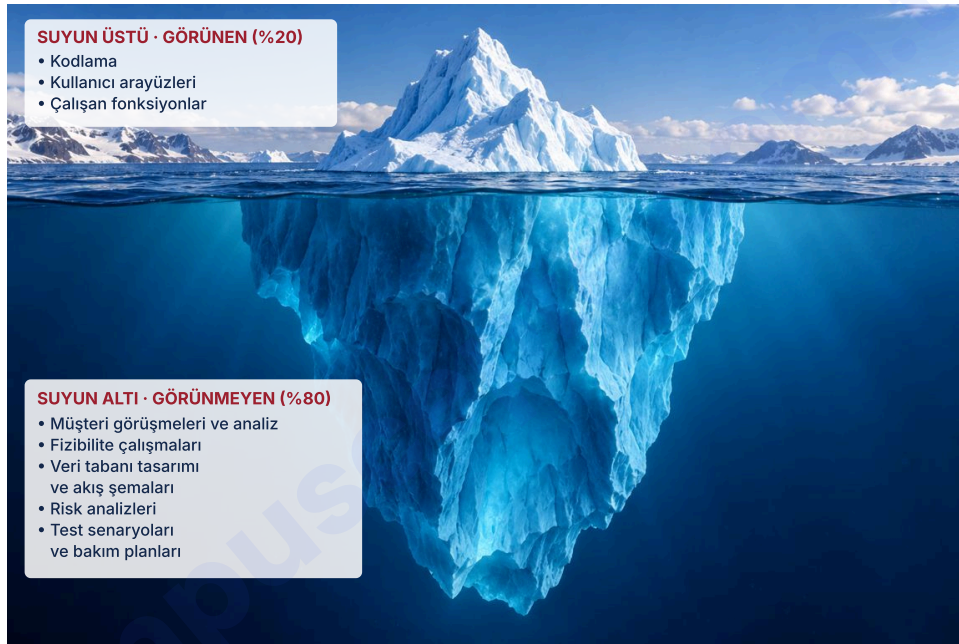


6.1 Yazılım Geliştirme Yaşam Döngüsü (SDLC)

Yazılım geliştirme, bilgisayar başına geçip kod yazmaktan ibaret değildir. Kodlama, tüm sürecin yalnızca %15-20'lik bir kısmını oluşturur. Geriye kalan %80; problemi doğru anlamak, kaliteyi denetlemek ve sürdürülebilirliği sağlamaktır. Bu bütünsel süreci yöneten disiplinin adı Yazılım Geliştirme Yaşam Döngüsü'dür (SDLC).

TANIM

Bir yazılım projesinin fikir aşamasından başlayarak kullanımdan kaldırılmasına, yani emekliye ayrılmasına kadar geçen tüm süreci yöneten yapısal bir metodolojidir.



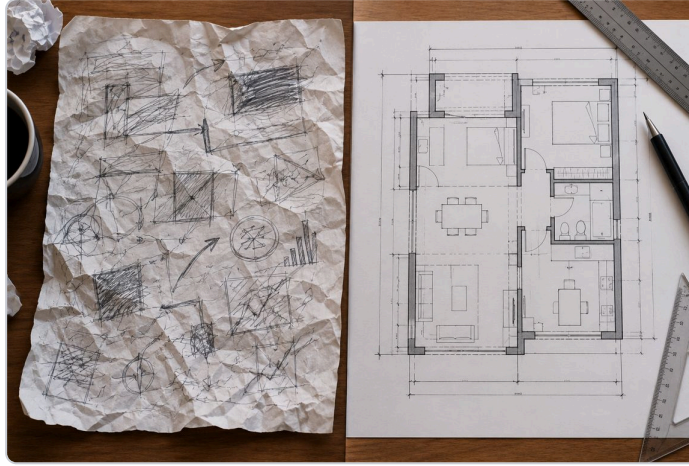
Buzdağı analojisi: görünen ve görünmeyen emek

Bir projeyi ayakta tutan ve başarılı kılan, suyun altındaki bu devasa kütledir. Yaygın yanlış, yazılım geliştirmenin kod yazmaktan ibaret olduğudur; oysa kod, buzdağının yalnızca görünen ucudur.

SDLC'nin Temel Amaçları

- ◆ **Disiplin:** Rastgele ve plansız çalışmayı önlemek.
- ◆ **Kontrol:** Projeyi belirlenen bütçe ve zaman sınırları içinde tutmak.
- ◆ **Kalite:** Müşteri gereksinimlerini tam olarak karşılayan, güvenilir bir ürün ortaya koymak.
- ◆ **Öngörülebilirlik:** Süreç adımlarını netleştirerek kötü sürprizleri en aza indirmek.

Program ve Yazılım Ürünü

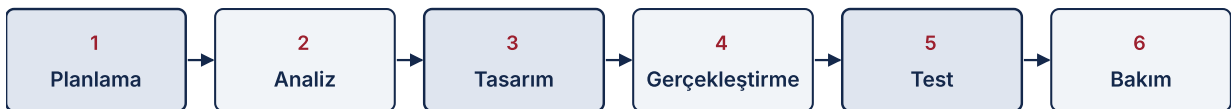


Solda dağınık bir karalama (program), sağda ölçülü ve düzenli bir teknik çizim (yazılım ürünü)

Program	Yazılım ürünü
Genellikle tek bir geliştirici tarafından yazılır.	Müşteri gereksinimlerine göre üretilir.
Dokümantasyonu yoktur veya zayıftır.	Kapsamlı testlerden geçer.
Yalnızca yazan kişi tarafından anlaşılabilir.	Kullanım kılavuzları ve teknik dokümantasyona sahiptir.
Bakımı ve geliştirilmesi zordur.	Bir ekip tarafından geliştirilebilir ve bakımı yapılabilir.

SDLC'nin amacı, "program yazmaktan" profesyonel standartlarda "yazılım ürünü" geliştirmeye geçişi sağlamaktır.

6.2 SDLC'nin Altı Temel Evresi



SDLC'nin altı temel evresi

1. Planlama

- ◆ Projenin fizibilitesi (yapılabilirliği) değerlendirilir. Bütçe, kaynak ve zaman planı oluşturulur. Temel soru: Bu projeyi yapmaya değer mi?
- ◆ SDLC'nin ilk aşaması olan planlama, net hedefler tanımlayarak ve bunlara ulaşmak için gerekenleri belirleyerek tüm projenin temelini atar.



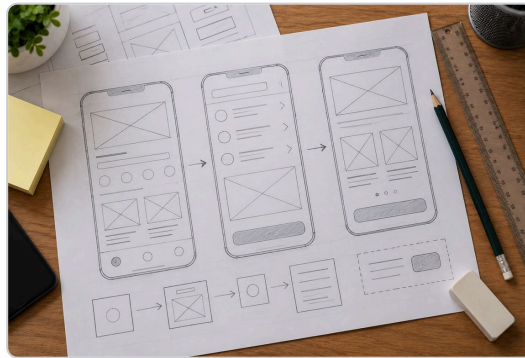
Planlama: takvim, bütçe ve zaman çizelgesi

2. Analiz

- ◆ Müşteri ve paydaşların ne istediği detaylıca anlaşılır.
- ◆ Sistem, fonksiyonel gereksinimleri tanımlar.

3. Tasarım

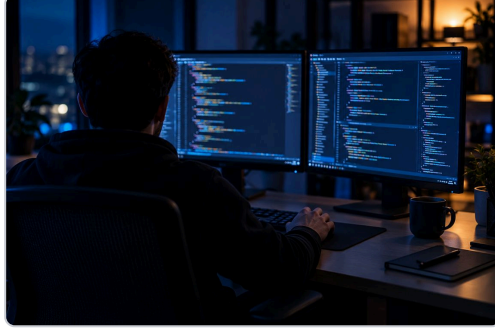
- ◆ Sistemin teknik mimarisi ve altyapısı çizilir.
- ◆ Veri tabanı şemaları ve kullanıcı arayüzleri tasarlanır.



Tasarım: kullanıcı arayüzü taslakları

4. Gerçekleştirme

- ◆ Tasarım, programlama dilleri ile koda dökülür.
- ◆ Veri tabanı ve sunucu altyapısı kurulur.
- ◆ Projenin “üretim” aşamasıdır.



Gerçekleştirme: tasarımın koda dökülmesi

5. Test

- ◆ Yazılımdaki hatalar tespit edilir ve düzeltilir.
- ◆ Sistemin, analiz evresindeki gereksinimleri karşılayıp karşılamadığı kontrol edilir.



Test: uygulamanın farklı cihazlarda kontrol edilmesi

6. Bakım

- ◆ Yazılım teslim edildikten sonraki süreçtir.
- ◆ Güncellemeler, yeni özellik eklemeleri ve hataların giderilmesini kapsar.

NOT

En uzun süren ve en maliyetli evre olarak kabul edilen aşama “Bakım” aşamasıdır.

EVRELERİN ÇIKTILARI

Analiz evresinin çıktısı Gereksinim Spesifikasyon Dokümanı'dır (SRS). Tasarım evresinde veri tabanı şemaları ER diyagramlarıyla, kullanıcı arayüzleri ise UI/UX tasarımlarıyla hazırlanır. Test evresinde tespit edilen hatalara "bug" denir.

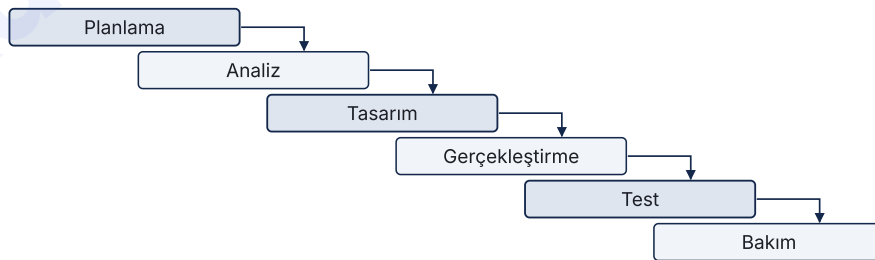
6.3 SDLC Modelleri

Her projenin doğası, büyüklüğü, risk seviyesi ve müşteri beklentileri farklıdır. Bu nedenle SDLC evrelerini uygulamak için tek bir doğru yol yoktur; farklı stratejik yaklaşımlar, yani modeller geliştirilmiştir. Doğru model seçimi, projenin başarısı için kritik bir yönetim kararıdır.

Şelale (Waterfall) Modeli

GELENEKSEL VE DOĞRUSAL

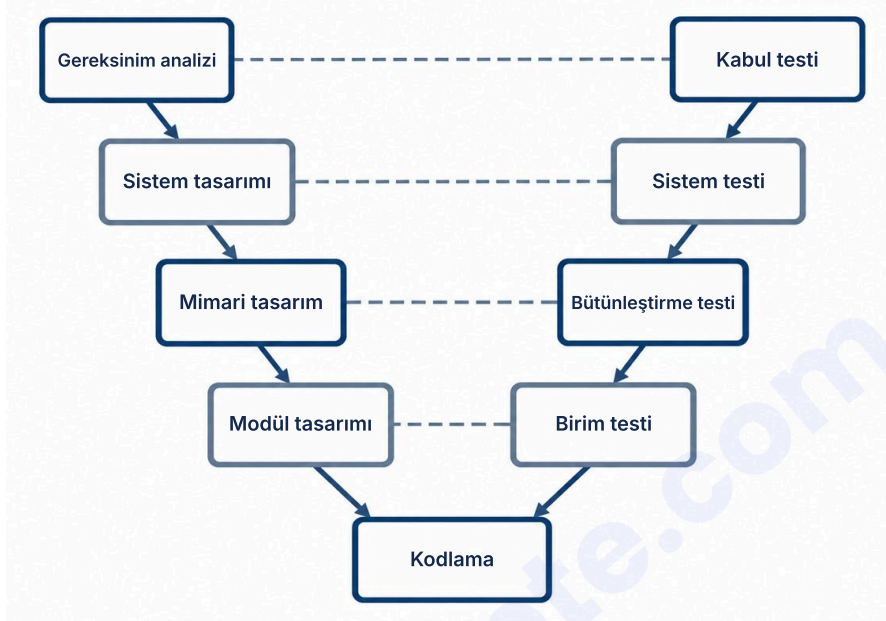
Her aşama bir öncekinin tamamen bitmesiyle başlar. Süreç yukarıdan aşağıya, doğrusal akar. Geriye dönüş çok zordur.



AVANTAJI	Yönetimi basit, dokümantasyonu güçlüdür.
DEZAVANTAJI	Değişime tamamen kapalıdır. Müşteri ürünü sadece projenin en sonunda görür.
KULLANIM ALANI	Gereksinimlerin asla değişmeyeceği askerî veya kamu projeleridir.

V-Modeli**TEST ODAKLI VE GÜVENLİ**

Şelalenin test odaklı versiyonudur. Her geliştirme adımının karşısında bir test adımı bulunur.

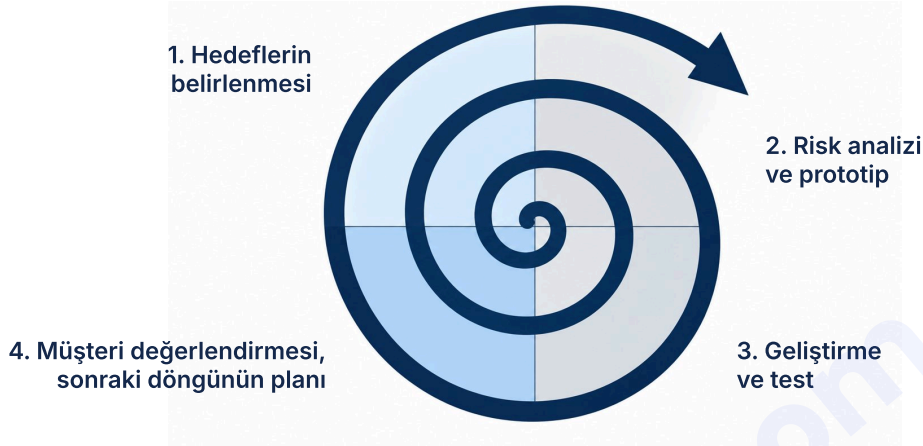


AVANTAJI	Hata tespiti çok erken başlar, kalite güvencesi yüksektir.
DEZAVANTAJI	Şelale gibi esnekliği azdır ve test süreçleri nedeniyle maliyetli olabilir.
KULLANIM ALANI	Hatanın kabul edilemez olduğu kritik sistemler (tıbbi cihazlar, havacılık yazılımları).

Spiral Model

RİSK ODAKLI VE TEKRARLI

Proje, risk analizi odaklı döngülerle (sarmallar) ilerler. Her döngüde bir prototip üretilir ve müşteri değerlendirmesi alınır.



AVANTAJI	Risk yönetimi mükemmeldir. Prototip sayesinde belirsizlikler ve sürprizler azalır.
DEZAVANTAJI	Çok karmaşık ve maliyetlidir. Küçük projeler için uygun değildir.
KULLANIM ALANI	Çok büyük, karmaşık ve risk seviyesi yüksek olan projeler.

Çevik (Agile) Model

ESNEK VE MÜŞTERİ ODAKLI

Proje, “sprint” adı verilen kısa (2-4 hafta) döngülere bölünür. Her sprint sonunda çalışan bir ürün parçası teslim edilir.



AVANTAJI	Değişime çok hızlı adapte olur. Sürekli müşteri geri bildirimi ile memnuniyet yüksektir.
DEZAVANTAJI	Yoğun müşteri katılımı gerektirir. Hızlı döngüler nedeniyle dokümantasyon zayıf kalabilir.
KULLANIM ALANLARI	Mobil uygulamalar, web projeleri.

EK BİLGİ

Çevik yaklaşımın en yaygın uygulama biçimi Scrum'dır. Scrum'da ekip her gün kısa bir toplantı yapar, her sprint sonunda da yapılan işi müşteriye gösterir.

6.4 Modellerin Karşılaştırılması

Kriter	Şelale	V-Model	Çevik Model	Spiral Model
Değişime esneklik	Çok düşük	Düşük	Çok yüksek	Orta
Risk yönetimi	Zayıf	İyi	İyi	Mükemmel
Müşteri katılımı	Sadece başta	Sadece başta	Sürekli	Periyodik
Teslimat hızı	Yavaş	Yavaş	Hızlı (parçalı)	Orta
Dokümantasyon	Çok güçlü	Güçlü	Zayıf	Güçlü

- ◆ Sonuç olarak en iyi model yoktur, projeye en uygun model vardır.
- ◆ Bir nükleer santral kontrol yazılımı için sıfır hata hedefiyle **V-Modeli** seçilmelidir.
- ◆ Pazar koşullarına hızlı adapte olması gereken bir e-ticaret sitesi için **Çevik (Agile)** model idealdir.
- ◆ Gereksinimleri en başından net olan bir devlet arşivi sistemi için **Şelale** modeli yeterli olabilir.

6.5

Vaka Analizi: Güvenlik Kritik ve Belirsiz Teknoloji Senaryosu

SORU

Bir biyoteknoloji firması, insan kanındaki patojenleri gerçek zamanlı analiz edip dozaj ayarlayan, vücuda implante edilebilir (gömülü) bir tıbbi cihaz için yazılım geliştirmektedir. Projenin parametreleri şöyledir:

- ◆ **Güvenlik kritik (safety-critical):** Yazılımdaki en ufak bir hata hastanın hayatını kaybetmesine neden olabilir (sıfır hata toleransı).
- ◆ **Teknolojik belirsizlik:** Kullanılacak biyosensör teknolojisi henüz çok yenidir ve geliştirme sürecinde donanım spesifikasyonlarının değişme ihtimali yüksektir.
- ◆ **Regülasyon zorunluluğu:** Ürünün piyasaya çıkabilmesi için birçok otoriteden onay alması gerekmekte, bu da çok sıkı bir dokümantasyon ve izlenebilirlik (traceability) zorunluluğu getirmektedir.

Bu proje için en uygun model nedir? Uygun olmayan diğer modeller bu spesifik senaryoda neden yüksek başarısızlık riski taşır? Gerekçeleriyle açıklayınız.



Senaryodaki ürün: vücuda yerleştirilebilir küçük bir sensör (temsili)

1. Adım: Projenin Üç Filtresi

Soruyu çözmek için önce senaryodaki üç parametre, her modelin geçmesi gereken birer filtre olarak düşünülür:

Hayat kurtarma sorumluluğu	Sıfır hata toleransı vardır. Yazılımın en küçük bir kusuru dahi hasta ölümüyle sonuçlanabilir. Bu, “fail fast” (hata yap, çabuk düzelt) felsefesini imkânsız kılar.
Teknolojik belirsizlik	Biyosensör donanımı değişkendir. Model, geliştirme sürecinde donanım spesifikasyonlarındaki değişikliklere adapte olabilmelidir. Katı, değişime kapalı yapılar başarısız olur.
Yoğun regülasyon yükü	FDA gibi otoriteler ve IEC 62304 gibi tıbbi cihaz yazılımı standartları, her gereksinimin koda ve teste kadar izlenebildiği kapsamlı dokümantasyon talep eder. Hafif dokümantasyon yaklaşımları kabul edilemez.

2. Adım: Elenen Modeller

Şelale (Waterfall)	Gereksinimler en başta “dondurulur”. Projenin ortasında biyosensör donanımı değişirse analiz fazına geri dönmek projeyi kilitler ve maliyeti katlar; yani değişime adapte olamaz. Ayrıca test aşaması projenin sonundadır: hayati bir hata projenin sonunda fark edildiğinde düzeltilmesi hem çok maliyetli hem de risklidir. Hasta hayatının söz konusu olduğu ve donanımın değişken olduğu bir sistemde kabul edilemez.
Çevik (Agile / Scrum)	Çevik yaklaşım “kapsamlı dokümantasyondan ziyade çalışan yazılım” der; otoriteler ise “kapsamlı dokümantasyon yoksa ürün de yoktur” der. İzlenebilirliği sağlamak çok zordur; ürün denetimden geçemez, onay alamaz. Ayrıca “fail fast” mantığı hasta üzerinde denenemez; sürekli değişim, sistemin başka bir yerinde kritik bir fonksiyonu bozma (regresyon) riskini artırır.
RAD (Hızlı Uygulama Geliştirme)	Hız odaklıdır; güvenlik ve doğrulama gibi kritik süreçleri atlama eğilimindedir. Tıbbi implant ile hızın birleşimi felaket riskidir.
Saf prototipleme	Hızlı ve deneyseldir, ancak formal doğrulama, izlenebilirlik ve regülasyon uyumu yoktur. Tek başına kullanılamaz; daha güçlü bir modelin içinde destekleyici bir araç olabilir.

3. Adım: Finale Kalan İki Model

V-Modeli

Güçlü yönü: Doğrulama ve geçerleme (V&V) sürecini merkeze alır; her geliştirme adımının karşısında bir test adımı planlanır. Bu, sıfır hata toleransını destekler. Tıbbi cihaz standartlarının istediği izlenebilirlik yapısını doğal olarak sağlar; denetimler için çok uygundur. **Zayıf yönü:** Klasik hâliyle katıdır; teknolojik belirsizliği tek başına yönetemez.

Spiral model

Güçlü yönü: Risk yönetimini merkeze alan klasik modeldir. Her döngüde prototipler ve simülasyonlarla donanım değişikliklerine kontrollü biçimde adapte olur; her döngüde risk analizi ve doğrulama adımları bulunduğu için hatalar erken yakalanır. **Zayıf yönü:** V-Modeli kadar net bir “gereksinim–test” eşleşmesi sunmaz; süreç yönetimi karmaşık ve maliyetli olabilir.

4. Adım: Modellerin Karnesi

Model	Hayat kurtarma sorumluluğu (sıfır hata)	Teknolojik belirsizlik (adaptasyon)	Regülasyon yükü (izlenebilirlik)
Şelale	Uygun değil	Uygun değil	Uygun
Çevik (Agile / Scrum)	Uygun değil	Uygun	Uygun değil
Saf V-Modeli	Uygun	Kısmen	Uygun
Saf Spiral model	Uygun	Uygun	Kısmen
Hibrit (V + Spiral)	Uygun	Uygun	Uygun

CEVAP

En uygun model: Risk odaklı Spiral model yaklaşımını, V-Modeli'nin doğrulama ve geçerleme (V&V) disipliniyle birleştiren **hibrit modeldir**. Ana çerçeve V-Modeli'dir: projenin genel yapısını, resmî onay kapılarını ve regülasyonlar için gereken tam izlenebilirliği sağlar. Esneklik katmanı Spiral'den gelir: her geliştirme döngüsü içinde, özellikle teknolojik belirsizliği yönetmek için risk analizi ve prototipleme fazları yürütülür.

Yüksek başarısızlık riski taşıyanlar: Şelale (değişime kapalı, hatayı çok geç bulur), Çevik / Scrum (regülasyon ve güvenlikle uyumsuz), RAD (hızı güvenliğin önüne koyar).

Çıkarım: İnsan hayatının söz konusu olduğu, teknolojisi belirsiz ve ağır regülasyona tabi sistemlerde; risk analizini merkeze alan ve bunu katı doğrulama süreçleriyle birleştiren hibrit bir yaklaşım, başarısızlık ihtimalini en aza indiren mantıklı seçimdir. Mühendislikte “en iyi” model yoktur, “en uygun” model vardır.

SINAVDA NASIL YAZILIR?

Böyle bir soruda yalnızca model adını yazmak yetmez. İyi bir cevap şu sırayı izler: (1) senaryodaki kısıtları tek tek say, (2) her modeli bu kısıtlara göre değerlendir, (3) elenenlerin neden elendiğini senaryodaki ifadelerle bağlayarak yaz, (4) seçtiğin modelin zayıf yönünü de belirt ve nasıl kapatılacağını söyle.

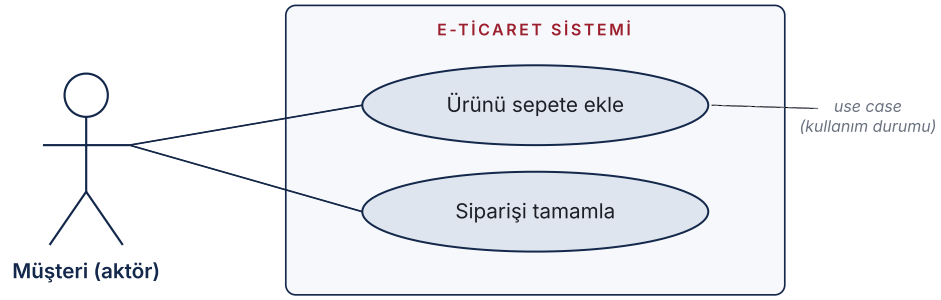
6.6 Use Case ve Sistem Gereksinimleri**TANIM**

Use case (kullanım durumu), bir sistem senaryosudur. Odak noktası sistemin “nasıl” yaptığı değil, “ne yapar?” sorusudur.

Film sahnesi benzetmesiyle düşünülebilir: aktör başroldedir, sistem ise yardımcı roldedir. Bir use case şu akışla okunur: **aktör → işlem → amaç → sistem tepkisi**.

Temel Bileşenler

Aktör	Sistemi kullanan kullanıcı ya da başka bir sistemdir.
Amaç	Aktörün ulaşmak istediği hedeftir.
Ön koşul	Senaryonun başlayabilmesi için sağlanması gereken giriş şartıdır.
Ana senaryo	Her şeyin yolunda gittiği akıştır (“mutlu yol”).
Alternatif senaryo	İstisna ve hata durumlarında izlenen akıştır.



Basit bir use case diyagramı: bir aktör ve iki kullanım durumu

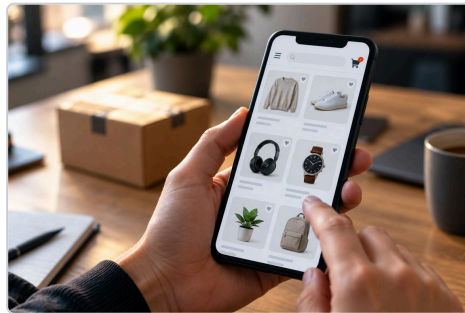
Hamburger Analjisi (Süreç)

Use case, yazılım sürecinin en üst katmanıdır; diğer katmanlar onun üzerine kurulur. Hamburgerin katmanları yukarıdan aşağıya şöyledir:



Hamburger analjisi: use case en üst katmandır, diğer katmanlar onun üzerine kurulur

E-Ticaret Örnekleri



Bir alışveriş uygulamasında ürün listesi ve sepet

	Senaryo 1: Ürün sepete ekleme	Senaryo 2: Sipariş tamamlama
Aktör	Müşteri	Müşteri
Amaç	Ürünü sepete atmak	Satın almayı bitirmek
Ön koşul	Ürün stokta olmalı	Sepet dolu, üye girişi yapılmış
Akış	“Sepete Ekle” butonuna tıkla → sistem stok kontrolü yapar → ürün sepete eklenir → “Eklendi” onayı gösterilir.	Adres seç → ödeme bilgisi gir → “Siparişi Onayla” tıkla → sistem ödemeyi alır → sipariş oluşturulur.
Alternatif	Stok yoksa uyarı verilir.	Ödeme reddedilirse hata gösterilir.

Diyafram Araçları

Use case diyagramları çizmek için PlantUML, draw.io ve Mermaid gibi araçlar kullanılabilir.