

# “Use-Case” Esaslı Gereksinim Analizi

# İçerik

- UML Giriş
- Gereksinim Analizi İçin Kullanılan Başlıca UML Elemanları
- “Use-Case” Esaslı Gereksinim Analizi
- Örnek Çözümleme: Kütüphane Destek Sistemi

# “Unified Modeling Language” - 1

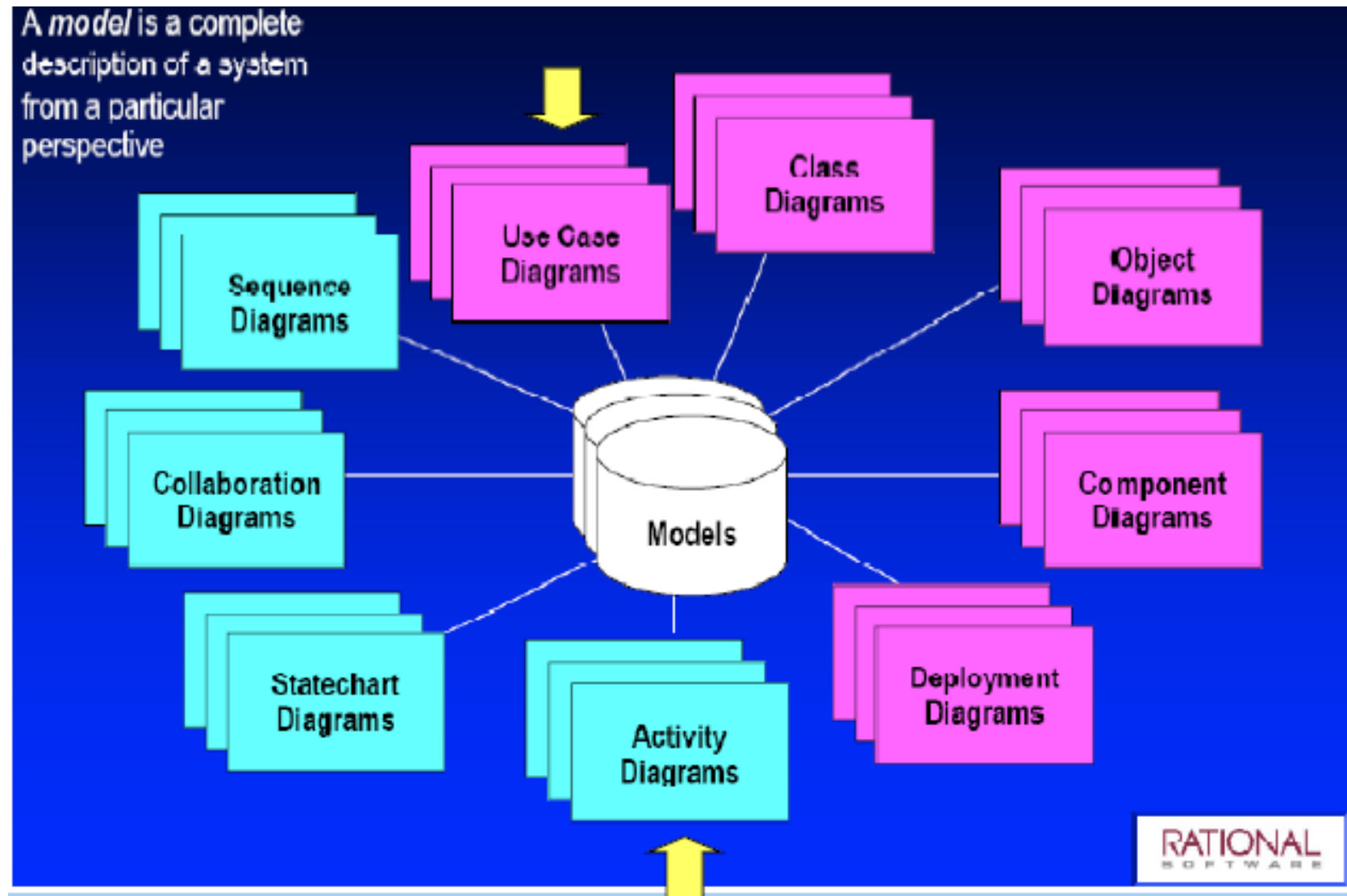
- Yazılım sistemlerinin modellemesi için geliştirilmiş standart bir dildir
  - Yazılım is ürünlerinin; tanımlanması, görsel hale getirilmesi, belgelendirilmesi
  - Açık standarttır; birçok araç tarafından desteklenir
  - Tüm yazılım geliştirme sürecini destekler
- Çıkış hedefleri:
  - Kullanımı kolay, görsel bir modelleme dili sunmak
  - Programlama dillerinden ve geliştirme sürecinden bağımsız olmak
  - En iyi yöntemleri bütünleştirmek
- 1995 yılında, yazılımlarda bir standart yaklaşım oluşturmak için geliştirilmiştir.
  - Son sürüm 2015 yılında geliştirilen UML 2.5
  - İlgili web sitesi: [www.uml.org](http://www.uml.org)

# “Unified Modeling Language” - 2

- Sundukları:
  - Yazılım ürünlerinin gösterimi için yapı tasları ve ilişkiler
  - Yapı taslarını ve ilişkileri kullanarak farklı bakış açılarını destekleyen diyagramlar
  - Tanımlanan yapı taslarının ve diyagramların bütünü, geliştirilen yazılım sistemine karşılık gelir
- Sunmadıkları:
  - Sisteminin nasıl geliştirilmesi gerektiğini tanımlamaz
  - Nesne yönelimli yazılım modellemesi için yapılar sunar, ancak;
  - Bu yapıların hangi sıra ile kullanılması gerektiğini tanımlamaz
  - Yapıların geliştirme sürecinin hangi aşamalarında kullanılması gerektiğini tanımlamaz

# Gereksinim Analizi İçin Kullanılan Başlıca UML Elemanları

# UML Diyagramları



# “Use-Case Diagram” Modelleme Ögeleri

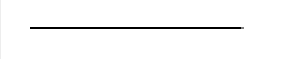
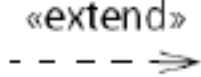
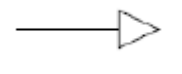
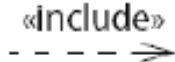


- Aktör
  - Sistemin kullanıcıları
- Use case
  - Sistemin destekleyeceği işler



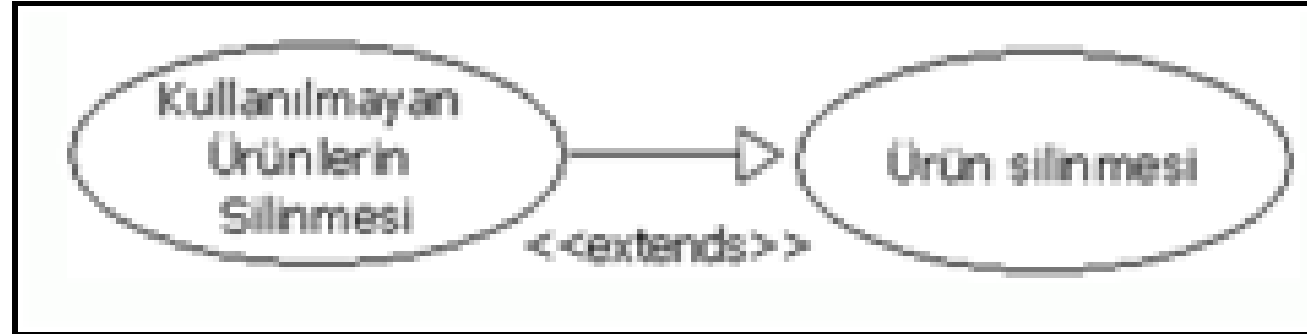
- İlişki (relationship)
  - Association
  - Generalization
  - Extend
  - Include

# “Use-Case” İlişki Türleri

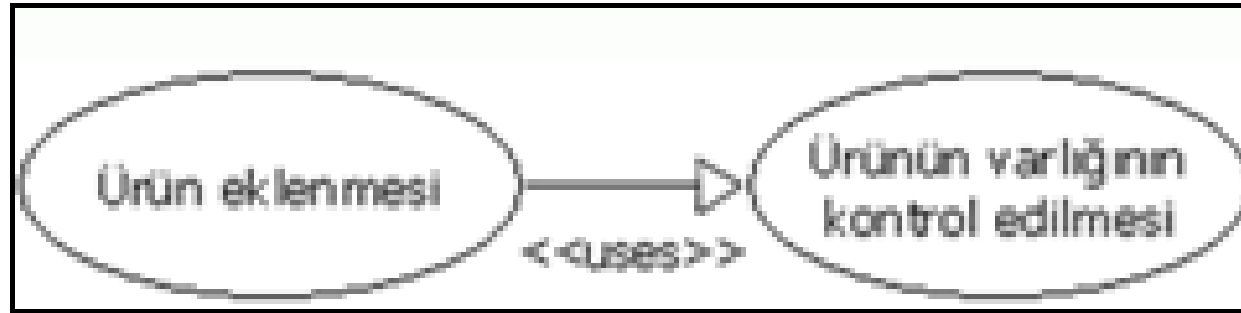
İlişki	Fonksiyon	Sembol
Association	Aktör ve “use-case” arasındaki bağlantıdır.	
Extend	Bir use-case’in bir başka use-case’in değişik biçimlerine (variation) sahip olduğunu gösterir.	
Generalization	Bir use-case’in diğerinin özel bir hali olduğunu gösterir.	
Include	Bir use-case’in bir başka use-case’i içine alması durumudur.	



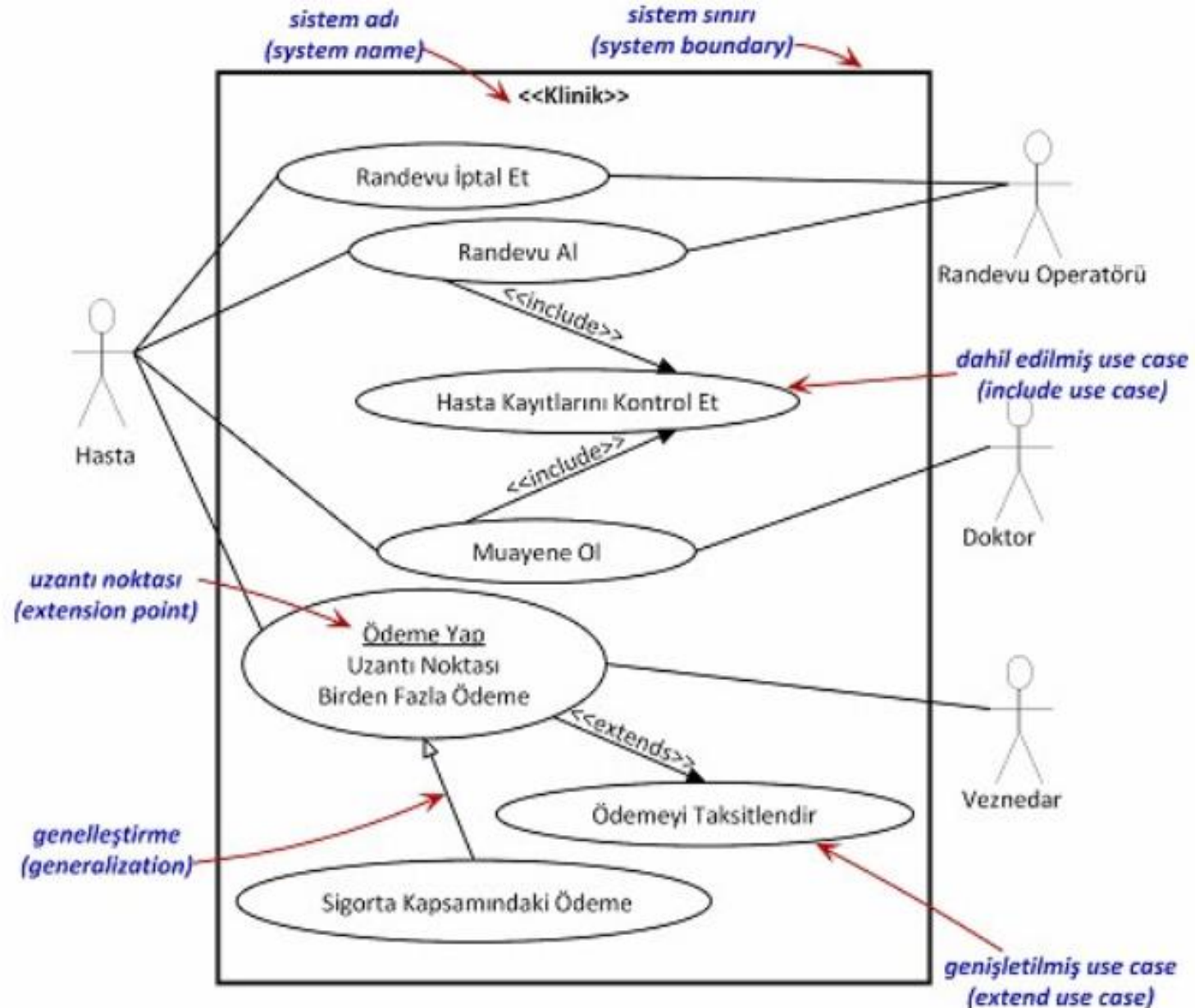
# “Use-Case” İlişki Türleri - Extends



# “Use-Case” İlişki Türleri - Include



# Örnek Use Case Diyagramı



# “Use-Case” Modeli

- Sistemin tüm “use-case” diyagramları, “use-case” modelini tanımlar
  - Sistem belirli büyüklüğün üzerinde olduğunda, gereksinimler birden fazla “use-case” diyagramı kullanılarak tanımlanır
  - Sistem sınırını gösteren dikdörtgen kutu sistemin içinde ve dışında kalan öğeleri belirtmek için kullanılır

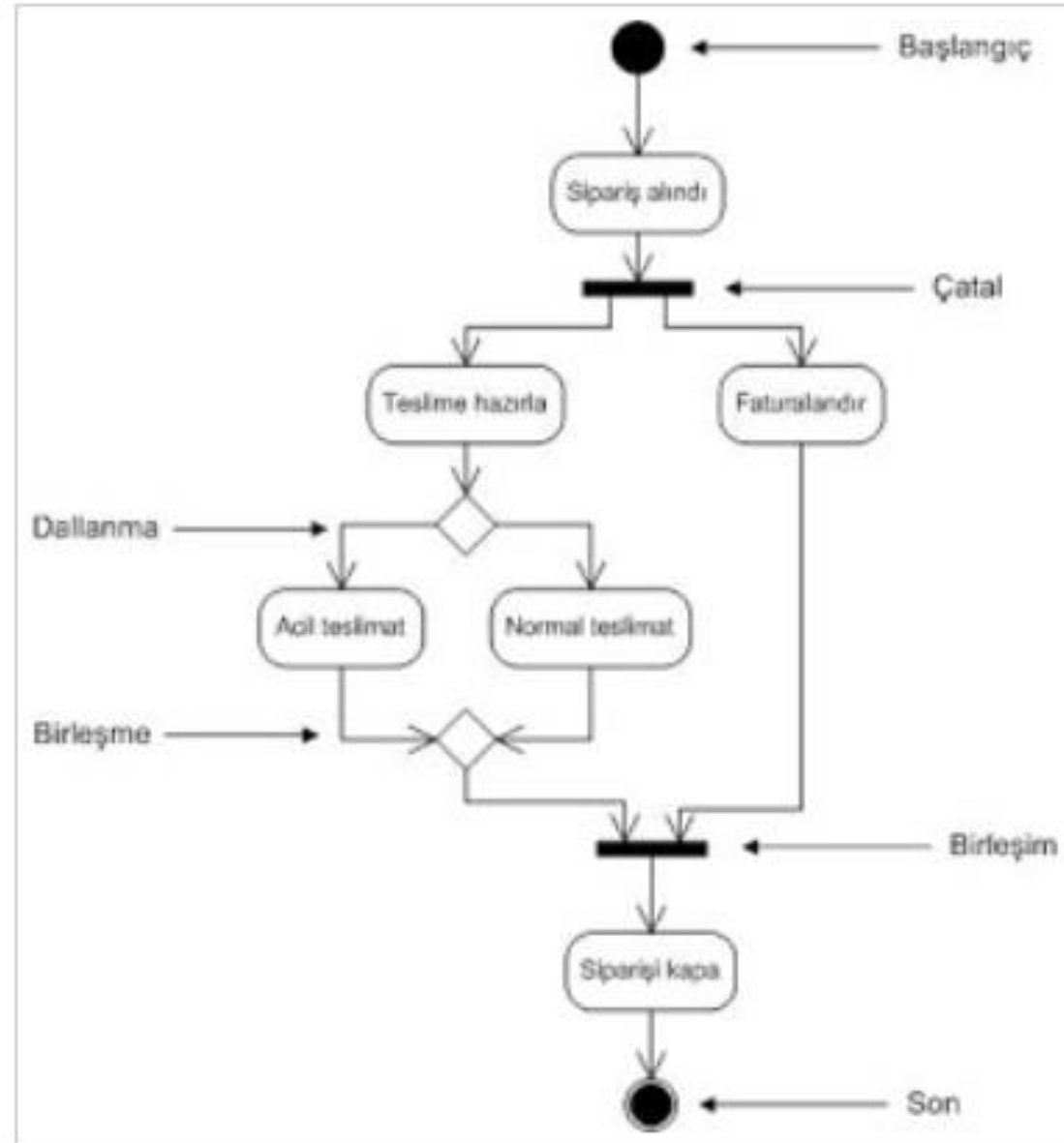
# Activity (Aktivite-Etkinlik) Diyagram

- Aktivite (etkinlik) diyagramı, bir UML diyagram çeşidi olup, aktivitelerin süreçteki dizilişini tanımlamak için kullanılır.
- İş analistleri, etkinlik diyagramlarını iş süreçlerindeki değişimi analiz etmek, bir iş akışını göstermek ya da karmaşık bir algoritmayı UML ile modellemek için kullanabilir.
- Aktivite diyagramları sayesinde iş süreçleri gözden geçirilerek, gereksiz şekilde ardışık yapılan işlemlerin paralel olarak yapılabileceği tespit edilebilir.
- Kullanım şekilleri (use case) iş sürecinin metinsel olarak anlatımını sağlamakta olup; eğer akışlar arasındaki bağlantılar çok kompleks ise, kullanım şekillerinin etkinlik diyagramları ile desteklenmesi gerekebilir.
- Etkinlik diyagramları, sistemsel dizayn ve test süreçleri için bir input olarak kullanılabilir. Sistem dizaynındaki detaylar ve test senaryoları, etkinlik diyagramında belirtilen iş akışına göre kolaylıkla oluşturulabilir.

# “Activity Diagram” Modelleme Öğeleri

- İlk Aktivite (Initial Activity): İşlem akışının ilk aktivitesi ya da başlangıç noktasıdır.
- Aktivite(Activity): Sistem ve aktörler tarafından yapılan işleri ifade etmek için kullanılır.
- Geçiş (transition): Etkinlikler arasındaki geçişleri ifade etmek için kullanılır.
- Branch” / “merge”: Koşullu davranışı ifade etmek için kullanılır.
- “Fork” / “join”: Paralel davranışı ifade etmek için kullanılır.
- Final node : Diyagramın bitişini göstermek için kullanılır.

# Activity Diyagram Örnek



# “Use-Case” Esaslı Gereksinim Analizi



# Gereksinim Analizinde “Use-Case” Yaklaşımı

- Bakış açısı: Sistem, kullanıcısı için “ne” yapacak ?
  - ✓ Sistem kapalı bir kutu (“black-box”)
  - ✓ Sistem-kullanıcı etkileşimi
  - ✓ Sistemin dışarıdan görünen davranışı
- İlgilenmediklerimiz:
  - ✓ Sistemin iç yapısı
  - ✓ Sistem belirlenen davranışı “nasıl” yapacak ?
  - ✓ Belirlenen davranış “nasıl” kodlanacak ?
  - ✓ *Bu bakış açısı, sistemdeki tüm işlevselliği değil, kullanıcılar için artı değer oluşturacak işlemleri düşünmemizi sağlar*

# “Use-Case” Nedir? (1)

- Yazılım sisteminin kullanıcıya değer döndüren, dışarıdan gözlemlenebilen davranışının bütünüdür
  - **Değer**
    - ✓ Sistem “use case” kapsamındaki işleri gerçekleştirdiğinde oluşacak çıktı, sonuç veya durumdur
      - **Gözlemlenebilirlik**
    - ✓ Kullanıcı ve sistem arasındaki etkileşimi tanımlar
      - **Bütünlük**
  - ✓ “Use-case” kullanıcının istediği değer üretilebilmesi için gereken tüm ilişkili adımları içerir

# “Use-Case” Nedir? (2)

- Her “use-case”in bir amacı vardır
  - Niye belirlendi ? Kullanıcının hangi işlemini gerçekleştirecek ?
  - Hangi davranışı modelliyor ? Ne yapacak ?
  - Ne zaman sonlanacak ? Hangi değeri üretecek ?
- Örnekler:
  - Öğrencinin bir derse kaydolması
  - Muhasebe görevlisinin aylık bordroları hazırlaması
  - Banka müşterisinin ATM’den para çekmesi
  - Bir kişinin asansörü çağırması

# Gereksinim Analizinde Yapısal Yöntem ya da “Use-Case” Yaklaşımı: Örnek

- Yapısal yöntem:
  - Sistem müşteri kartını kabul eder.
  - Sistemde işlemlerin yapılabilmesi için şifre geçerli olmalıdır.
  - Sistemde para çekme ve para yatırma seçenekleri mevcuttur.
  - Sistem istendiğinde işlemlerin makbuzunu verir.
  - .....
- Use case yaklaşımı:
  - Müşteri kartını yerleştirir
  - Sistem şifreyi sorar
  - Müşteri şifresini girer
  - Şifre doğruysa, sistem para çekme ve yatırma seçeneklerini sunar.
  - Müşteri, para çekme işlemini sağlar.
  - .....

# “Use-Case” Esaslı Gereksinim Analizi – Kazançlar

- Kullanıcının gereksinimi olmayan özellikleri tanımlamamızı engeller.
- Kullanıcının da anlayabileceği şekilde sistemin davranışlarını ve sorumluluklarını tanımlar.
- Kullanıcı ile iletişimi kolaylaştırır.
- Kullanıcı arayüzlerinin tasarlanmasını kolaylaştırır.
- Kullanıcı kılavuzlarını yazarken başlangıç noktasını oluşturur.
- Geliştirme sürecini başlatır ve tüm temel iş adımlarını birbirine bağlar.
- Tasarlanacak test durumlarına esas oluşturur.

# “Use-Case” Esaslı Gereksinim Analizi – Dikkat Edilecek Noktalar

- “Use-case”ler sistemin iç yapısını tanımlamaz (sistem kapalı kutu)
  - ✓ Tasarım ögeleri belirsizdir
    - Tipik olarak birden çok tasarım ögesi, bir use case’in işletilmesi için kullanılır
  - ✓ Yapısal veya nesne yönelimli yaklaşımlarda kullanılabilmesinin temel nedeni budur
- “Use case”ler sadece işlevsel gereksinimleri, kullanıcı bakış açısıyla tanımlar
  - ✓ Sistemin iç davranışına ilişkin gereksinimler, kullanıcı gereksinimleri gerçekleştirilirken daha sonraki adımlarda karşılanır.
    - Bu tür gereksinimler sıklıkla, iş kuralı veya tasarım kısıtı olarak “use case”lerle ilişkilendirilir

# “Use-Case” Esaslı Gereksinim Analizi: Yöntem

## 1. Aktörleri ve “use-case”leri belirle

Amaç: Sistemin aktörlerini ve “use-case”lerini belirlemek ve üst seviye “use-case” modelini oluşturmak

- Aktörler belirlenir
- “Use-case”ler belirlenir
- Her aktör ve “use case” kısaca tanımlanır
- Üst seviye “use-case” modeli tanımlanır

## 2. “Use-case”leri detaylandır

Amaç: Belirlenen tüm “use-case”lerin is akışlarını detaylı olarak tanımlamak

- Ana akış tanımlanır
- Alternatif akışlar tanımlanır

# “Use-Case” Esaslı Gereksinim Analizi: Yöntem (devamı)

## 3. “Use-case” modelini yapılandır

Amaç: Oluşturulan use case modelini ortak noktaları en aza indirecek şekilde yapılandırmak

- Gereken yerlerde “extend” ve “include” ilişkileri kullanılabilir
- Yapılandırılan “use-case” modeli, is süreçlerini referans alınarak değerlendirilir

## 4. Kullanıcı arayüzlerini tanımla

Amaç: Use case tanımları esas alınarak kullanıcı arayüzlerini üst seviyeli olarak tanımlamak

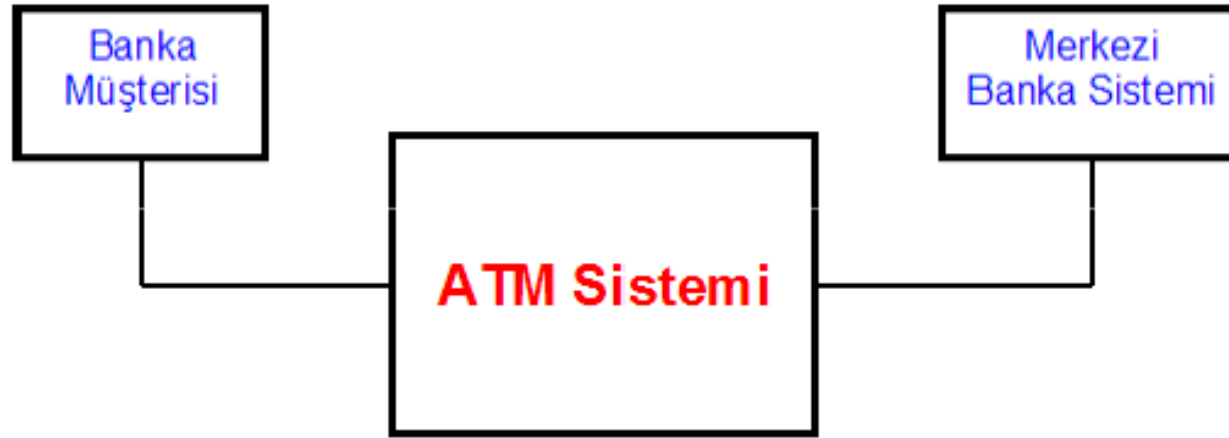
- Kağıt üzerinde çizim yapılabilir
- Arayüz prototipleme aracı kullanılabilir



# Örnek: ATM Uygulaması

- Bir bankanın ATM cihazı için yazılım geliştirilecektir. ATM, banka kartı olan müşterilerin hesaplarından para çekmelerine, hesaplarına para yatırmalarına ve hesapları arasında para transferi yapmalarına olanak sağlayacaktır. ATM, banka müşterisi ve hesapları ile ilgili bilgileri, gerektiğinde merkezi banka sisteminden alacaktır. Banka sistemi ayrıca her günün sonunda, ATM'den günlük işlemlerin bir özetini isteyecektir.

# ATM Uygulaması – Kapsam



*Bağlam (“context”) diyagramı*

# ATM Uygulaması (Adım 1. Aktörleri ve “Use Case”leri Belirle) – Aktörler

Soru: ATM uygulama yazılımının kullanıcıları kimlerdir?

- Banka müşterisi
- Merkezi Banka Sistemi

# ATM Uygulaması (Adım 1. Aktörleri ve “Use Case”leri Belirle) – “Use Case”ler

Soru: Belirlenen aktörler ATM’den ne istiyorlar ?

- Aktör: Banka müşterisi
  - Para çekme
  - Para yatırma
  - Para transferi
- Aktör: Merkezi Banka Sistemi
  - Günlük özet alma

# ATM Uygulaması (Adım 1. Aktörleri ve “Use Case”leri Belirle) – Kısa Tanımlar

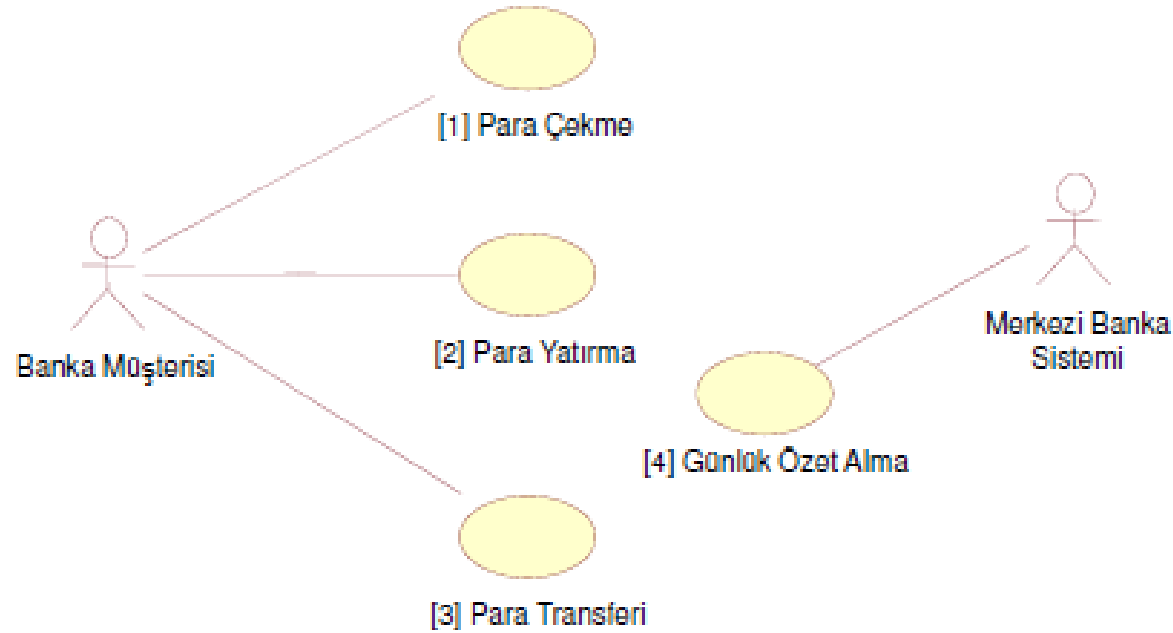
- Aktör: Banka müşterisi

Bankada hesabı ve banka kartı olan, ATM’den işlem yapma hakkı olan kişi

- Use case: Para çekme

Banka müşterisinin nasıl para çekeceğini tanımlar. Para çekme işlemi sırasında banka müşterisinin istediği tutarı belirtmesi ve hesabında bu tutarın mevcut olması gerekir.

# ATM Uygulaması (Adım 1. Aktörleri ve “Use Case”leri Belirle) – “Use-Case” Diyagramı



*Her use case biricik (“unique”) olarak numaralandırılmış*

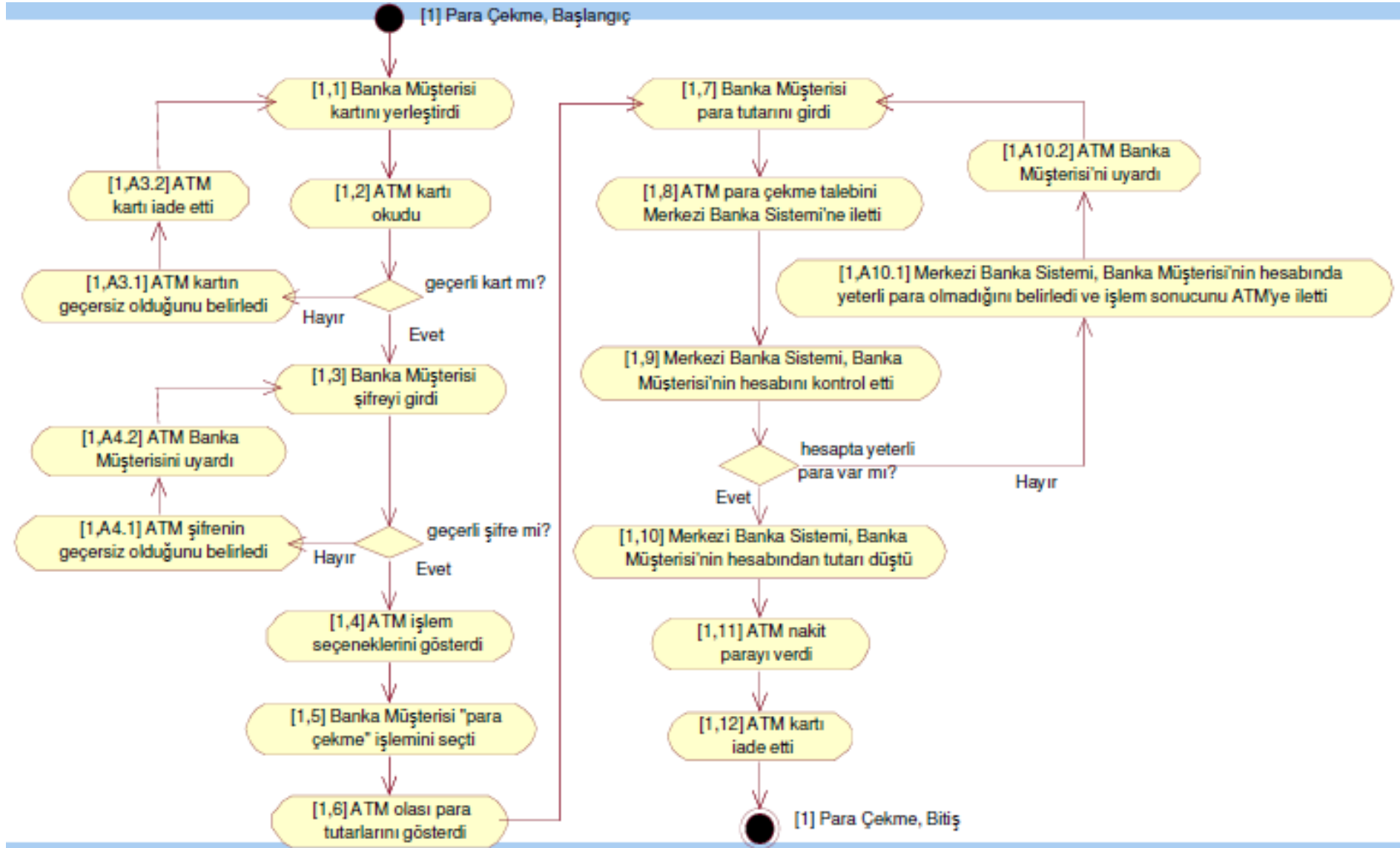
# ATM Uygulaması (Adım 2. “Use Case”leri Detaylandır) – Para Çekme (Ana Akıs)

1. Banka Müşterisi kartını yerleştirir
2. ATM kartı okur
3. Banka Müşterisi şifreyi girer
4. ATM işlem seçeneklerini gösterir
5. Banka Müşterisi “para çekme” işlemi seçer
6. ATM olası para tutarlarını gösterir
7. Banka Müşterisi para tutarını girer
8. ATM para çekme talebini Merkezi Banka Sistemi'ne iletir
9. Merkezi Banka Sistemi, Banka Müşterisi'nin hesabını kontrol eder
10. Merkezi Banka Sistemi, Banka Müşterisi'nin hesabından tutarı düşü ve işlem sonucunu ATM'ye iletir
11. ATM nakit parayı verir
12. ATM kartı iade eder



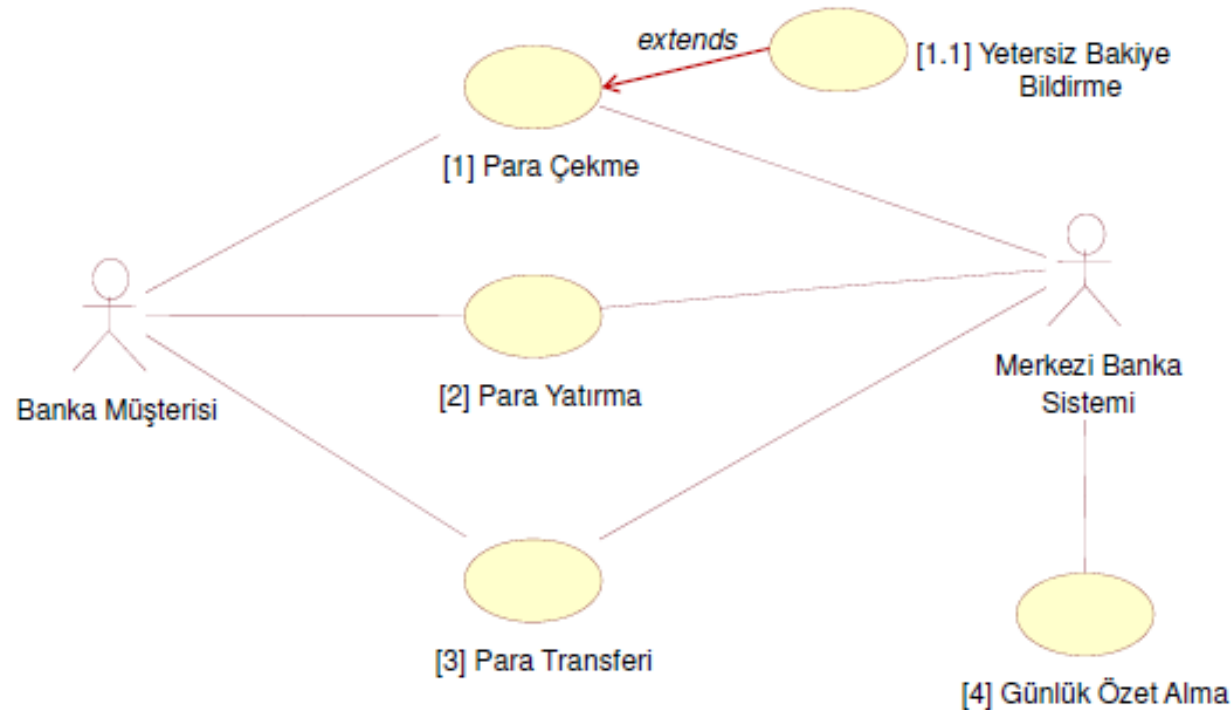
Olası sapma noktaları

# ATM Uygulaması (Adım 2. “Use Case”leri Detaylandır) – Para Çekme (“Activity Diagram”)





# ATM Uygulaması (Adım 3. “Use-Case” Modelini Yapılandır) – “Use-Case” Diyagramı



# “Use-Case” Esaslı Gereksinim Analizi: Basarı İçin Anahtar Noktalar

- “Use-case” modelini iteratif olarak geliştirin
- Kullanıcıları analize dahil edin
- “Use-case”leri görsel olarak modelleyin
- “Use-case”leri işlevsel olmayan gereksinimleri çıkarmak için kullanın
- “Use-case”leri ve “use-case” senaryolarını önceliklendirin
- “Use-case”leri doğrulayın ve diğer geliştirme öğelerine izlenirliğini kurun