

TÜMLEŞİK MODELLEME DİLİ

UML

(Unified Modeling Language)

UML NEDİR?

- * Yazılım ve donanımların bir arada düşünülmesi gereken,
- * Zor ve karmaşık programların,
- * Özellikle birden fazla yazılımcı tarafından kodlanacağı durumlarda,
- * Standart sağlamak amacıyla endüstriyel olarak geliştirilmiş grafiksel bir dildir.
- * ~~Programlama dili~~ - Diyagram çizme ve ilişkisel modelleme dili

UML NEDİR?

- * UML 'in doğuşu son yıllarda yazılım endüstrisindeki en büyük gelişmelerden biri olarak kabul edilebilir.
- * UML 1997 yılında yazılımın, diyagram şeklinde ifade edilmesi için bir standartlar komitesi tarafından oluşturuldu.
- * Diğer dallardaki mühendislerin standart bir diyagram çizme aracı -Programcıların UML

UML NEDİR?

- * UML yazılım sisteminin önemli bileşenlerini tanımlamayı, tasarlamayı ve dokümantasyonunu sağlar
- * Yazılım geliştirme sürecindeki tüm katılımcıların (kullanıcı, iş çözümleyici, sistem çözümleyici, tasarımcı, programcı,...) gözüyle modellenmesine olanak sağlar,
- * UML gösterimi nesneye dayalı yazılım mühendisliğine dayanır.

UML Faydası

- * Yazılımın geniş bir analizi ve tasarımı yapılmış olacağından kodlama işlemi daha kolay ve doğru olur
- * Hataların en aza inmesine yardımcı olur
- * Geliştirme ekibi arasındaki iletişimi kolaylaştırır
- * Tekrar kullanılabilir kod sayısını artırır
- * Tüm tasarım kararları kod yazmadan verilir
- * Yazılım geliştirme sürecinin tamamını kapsar
- * “resmin tamamını” görmeyi sağlar

Grafiksel Gösterimler

UML Grafiksel Gösterimler

Yapısal
Diyagramlar

Davranışsal Diyagramlar

Sınıf

Nesne

Bileşene

Dağılım

Kullanım
Senaryosu

Ardışık

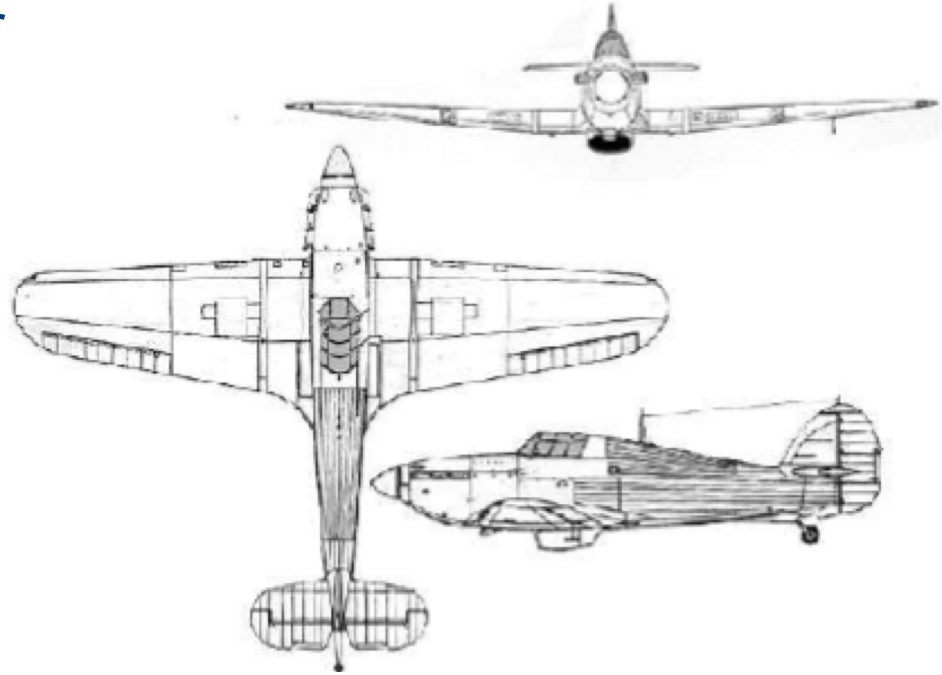
İşbirliği

Durum

Etkinlik

Grafiksel Gösterimler

- * Bu grafiksel gösterimler aynı şeye farklı şekillerde bakabilmeyi sağlar.



4+1 bakış

- * Bir yazılım sistemi oluşturulurken sadece tek boyutta analiz ve modelleme yapılmaz.
- * Bu amaçla; yazılım geliştirme sürecinin farklı aşamalarında farklı UML diyagramlarını kullanmak gerekmektedir.
- * 4+1 bakış, bu diyagramları sınıflandırmak ve yazılım yaşam çevrimindeki kullanım yerlerini ortaya koymak için kullanılan bir kavramdır.

4+1 bakış

1

- Kullanıcı Bakışı (User View)

2

- Yapısal Bakış (Structural View)

3

- Davranış Bakışı (Behavioral View)

4

- Gerçekleme Bakışı (Implementation View)

5

- Ortam Bakışı (Environment View)

4+1 bakış

1

- Kullanıcı Bakışı (User View)

2

- Yapısal Bakış (Structural View)

3

- Davranış Bakışı (Behavioral View)

4

- Gerçekleme Bakışı (Implementation View)

5

- Ortam Bakışı (Environment View)

4+1 bakış

1-) Kullanıcı Bakışı (User View)

- * Müşteri gereksinimlerini ortaya koymak ve müşteriye, sistemi tanıtmak amacı ile kullanılan bakış açısıdır.
- * Bazı kaynaklarda; **kullanım senaryosu (use case)** bakışı olarak da açıklanmaktadır.
- * Bu amaçla "kullanım senaryosu" (use case) diyagramları kullanılmaktadır.

4+1 bakış

1

- Kullanıcı Bakışı (User View)

2

- Yapısal Bakış (Structural View)

3

- Davranış Bakışı (Behavioral View)

4

- Gerçekleme Bakışı (Implementation View)

5

- Ortam Bakışı (Environment View)

4+1 bakış

2-) Yapısal Bakış (Structural View)

- * Sistemin nelerden meydana geldiğini gösteren bakış açısıdır.
- * Bu amaçla; "sınıf" (class) diyagramları ve "nesne" (object) diyagramları kullanılmaktadır.

4+1 bakış

1

• Kullanıcı Bakışı (User View)

2

• Yapısal Bakış (Structural View)

3

• Davranış Bakışı (Behavioral View)

4

• Gerçekleme Bakışı (Implementation View)

5

• Ortam Bakışı (Environment View)

4+1 bakış

3-) Davranış Bakışı (Behavioral View)

- * Sistemin dinamik yapısını ortaya koyan bakış açısıdır.
- * Bu amaçla; «**ardışık**» (*sequence*), "**işbirliği**" (*collaboration*), "**durum**« (*state*) ve «**etkinlik**" (*activity*) diyagramları kullanılmaktadır.

4+1 bakış

1

• Kullanıcı Bakışı (User View)

2

• Yapısal Bakış (Structural View)

3

• Davranış Bakışı (Behavioral View)

4

• Gerçekleme Bakışı (Implementation View)

5

• Ortam Bakışı (Environment View)

4+1 bakış

4-) Gerçekleme Bakışı (Implementation View)

- * Sistemin alt modüllerini ortaya koyan bakış açısıdır.
- * Bu amaçla; "bileşen" (component) diyagramları kullanılmaktadır.

4+1 bakış

1

• Kullanıcı Bakışı (User View)

2

• Yapısal Bakış (Structural View)

3

• Davranış Bakışı (Behavioral View)

4

• Gerçekleme Bakışı (Implementation View)

5

• Ortam Bakışı (Environment View)

4+1 bakış

5-) Ortam Bakışı (Environment View)

- * Donanımın, fiziksel mimarisinin ortaya konduğu bakış açısıdır.
- * Bu amaçla; "dağıtım" (deployment) diyagramları kullanılmaktadır.

Sınıf Diyagramları

- * Sınıf Diyagramları UML 'in en sık kullanılan diyagram türüdür.
- * Sınıflar nesne tabanlı programlama mantığından yola çıkarak tasarlanmıştır.
- * Sınıf diyagramları bir sistem içerisindeki nesne tiplerini ve birbirleri ile olan ilişkileri tanımlamak için kullanılırlar.

Sınıf Diyagramları

- * Sınıfların
 - * bir adı
 - * nitelikleri ve
 - * İşlevleri vardır
 - * Bunlara ek olarak
 - * “notes”
 - * “Constraints”



Sınıf Diyagramları

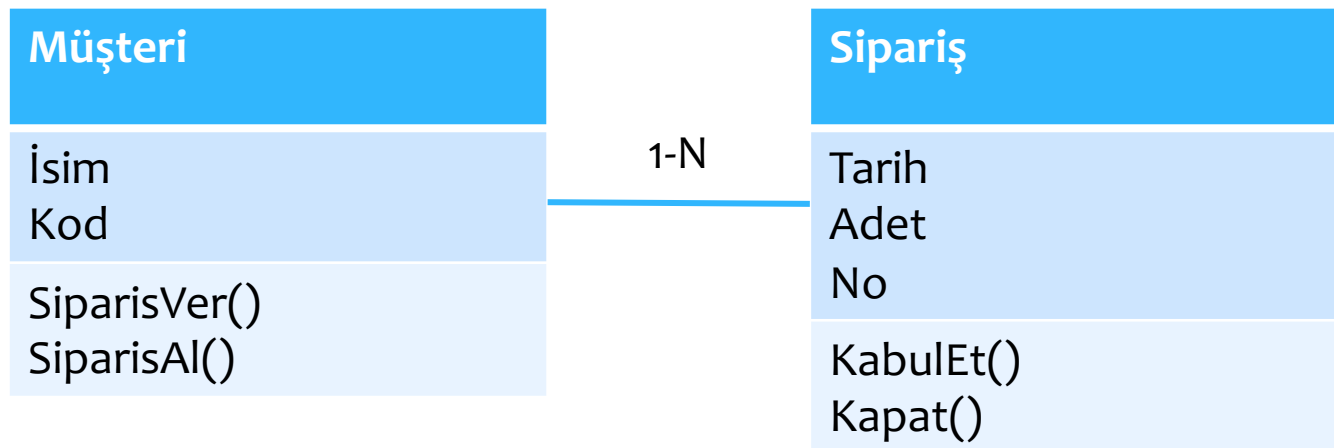
* Örnek:

SınıfAdı
Özellik 1 : tür 1 Özellik 2 : yaş="19" ...
İşlev 1() İşlev 2(parametreler) İşlev 3():geri dönen değer tipi ...

Müşteri
İsim Kod
SiparisVer() SiparisAl()

Sınıf Diyagramları

- * Sınıflar arasındaki ilişkiyi göstermek için iki sınıf arasına düz bir çizgi çekilir.
- * İlişkiyi gösteren çizginin üzerine ilişkinin türü yazılır.

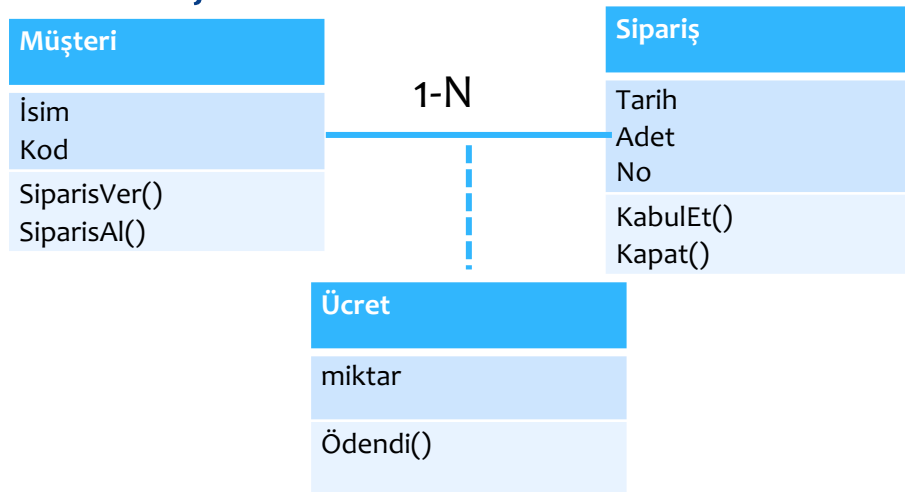


Sınıf Diyagramları

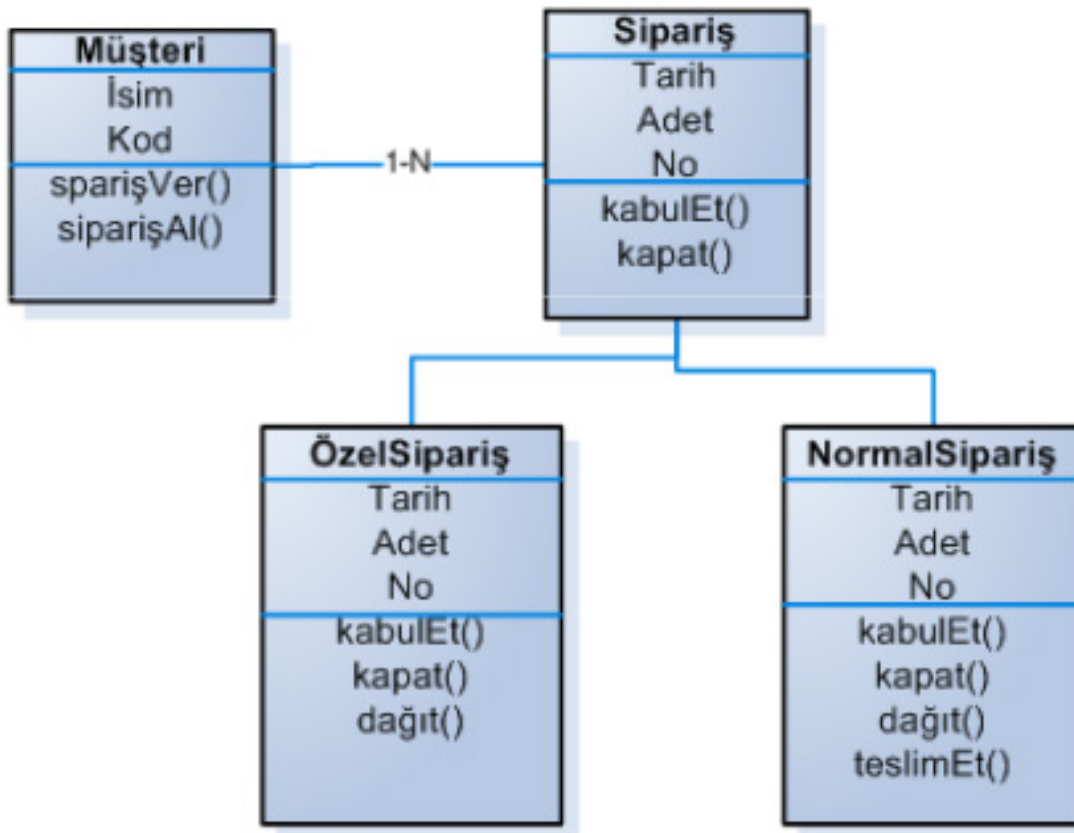
- * Bazı durumlarda sınıflar arasındaki ilişki, bir çizgiyle belirtebilecek şekilde basit olmayabilir.
- * Bu durumda ilişki sınıfları kullanılır.
- * İlişki sınıfları bildiğimiz sınıflarla aynıdır.
- * Sınıflar arasındaki ilişki eğer bir sınıf türüyle belirleniyorsa UML ile gösterilmesi gerekir.

Sınıf Diyagramları

- * Müşteri ile Sipariş sınıfı arasında ilişki vardır. Fakat müşteri satın alırken Ücret ödemek zorundadır
- * Bu ilişkiyi göstermek için Ücret sınıfı ilişki ile kesikli çizgi ile birleştirilir.



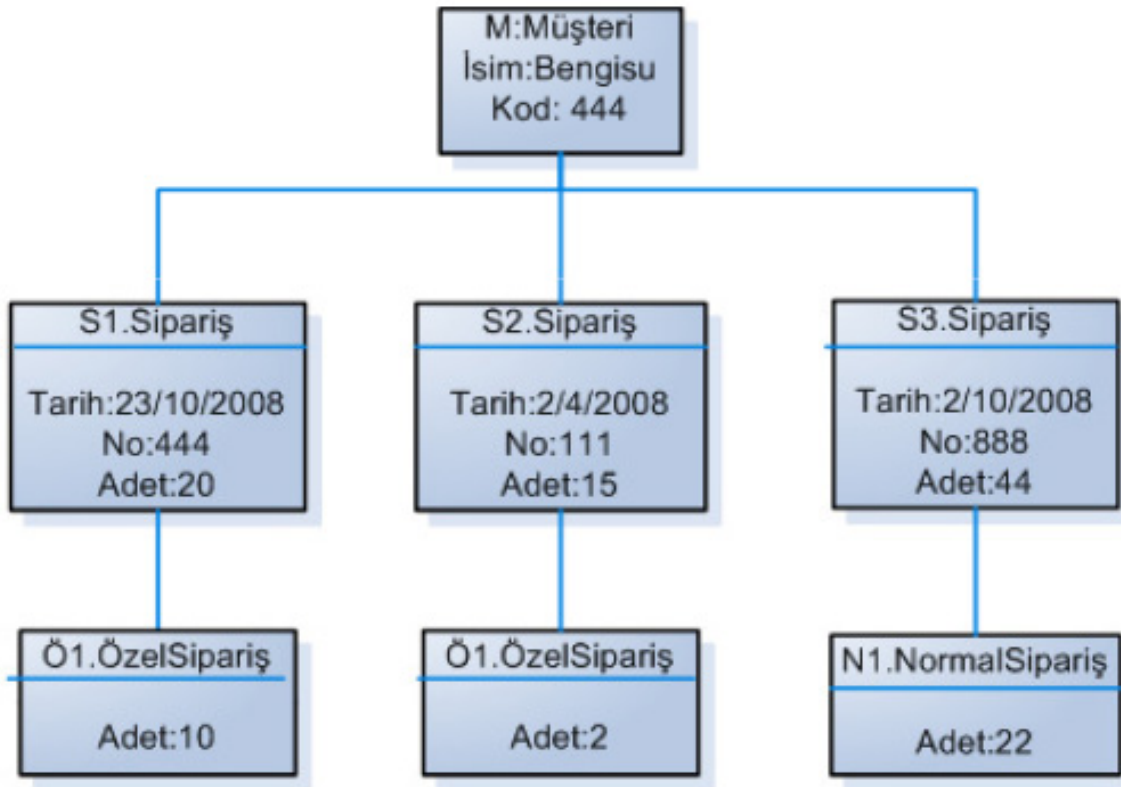
Örnek Sınıf Diyagramı



Nesne Diyagramları

- * Nesne Diyagramları nesneler ve aralarındaki bağıntıları gösterirler.
- * Nesneler, sınıfların somut örnekleri olduğundan sınıf özelliklerinin değer almış halidirler.

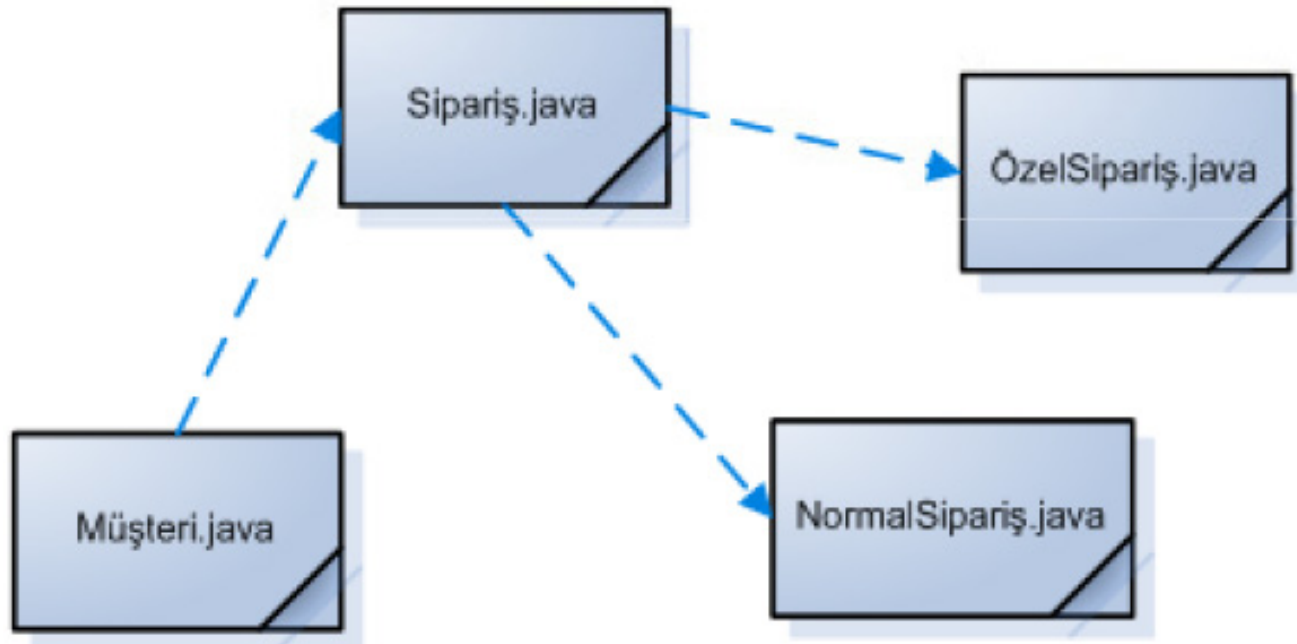
Örnek Nesne Diyagramı



Bileşen Diyagramları

- * Bileşen Diyagramları yazılım sistemine daha yüksek bir seviyeden bileşenler seviyesinden bakabilmeyi sağlar.
- * Bunlar sistemdeki sınıflar, çalıştırılabilen program parçaları kütüphane bileşenleri veya veritabanı tabloları olabilir.
- * Bu gösterim bileşenler, arayüzler ve bağımlılık ilişkilerinden oluşur ve sistemin fiziksel gösterimini sağlar.

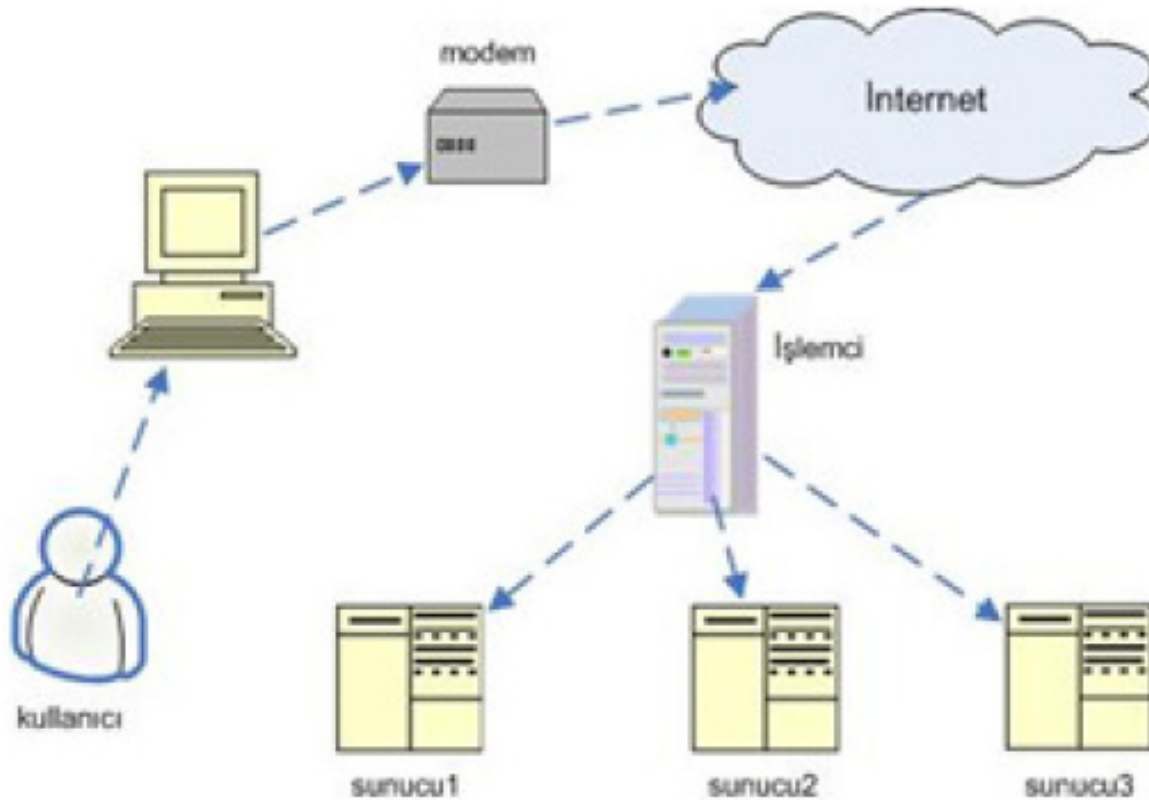
Örnek Bileşen Diyagramı



Dağılım Diyagramları

- * Sistemin çalışma platformundaki durumlarını gösterirler.
- * Sistemdeki yürütülebilen parçalar, kütüphaneler, tablolar ve dosyalar gibi bileşenlerin dağılımını belirler.
- * Bu diyagramlar, sistem donanım mimarisinin gösterilmesi, gerekli donanım bileşenlerinin tanımlanması amacıyla kullanılabilirler.

Örnek Dağılım Diyagramı



Kullanım Senaryosu Diyagramları

- * Kullanım senaryosu diyagramları sistemin işlevsel gereksinimlerinin ortaya çıkarılması için kullanılır.
- * Sistemin kullanıcısının bakış açısıyla modellenmesi amacıyla kullanılır.
- * Kullanım senaryosu diyagramları sistemin kabaca ne yaptığı ile ilgilenir, kesinlikle nasıl ve neden yapıldığını incelemmez.

Kullanım Senaryosu Diyagramları

- * Kullanım senaryosu
 - * aktörler (sistem dışı aktörler –kullanıcılar veya diğer sistemler)
 - * senaryolar (kullanıcı tarafından başlatılan çeşitli olaylar dizisi)

Kullanım Senaryosu Diyagramları



Kullanıcı

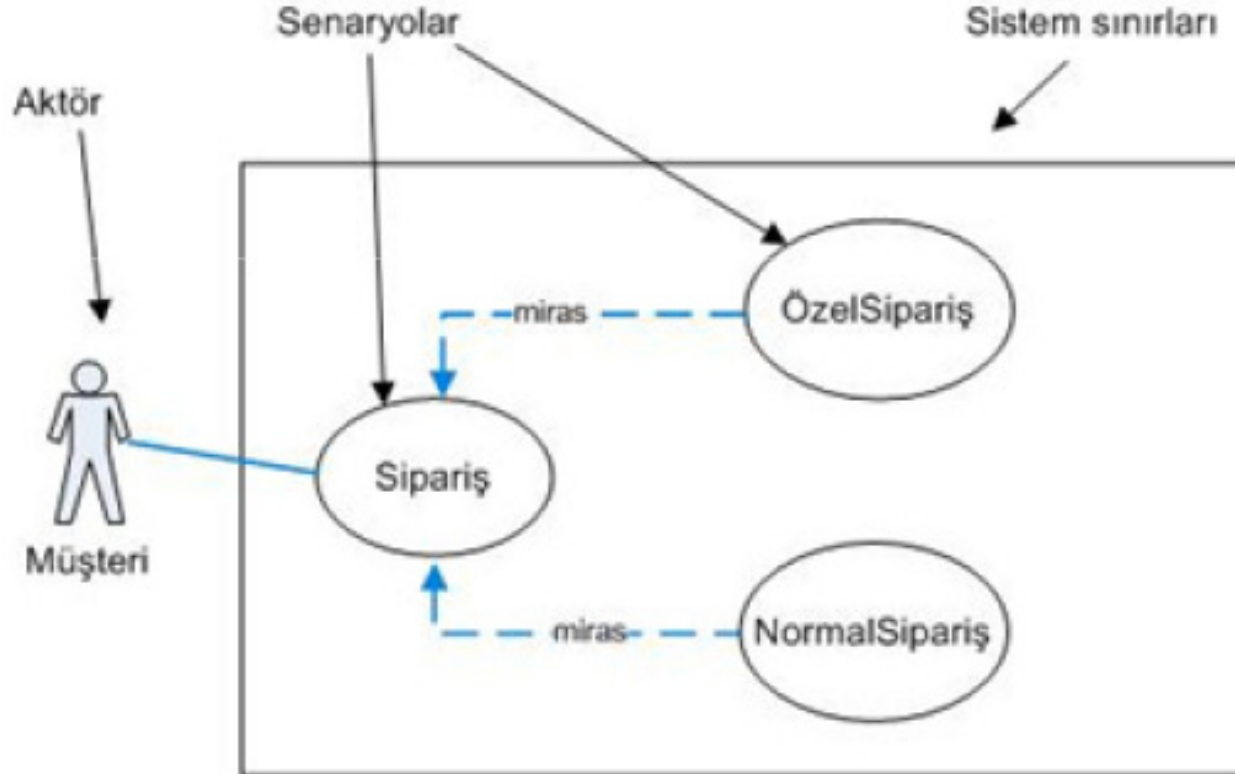
Aktör

- * Aktör genellikle “insan” olarak düşünülmele birlikte başka “sistem” ve “donanım” da olabilir.
- * Aktör sistemi “uyarır” ,işlevleri haricen “tetikler”(aktif) yada sistemden “uyarıcı alır”(pasif).
- * Aktör sistemin parçası değildir, “harici” dir.

Kullanım Senaryosu Diyagramları

- * Aktörler belirlenir
- * Her aktörün “ne” yapmak istediği belirlenir
- * Her aktörün “ne” si için “ana senaryo özeti” çıkarılır
- * Tüm sistemin ana senaryo özetleri incelenir, ayıklanır, birleştirilir
- * Her senaryo için ana işlem adımları belirtilir.

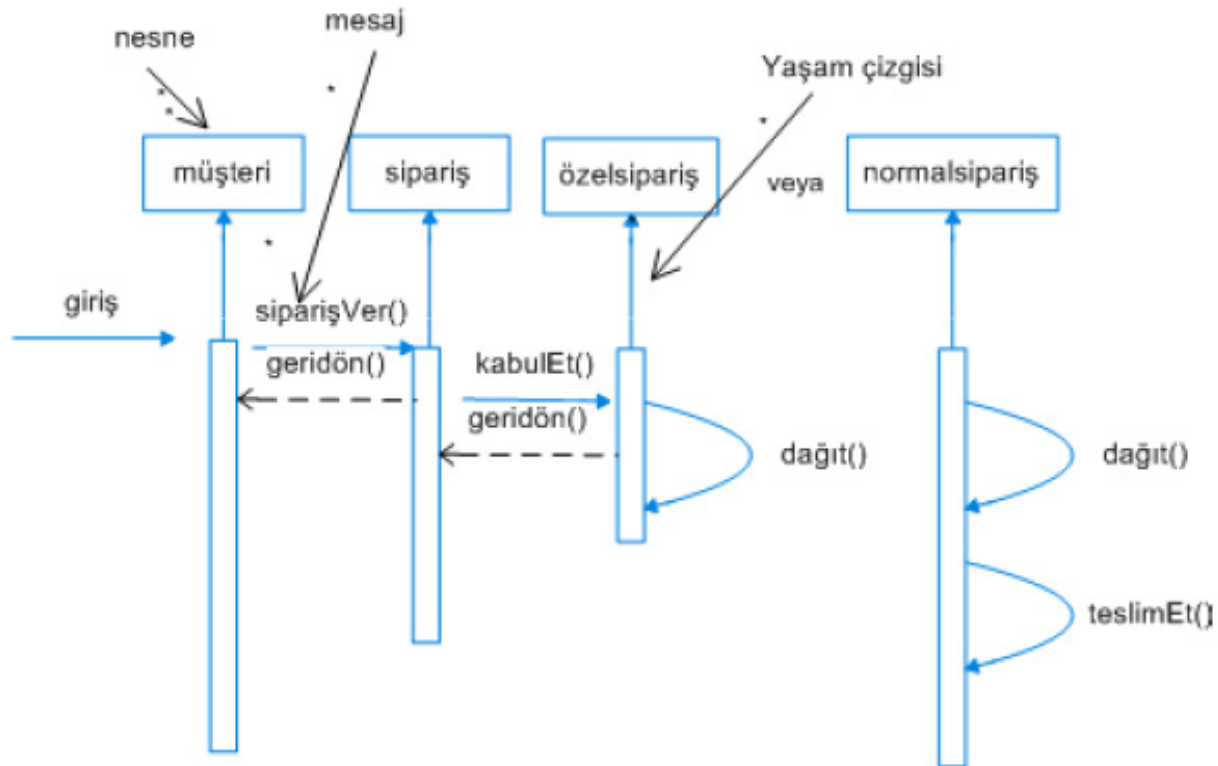
Örnek Kullanım Senaryosu Diyagramı



Ardışık Diyagramlar

- * Sistem içindeki nesnelerin zaman içindeki ardışık aksiyonlarını gösterirler
- * Sistemin dinamik bir resmini çizer
- * Aktörün hayat süresi boyunca gerçekleştirdiği işlemler gösterilir.
 - * Aksiyonlar->dikdörtgen
 - * Etkileşimler->mesajlar

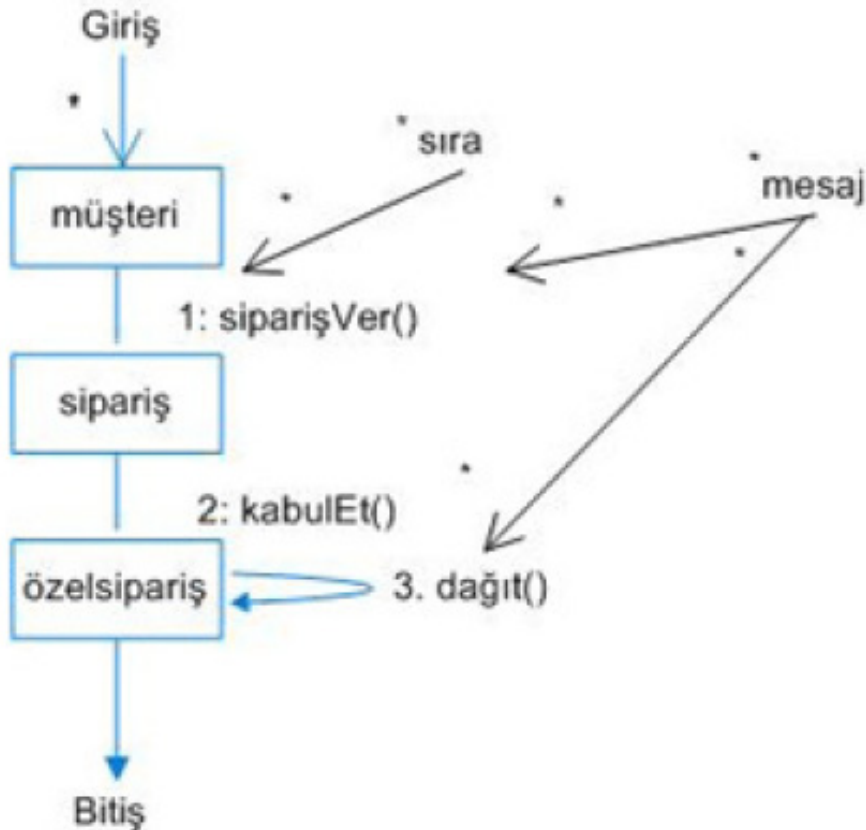
Ardışık Diyagramlar



İşbirliği Diyagramları

- * Parçaların bütünü nasıl oluşturduğunu ve sistemin dinamik davranışını göstermek amacıyla kullanılır.
- * Nesneler arasındaki bağıntıları sıralı mesajlar şeklinde gösterirler
- * Ardışık diyagramlardan zaman kavramının olmayışı ile farklılık gösterirler.

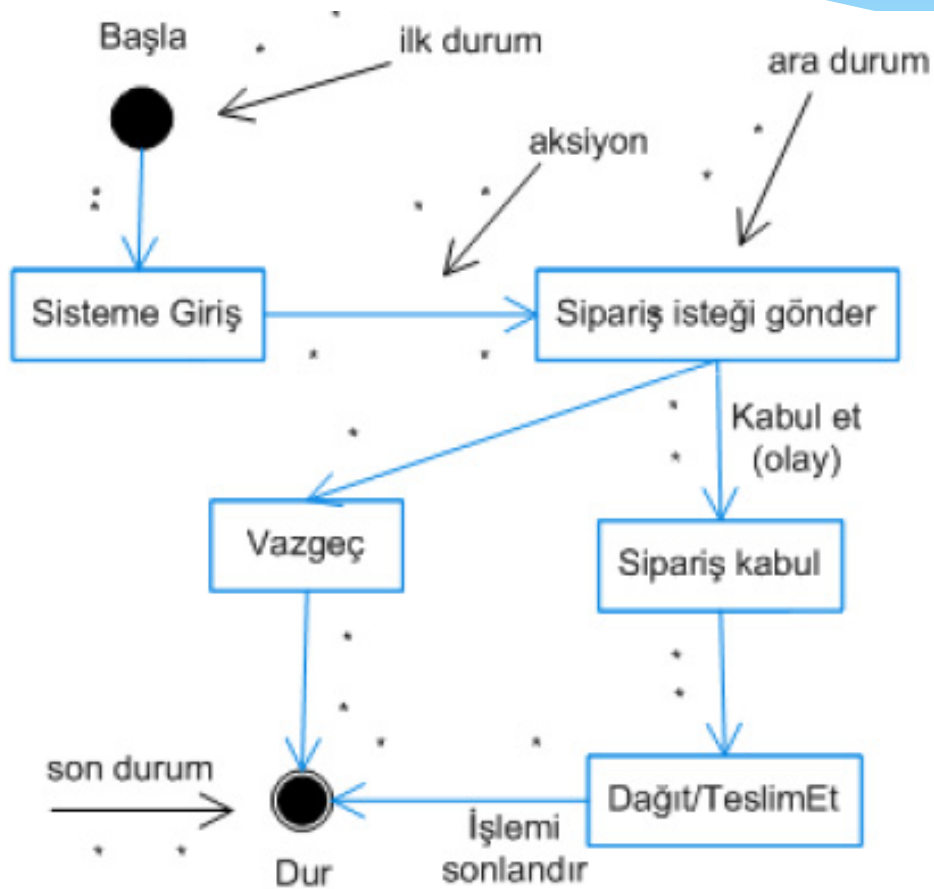
İşbirliği Diyagramları



Durum Diyagramları

- * Sistemdeki nesnelerin anlık durumlarını göstermek amacıyla kullanılırlar.
- * Sistemin küçük alt sistemlere veya nesnelere ilişkin dinamik davranışlarının ortaya çıkartılması amacıyla yararlanılır
- * Sistemin durumları ve bunların birbirlerini tetikleme ilişkileri belirtilir

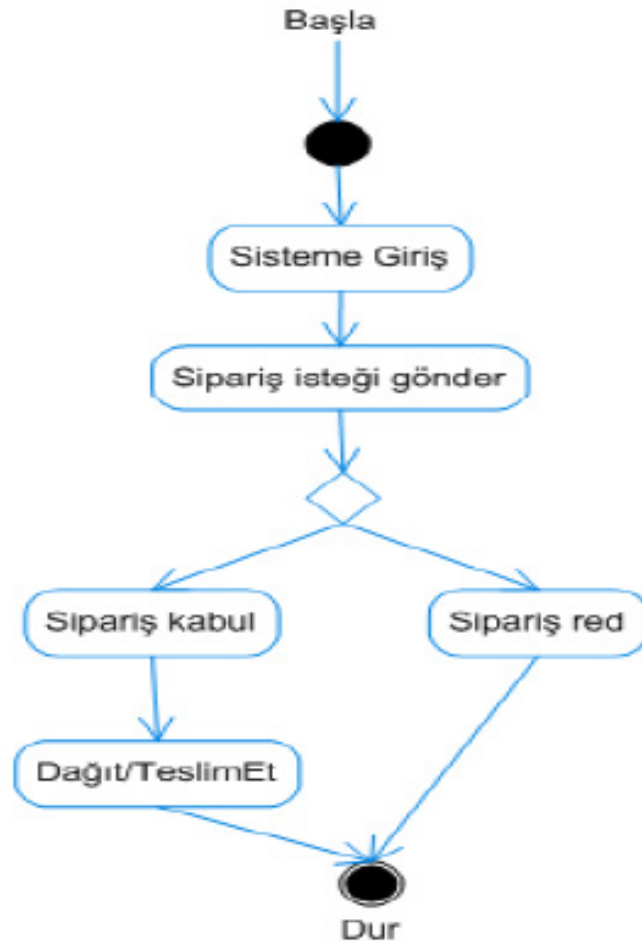
Durum Diyagramları



Etkinlik Diyagramları

- * Sistemdeki nesnelerin faaliyetlerini göstermek amacıyla kullanılırlar
- * Durum diyagramlarının bir alt kümesi olarak değerlendirilen etkinlik diyagramları, iş akışlarının grafiksel gösteriminde kullanılırlar.

Etkinlik Diyagramları



Kaynakça

- * Algan, S. (2002).
<http://www.csharpnedir.com/articles/read/?id=6>
- * Kurt, B. Ders notları:
http://web.itu.edu.tr/bkurt/Courses/oose/ooswe_release_1_o_6.pdf
- * Bozkır, S. <http://www.slideshare.net/aselmanb/uml-ile-modelleme>
- * Tokdemir, G. ve Çağıltay, N. E. (2010). *Veritabanı Sistemleri Dersi*. Seçkin yayıncılık, Ankara.