**UML'e giriş**

*Unified Modelling Language* (UML) yazılım sistemlerinin olgularını tanımlamak, görselleştirmek, oluşturmak ve dokümante etmek için kullanılan bir standartlar dilidir. UML, büyük ve karmaşık sistemlerin modellenmesinde başarısı ispatlanmış deneyimlerin bir araya gelmesiyle oluşmuştur ve *Object Oriented Software* (OOS) ve yazılım geliştirme sürecinin önenli bir parçasıdır. UML, yazılım projelerinin tasarımını daha çok grafiksel öğeler ile belirtir. UML kullanımı ekip içerisindeki iletişimi güçlendirir, potansiyel tasarımların ortaya çıkmasını sağlar ve yazılımın mimari tasarımını ortaya koyar.

**UML’in amaçları**

UML’in başlıca amaçları:

1. Kullanıcılara geliştirip paylaşabilecekleri anlamlı bir göresel modelleme dili sunmak.
2. Temel kavramları genişletmeye yönelik olarak genişleyebilir ve özelleştirilebilir bir mekanizma sunmak.
3. Uygulama geliştirme dillerinden ve geliştirme süreçlerinden bağımsızlığı sağlamak.
4. Modelleme dillerini anlamak üzere biçimsel bir zemin hazırlamak.
5. Bileşen, pattern, framework ve bir arada yürütülen projelerin ileri seviyede geliştirilmesine destek sağlamak.

**UML Diyagram tipleri:**

Aslında UML tipleri burada bahsedeceklerimizden çok daha fazladır. Fakat giriş seviyesinde ortalama bir programcının bilmesi gereken ve en sık kullanılan UML diyagramlarına öncelik vermek istedim. Diğer diyagramları da sıra geldikçe açıklamaya çalışacağım.

Use case diyagramı: Aktörler ve *use case*’ler arasındaki ilişkiyi gösterir.

Sınıf diyagramı: *Sınıf*, *package* ve *nesne* gibi tasarım elemanlarını kullanarak sınıf yapısını ve içeriğini gösterir.

Etkileşim diyagramı:

* Dizge (Sequence) diyagramı: Etkileşimde bulunan nesnelerin zaman içindeki sıralanışını gösterir. Dikey boyut (zaman) ve yatay boyut (nesneler)’tan oluşur.
* İşbirliği (Collaboration) diyagramı: Nesnelerin çevreleri ile kurduğu etkileşimleri ve birbirlerine olan bağlantılarını gösterir. Mesajların sırası numaralar ile belirtilir.

Durum (State) diyagramı: Bir nesnenin yaşam döngüsü içerisinde gelen isteklere verdiği tepkileri; yanıtları ve eylemleri ile birlikte durum sırasında gösterir.

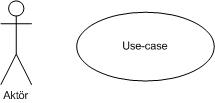
Etkinlik (Activity) diyagram: Çoğu durumun eylem durumu olduğu ve geçişlerin bir durumdaki eylemin sonuçlanması ile tetiklendiği özel bir *durum diyagramı* türüdür. Bu diyagram daha çok iç işlemler esnasındaki akışı gösterir.

Fiziksel diyagramlar:

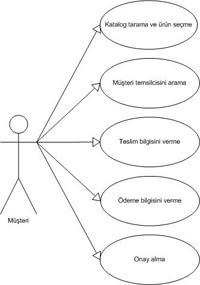
* Bileşen (Component) diyagramı: Kodun yüksek seviyede paketlenmiş yapısını gösterir. Bileşenler arasındaki bağlantılar kaynak kodu bileşeni, ikili (binary) kod bileşenler ve çalıştırılabilir bileşenleri de içerecek şekilde gösterilir. Bazı bileşenler derleme zamanında, bağlama zamanında, çalışma zamanında ya da birkaçında birden var olabilir.
* Dağıtım (Deployment) diyagramı: Çalışma zamanında işlenen elemanları ve yazılım bileşenlerini, süreçleri ve üzerinde yaşayan nesnelerin düzenleşimlerini gösterir.

**Use-case diyagramları**

Use case, bir kullanıcı ile sistem arasındaki etkileşimi göstermek üzere kullanılan senaryolar topluluğudur. Bir use case diyagramı aktörler ile use case’ler arasındaki ilişkiyi gösterir. Bu diyagram türünün iki temel elemanı aktör ve use case’dir.

[](http://1.bp.blogspot.com/_1EPXNbAy3L0/Spuprhvhy4I/AAAAAAAAACE/POxlto5LdIc/s1600-h/uml01.jpg)

Aktör, modellemekte olduğumuz sistem ile etkileşimde bulunan bir kişi ya da diğer bir sistemi temsil eder. Use case ise kullanıcının bir işlemini yerine getirmek üzere yapılan eylemleri yapan sistemin, dışarıdan bir görünüşüdür.  
  
**Use case diyagramları nasıl çizilir:**  
  
UML diyagramlarının çizimi kolay bir iş olmasına rağmen örnek olarak çok basit bir model ele alınmıştır.   
  
Bunun için önce bir işlemi tamamlamak için kullanıcının ilerleyeceği süreçler sıralanır. Örneğin bir firmaya sipariş verecek olan kişi aşağıdaki adımları izleyecektir.  
  
1. Katalog tarama ve ürün seçme  
  
2. Müşteri temsilcisini arama  
  
3. Teslimat bilgisini verme  
  
4. Ödeme bilgilerini verme  
  
5. İşlemi onaylatma  
  
Bahsedilen bu adımların use case diyagramı aşağıdaki gibi olmaktadır.

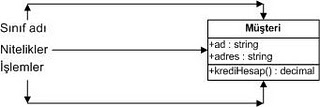
[](http://4.bp.blogspot.com/_1EPXNbAy3L0/Spupw5DvRkI/AAAAAAAAACM/2O5gYMt6zK4/s1600-h/uml02.jpg)

Bu örnekte aktör olarak müşteri gösterilmiştir çünkü sipariş sürecini yürüten kişi müşteridir. Diyagram, yukarıda listelenen adımları alır ve müşterinin gerçekleştirebileceği eylemler olarak gösterir. Bu diyagrama müşteri temsilcisi de dahil edilmelidir, çünkü bu süreçte o da etkileşimde bulunmaktadır.  
  
Yukarıda gösterilen diyagramdan sipariş sürecinin işleyişi kolaylıkla çıkarılabilmektedir. Sistem gereken eylemleri yerine getirebilmek için listelenen bütün use case’leri gerçekleştirmelidir.

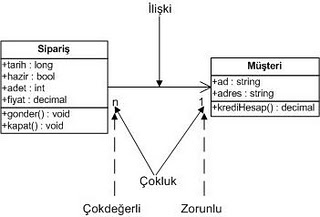
**Sınıf diyagramı**

Sınıf diyagramları bir sistem içerisindeki nesne tiplerini ve birbirleri ile olan ilişkileri tanımlamak için kullanılırlar. Sınıf diyagramları, sınıf yapısını ve içeriğini sınıflar, paketler ve nesneler kullanarak modeller ve bir sistemi 3 farklı perspektiften; kavramsal, tanımsal ve yaşamsal açıdan ele alır.

Sınıf’lar üç şeyden oluşur: bir isim, nitelikler (property) ve işlemler (method).

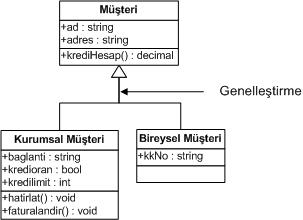
[](http://1.bp.blogspot.com/_1EPXNbAy3L0/SpuyCiS5F5I/AAAAAAAAACU/o0VjirP478o/s1600-h/class_01.jpg)

Sınıf diyagramları aynı zamanda içerme, kalıtım, ilişki ve diğer bağlantıları ifade eder. Aşağıda bir ilişki bağlantısı gösterilmektedir.

[](http://2.bp.blogspot.com/_1EPXNbAy3L0/SpuyHfgHh9I/AAAAAAAAACc/v3eRJixgUjA/s1600-h/class_02.jpg)

İlişki bağlantıları bir sınıf diyagramdaki en genel bağlantıdır. İlişki sınıf’ın örnek (instance)’leri arasındaki bağlantıları gösterir. Örneğin Sipariş sınıfı Müşteri sınıfı ile ilişkilidir. Çokluk ilişkisi bu ilişkiye dahil olabilecek nesne sayısını temsil eder. Örnekte Sipariş nesnesi sadece bir Müşteriye ilişkilendirilebilirken bir Müşteri birden fazla Sipariş ile ilişkilendirilebilir.

Sınıf diyagramlarında sıklıkla kullanılan bir diğer ilişki de genelleştirmedir. Genelleştirme iki sınıfın biribirine benzer olup da sadece küçük farklarla ayrıldığı durumlarda kullanılır. Aşağıda bir genelleştirme örneği görülmektedir.

[](http://4.bp.blogspot.com/_1EPXNbAy3L0/SpuyMPLmJQI/AAAAAAAAACk/UUfJjdM_7XU/s1600-h/class_03.jpg)

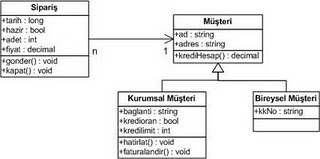
Bu örnekte Kurumsal Müşteri ve Bireysel Müşteri sınıflarında ad ve adres gibi alanlar ortak olmakla birlikte her sınıfın kendine özgü alan ve metodları da mevcuttur. Müşteri sınıfı ise her iki sınıfın genel yapısını temsil eder.

**Sınıf Diyagramları nasıl çizilir:**

Sınıf diyagramları en zor çizilen UML diyagramları arasındadır. Detaylı ve kullanışlı bir diyagram çizebilmek uzun süreli bir UML ve OO bilgisi gerektirir. Burada ise sadece genel bir bakış ortaya koyan, temel bir örnekle yetinilecektir.

Sınıf diyagramı çizmeden önce, çizilecek diyagramın sistemi her üç perspektiften de (kavramsal, tanımsal ve yaşamsal) göstermesi gerektiğini unutmayın. Diyagram tek bir perspektife odaklanmamalı, hepsinin bir arada nasıl çalışabileceğini göstermelidir.

Bir sınıfı tasarlarken hangi özelliklere ve işlemlere sahip olması gerektiğini göz önünde bulundurun. Daha sonra bu sınıftan türetilen örneklerin birbirleri ile nasıl etkileşeceğini düşünün. Bunlar her ne kadar bir sınıf diyagram çizmenin ilk adımları olsa da bu temel çizim ile bile sistemin genel yapısı ortaya koyulabilir.

[](http://2.bp.blogspot.com/_1EPXNbAy3L0/SpuyQk3GLqI/AAAAAAAAACs/CP-1rHOT5NE/s1600-h/class_04.jpg)

**Etkileşim diyagramı**

Etkileşim diyagramları, use case’lerin görevleri yerine getirmek üzere nesnelerin nasıl etkileştiklerini modeller. İki tür etkileşim diyagramı vardır; dizge ve ortak çalışma diyagramları.

**Etkileşim diyagramları ne zaman kullanılır:**

Etkileşim diyagramları bir use case içerisindeki farklı nesnelerin davranışlarını modellemek istediğimizde kullanılır ve davranışı oluşturmak üzere nesnelerin ne şekilde işbirliği yaptıklarını ortaya koyar. Etkileşim diyagramları davranış ile derinlemesine bilgi vermediğinden belirli bir nesnenin farklı use case’ler için nasıl davrandığını görmek için durum diyagramlarını kullanınız.

**Etkileşim diyagramları nasıl çizilir:**

Dizge diyagramları, Ortak çalışma diyagramları ve ikisi bir arada nesnelerin bir use case içerisindeki etkileşimlerini göstermek üzere kullanılabilir. Dizge diyagramları genel olarak oluşan olayların sırasını gösterir. Ortak çalışma diyagramları ise nesnelerin statik olarak birbirlerine bağlantılarını gösterir. Her iki diyagram da ortak elemanlar içerirler ve çizimleri kolaydır.

**Dizge (Sequence) diyagramı**

Dizge diyagramları bir use case içerisindeki nesnelerin davranışlarını ve birbirlerine ilettikleri mesajları gösterir. Diyagram soldan sağa ve yukarıdan aşağı okunur. Bir dizge diyagramı temel olarak nesnelerden, mesajlardan ve zaman eksenlerinden oluşur. Dizge diyagramları, tasarımı yapan kişi ya da ekibin bakış açısına, gerekliliklerine ve modülün tüm çalışanlar tarafından bilinirliğine göre değişebilmektedir. Kolayca anlaşılabilecek sistemler için basit bir dizge diyagramı yeterli olabilecekken yeni geliştirilen bir sistem için her ayrıntıyı gösteren bir siyagram gerekebilmektedir. Bu ayrım gözetildiğine dizge diyagramları iki türe ayrılmaktadır. Genellike basit bir şemanın çizildiği, ve en iyi olasılıkları ele alan “*best case*” senaryo üzerinden yürüyen **örnek (instance)** diyagramı ve modelin her yönüyle ele alındığı ve daha karışık bir model olan **genel (generic)** diyagram.

Nesneler, diğer UML diyagramlarında olduğu gibi belirli görevleri gerçekleştirmek üzere tanımlı yapı bloklarıdır. Dizge diyagramında nesneler, diyagramında fiziksel olarak en üstten alta ve soldan sağa okunur. Her bir nesnenin altından çıkan kesikli çizgi zaman çizgisini (*timeline*) göstermektedir. Kesikli çizgi zaman akışını gösterirken üzerinde bulunan ince uzun dikdörtgenler o nesnenin zaman içerisinde meydana getirdiği aktivitelerdir. Her ne kadar teorik olarak dikdörtgenin uzunluğu aktivitenin uzunluğu ile doğru orantılı olsa ve da genellikle bu kurala pek uyulmaz.

Mesajlar bir nesnenin zaman çizgisinden diğer bir nesnenin zaman çizgisine doğru çizilen ok şekillerinden oluşmaktadır. Nesnelerin birbirleri ile haberleşmesi, birbirlerine gönderdikleri komutları temsil eden bu oklarla gösterilir. Bir nesne başka bir nesnenin dışında kendine de mesaj gönderebilir (öz yineleme - recursion). Bunun yanında, Actor gibi nesne olmayan UML bileşeni de nesnelere meaj gönderebilir.

Dizge diyagramlarında 3 çeşit mesaj tip kullanılmaktadır.

Basit (Simple) mesaj: Bu mesaj tipi bir nesnenin akış kontrolünü diğer bir nesneye verdiği durumlarda kullanılır. Sık kullanılan bir mesaj tipi değildir. Gösterimi aşağıdaki gibi olmaktadır:

[http://2.bp.blogspot.com/_1EPXNbAy3L0/Spu5Xn5ylaI/AAAAAAAAAC0/dAVXPdUw6P4/s320/seq01.jpg](http://2.bp.blogspot.com/_1EPXNbAy3L0/Spu5Xn5ylaI/AAAAAAAAAC0/dAVXPdUw6P4/s1600-h/seq01.jpg)

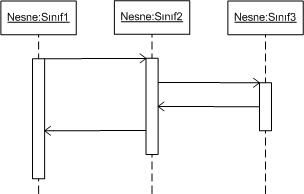
Senkron (Synchronous) mesaj: Bir nesnenin mesajı gönderdikten sonra, zaman çizelgesinde devam edebilmek için karşı nesneden cevap beklemesi gereken durumlarda kullanılır. Varsayılan mesaj tipi olduğundan sıklıkla kullanılmaktadır. Gösterimi aşağıdaki gibidir:

[http://1.bp.blogspot.com/_1EPXNbAy3L0/Spu5cw44PeI/AAAAAAAAAC8/SbkvF13tLo0/s320/seq02.jpg](http://1.bp.blogspot.com/_1EPXNbAy3L0/Spu5cw44PeI/AAAAAAAAAC8/SbkvF13tLo0/s1600-h/seq02.jpg)

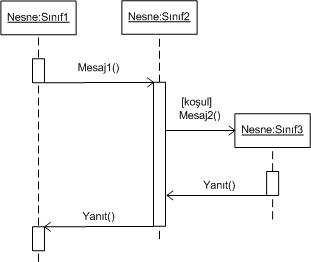
Asenkron (Asynchronous) mesaj: Senkron mesajların aksine bir nesnenin mesajı gönderdikten sonra, zaman çizelgesinde devam edebilmek için karşı nesneden cevap beklemesi gerekmediği durumlarda kullanılır. Sıklıkla asenkron işlemesi gereken komut zincirlerinde kullanılmaktadır. Gösterimi aşağıdaki gibidir:

[http://3.bp.blogspot.com/_1EPXNbAy3L0/Spu5juKD3YI/AAAAAAAAADE/XQ6zqlnL9DQ/s320/seq03.jpg](http://3.bp.blogspot.com/_1EPXNbAy3L0/Spu5juKD3YI/AAAAAAAAADE/XQ6zqlnL9DQ/s1600-h/seq03.jpg)

Aşağıdaki örnek, Sınıf1 nesnesinin Sınıf2 nesnesine bir mesaj göndermesi ile başlamakta olup Sınıf3 son mesajı alıncaya kadar farklı nesneler arasındaki mesaj alışverişi ile devam etmektedir.

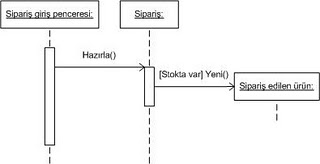
[](http://3.bp.blogspot.com/_1EPXNbAy3L0/Spvjbiw3n8I/AAAAAAAAADM/vvchsveuZT8/s1600-h/seq04.jpg)

Aşağıda ise biraz daha karmaşık bir örnek görülmektedir. Açık mavi dikey kutular nesnenin aktif olduğu süreci gösterirken yeşil kesikli çizgi ise nesnenin ömrünü göstermektedir. Yeşil dikey kutular ise o anda kontrolü elinde tutan nesneyi ifade eder. Turuncu kutu ise nesnenin yok edildiğini belirtir. Bu diyagram aynı zamanda bir nesneye mesaj gönderilme koşullarını gösterir. Koşul, mesajın yanına köşeli parantez içerisinde yazılır. Örneğin aşağıdaki örnekte Sınıf2’nin Sınıf3’e mesajını gönderebilmesi için [koşul] ifadesi sağlanmalıdır.

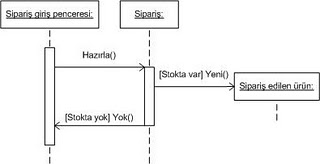
[](http://3.bp.blogspot.com/_1EPXNbAy3L0/SpvjiwR-9RI/AAAAAAAAADU/_3-5tc_4gRk/s1600-h/seq05.jpg)

Sonraki diyagram, bir sipariş alımı için olan dizge diyagramının başlangıç kısmını göstermektedir. “Sipariş Giriş Penceresi” adında bir nesne yaratılmakta ve siparişin hazırlanması için Sipariş nesnesine mesaj göndermektedir. Diyagramda nesne isimlerinden sonra iki nokta üst üste konulduğuna dikkat ediniz. Şekilde nesnelerin türedikleri sınıf adları belirtilmemiştir. Buna rağmen “nesneAdı : sınıfAdı” biçiminin korunması için iki nokta üst üste işaretinin kullanılması gerektiğini unutmayınız.

Daha sonra Sipariş nesnesi ürünün stokta olup olmadığına bakmakta ve eğer [StoktaVar] koşulu sağlanırsa “Sipariş Edilen Ürün” nesnesine mesaj göndermektedir.

[](http://4.bp.blogspot.com/_1EPXNbAy3L0/SpvjqV3lrGI/AAAAAAAAADc/BtwtlVMFERw/s1600-h/seq06.jpg)

Bir sonraki diyagramda ise Sipariş nesnesine koşullu mesaj eklenmiştir. Eğer ürün [StoktaYok] koşulunu sağlarsa “Sipariş Giriş Penceresi” nesnesine ürünün kalmadığını belirten bir mesaj gönderir.

[](http://4.bp.blogspot.com/_1EPXNbAy3L0/Spvju3yfiAI/AAAAAAAAADk/tfEu1aHqVTo/s1600-h/seq07.jpg)

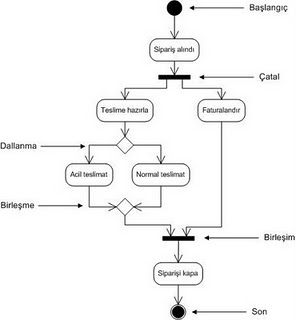
## 01 09 2009

### [Etkinlik (Activity) diyagramı](http://umlnedir.blogspot.com/2009/09/etkinlik-activity-diyagram.html)

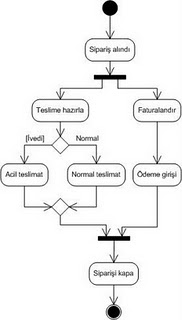
Etkinlik diyagramları, sistemin akış yönünden davranışını betimler. Etkinlik diyagramları bir şeyin durumlarını temsil etmesi niteliği ile durum diyagramlarına benzer. Fakat durum diyagramları gerçekleşen eylemler neticesinde oluşan durumları nitelerken etkinlik diyagramları ise koşullu ya da paralel ilerleyen eylemleri betimler.

**Etkinlik diyagramları nasıl çizilir:**

Etkinlik diyagramları sistem içerisindeki eylemlerin akışını gösterir. Diyagram yukarıdan aşağıya doğru okunur ve dallanmalar paralel eylemleri ve durumları belirtir. Dallanma, aynı anda birden fazla sürecin işlediğini gösterir. Aşağıdaki diyagramda 1.Aktivite’den sonra bir çatal yer almakta ve dallanma meydana gelmektedir. Bu, 2.Aktivite ve 3. Aktivite’nin aynı anda gerçekleştiğini göstermektedir. 2.Aktivite’den sonra da bir dallanma vardır. Dallanma belirli koşullar altında hangi eylemlerin gerçekleşeceğini gösterir. Tüm dallar bir noktada koşullu durumda işletilecek eylemlerin sonunu belirtmek üzere birleşirler. Tüm paralel aktiviteler sonuca ulaşmadan önce birleşmek zorundadır.

[](http://2.bp.blogspot.com/_1EPXNbAy3L0/Sp0EHsr4QII/AAAAAAAAAF0/OGPTv4gAFfg/s1600-h/act01.jpg)

Aşağıda, sipariş işleminin olası bir etkinlik diyagramı görülmektedir. Diyagram, sistem akışı içerisinde eylemlerin akışını göstermektedir. Sipariş alındıktan sonra eylemler iki farklı eylem kümesine ayrılmaktadır. Bir yönde sipariş hazırlanmakta ve gönderilmekte, diğerinde ise faturalandırma işlemi yerine getirilmektedir. Sipariş hazırlama dalında teslim metodu duruma göre tespit edilmektedir. Duruma göre Acil teslimat ya da Normal teslimat işlemi gerçekleştirilir. Son olarak paralel yürüyen aktiviteler siparişi kapatmak üzere birleşirler.

[](http://4.bp.blogspot.com/_1EPXNbAy3L0/Sp0EK58p2OI/AAAAAAAAAF8/JkSC797rdjM/s1600-h/act02.jpg)

Gönderen Utku Ozan ÇANKAYA zaman: [14:23](http://umlnedir.blogspot.com/2009/09/etkinlik-activity-diyagram.html) [0 yorum](http://umlnedir.blogspot.com/2009/09/etkinlik-activity-diyagram.html#comments) [http://img2.blogblog.com/img/icon18_edit_allbkg.gif](http://www.blogger.com/post-edit.g?blogID=7928591152747902024&postID=7659261389327902306&from=pencil)

### [Durum diyagramı](http://umlnedir.blogspot.com/2009/09/durum-diyagram.html)

Durum diyagramları bir sistemin davranışlarını modeller ve bir olay gerçekleştiğinde olası tüm durumları tanımlar. Her bir diyagram bir sınıfın tek bir nesnesini ele alır ve sistem içerisindeki farklı durumlarını irdeler. Bir durum diyagramı aşağıdaki elemanlardan oluşur:

Durum makinesi (State machine): Bir nesnenin bütün durumlarını bir şema halinde gösteren yapıdır.

Durum (State): Tek bir nesnenin herhangi bir durumuna verilen addır.

Olay (Event): Nesnenin durumları arasında geçişini sağlayan yordama olay denir.

Eylem (Action): Nesnenin bir durumdan diğer bir duruma geçtiğinde yaptığı işlere eylem denir. Eylem, çalışıtırılabilir herhangi bir durum olabilir.

Geçiş (Transition): Nesnenin durumları arasındaki, geçişin gösterilmesine geçiş denir. Bir nesnenin her durumu arasında bir ilişki olmayabileceğinden dolayı iki durum arasında bir geçiş olmak zorunda değildir. Bir geçişte 4 yapı vardır. Bu yapılardan ikisi hedef (target) ve kaynak (source)’tır. Geçişler daima kaynak durumdan hedefv duruma doğru yapılır. Üçüncü yapı, kaynak durumdan hedef duruma geçişi sağlayan olay ateşleyicisi (event trigger)’dir. Son yapı da nesnenin geçiş sonrası ne şekilde davranacağını belirleyen eylem (action)’dir.

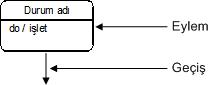
Özgeçiş (Self transaction): Bazen bir nesnenin gçişleri iki farklı durum arasında olmayıp, kendi üzerine döner. Daha net söylersek; geçişin kaynak ve hedef durumu aynı olur. Bu durum özgeçiş olarak adlandırılır.

İlk durum (Initial state): Programtaik olarak bir koda karşılık geşmese de ilk durum bir nesnenin ilk geçişinden hemen önceki durumunu temsil eder ve geçeceği ilk duruma geçişi sağlamak için bir ilk nokta teşkil eder ve durum kutusunda farklı olarak içi dolu daire olarak gösterilir.

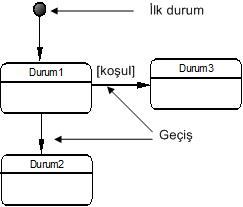
Son durum (Final state): Nesnenin ömrü bittiğinde aldığı son durumu ifade etmek için kullanılan semboldür. Bir nesne birden fazla son duruma sahip olabilir.

**Durum diyagramları nasıl çizilir:**

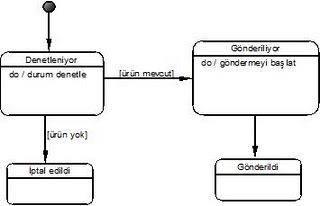
Durum diyagramları çok az sayıda eleman içerir. Temel elemanlar, nesnenin durumunu temsil eden kutular ve bir sonraki duruma geçişi temsil eden oklardır. Durum sembolünün eylem kısmı nesnenin o durumda ne yapacağını betimler.

[](http://4.bp.blogspot.com/_1EPXNbAy3L0/Spz3rIQ5-II/AAAAAAAAAFU/QdJ_ovT31_w/s1600-h/state01.jpg)

Tüm durum diyagramları, nesnenin ilk durumu ile başlar. Bu durum nesnenin ilk yaratıldığı andaki durumudur. İlk durumu takiben nesne durum değiştirmeye başlar. Eylemlerin koşuluna göre nesnenin geçeceği bir sonraki durum çeşitlenir.

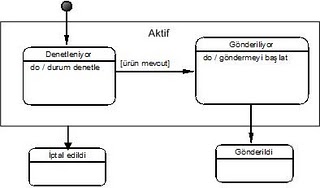
[](http://1.bp.blogspot.com/_1EPXNbAy3L0/Spz3wKguYgI/AAAAAAAAAFc/nHMDTgsDyS4/s1600-h/state02.jpg)

Aşağıda sipariş örneğine uygun bir durum diyagramı görülmektedir. Nesne “Denetleniyor” durumuna geçtiğinde “ürün denetle” eylemini gerçekleştirir. Eylem tamamlandıktan sonra bir sonraki geçilecek durum “ürün mevcut” ve “ürün yok” koşullarına göre belirlenir. Eğer ürün mevcut değilse sipariş iptal edilir. Eğer ürün stoklarda mevcut ise ürün gönderilir. Nesne “Gönderiliyor” durumuna geçtiğinde ise “göndermeyi başlat” eylemi gerçekleşir ve nesne “Gönderildi” durumuna geçer.

[](http://3.bp.blogspot.com/_1EPXNbAy3L0/Spz31N-UwoI/AAAAAAAAAFk/ZocYq676YW0/s1600-h/state03.jpg)

Durum diyagramları aynı zamanda bir nesnenin üst durumunu da gösterebilir. Üst durum, pek çok geçişin belirli bir duruma gittiği zamanlarda kullanılır. Son duruma kadar olan ara geçişleri göstermek yerine tek bir üst durum içerisinde durumlar tanımlanıp, son duruma geçiş üst durum üzerinden gösterilebilir. Bu, durum diyagramını okumayı kolaylaştırır.

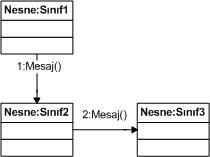
Aşağıdaki diyagramda bir üst durum görülmektedir. “Denetleniyor” ve “Gönderiliyor” durumları “İptal edildi” durumuna geçebileceğinden geçiş sadece bir üst durum olan “Aktif” durumundan “İptal edildi” durumuna gösterilmiştir. Bunun tam tersine “Gönderildi” durumuna sadece “Gönderiliyor” durumundan geçilebileceği için “Gönderildi” durumuna gelen ok sadece “Gönderiliyor” durumundan çıkarılır.

[](http://3.bp.blogspot.com/_1EPXNbAy3L0/Spz34pKu1DI/AAAAAAAAAFs/E4P45212qKs/s1600-h/state04.jpg)

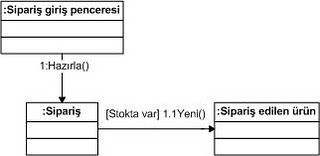
Gönderen Utku Ozan ÇANKAYA zaman: [13:31](http://umlnedir.blogspot.com/2009/09/durum-diyagram.html) [0 yorum](http://umlnedir.blogspot.com/2009/09/durum-diyagram.html#comments) [http://img2.blogblog.com/img/icon18_edit_allbkg.gif](http://www.blogger.com/post-edit.g?blogID=7928591152747902024&postID=5023505377635290692&from=pencil)

### [İşbirliği (Collaboration) diyagramı](http://umlnedir.blogspot.com/2009/09/isbirligi-collaboration-diyagram.html)

İşbirliği diyagramları nesneler arasındaki ilişkileri ve aralarında gezen mesajların sırasını gösterir. Nesneler kutular ile gösterilirken, oklar nesnelerin arasında gezen mesajları gösterir. Mesajların yanındaki numaralar “sıra (dizge) numarası” olarak adlandırılır ve nesneler arasında iletilen mesajların sırasını gösterir. UML 1,2,3.. gibi numaralandırma sistemlerinin yanında 1.1, 1.2, 1.3, 2.1, 2.2,... gibi numaralandırma sistemlerini de kabul edilmektedir.

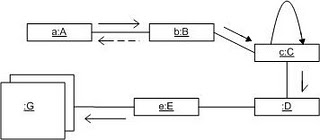
[](http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzEbAlS14I/AAAAAAAAADs/9SB8S88NxeQ/s1600-h/col01.jpg)

Aşağıdaki örnekte ise sipariş verme use case’i ile ilgili basit bir işbirliği diyagramını göstermektedir.. Bu sefer iki noktadan sonra sınıf isimleri yer almaktadır. Bunun nedeni ise bu sınıftan türetilen tüm nesnelerin aynı şekilde davranacağındandır.

[](http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzEgyUHMvI/AAAAAAAAAD0/LwD9cCs4spQ/s1600-h/col02.jpg)

İşbirliği diyagramlarında nesneler iki türde yer alır. Tedarikçi (supplier) ve istemci (client). Bu iki kavram işbirliği diyagramlarında parametreler gösterilirken karşımıza çıkar. Temel olarak tedarikçi nesneler çağrılan metodu sunan nesnelerdir. Bir dizge diyagramında aradaki bağlantıyı gösteren okun ucundaki nesne, mesajı alan ve çağrılan metodu sunan tedarikçi nesnedir. İstemci nesneler tedarikçi nesne üzerindeki metodları çağırır.

İşbirliği diyagramında kullanılan nesneleri daha detaylı incelersek:

[](http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzEmm_czsI/AAAAAAAAAD8/TnZS6eELCXA/s1600-h/col03.jpg)

Nesne: Yukarıdaki işbirliği diyagramında bize dizge diyagramlarından da tanıdık geleceği üzere nesne aynı şekilde gösterilmektedir. Nesne kutusunun içerisinde yer alan “a” harfi nesneyi, “A” harfi ise sınıfı temsil etmektedir.

[http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzEswb106I/AAAAAAAAAEE/-WR01L8gjE8/s320/col04.jpg](http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzEswb106I/AAAAAAAAAEE/-WR01L8gjE8/s1600-h/col04.jpg)

Senkron mesaj: Gönderildiği nesneden yanıt bekleyen mesajlardır ve belirli bir süre sonra, işlem tamamlandığında aynı bağlantı üzerinden yanıt alınmaktadır. Bir senkron mesaj

[http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzEzcrLTyI/AAAAAAAAAEM/2_O084nJjWw/s320/col05.jpg](http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzEzcrLTyI/AAAAAAAAAEM/2_O084nJjWw/s1600-h/col05.jpg)

ile gösterilirken geri dönen yanıt da

[http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzE3xfXkOI/AAAAAAAAAEU/E0jjVfJxBtU/s320/col06.jpg](http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzE3xfXkOI/AAAAAAAAAEU/E0jjVfJxBtU/s1600-h/col06.jpg)

işaretiyle gösterilir.

Asenkron mesaj: Senkron mesajların aksine cevap beklemeyen mesajlardır. Bir işbirliği diyagramında asenkron mesaj aşağıdaki simge ile gösterilir.

[http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzE8Q25JAI/AAAAAAAAAEc/ARVL64nv6bs/s320/col07.jpg](http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzE8Q25JAI/AAAAAAAAAEc/ARVL64nv6bs/s1600-h/col07.jpg)

Prosedür çağrısı: Yukarıdaki diyagramda “b” nesnesi “c” nesnesi üzerinden bir prosedürü çağırmaktadır.

[http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzFCCg00HI/AAAAAAAAAEk/M1OfWB-hsII/s320/col08.jpg](http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzFCCg00HI/AAAAAAAAAEk/M1OfWB-hsII/s1600-h/col08.jpg)

Sıra numarası: İşbirliği diyagramlarında zaman bilgisi bulunmadığından mesajlaşmalarda sıra numarası kullanılır. Dizge diyagramlarında zaten bir zaman ekseni mevcut olduğundan sıra bilgisini yazmak seçime bağlıdır. Fakat işbirliği diyagramlarında zamanı ekseni olmadığından mesaj numarası kullanmak zorunludur.

[http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzFGp8O5sI/AAAAAAAAAEs/KV8xcGNYIFI/s320/col09.jpg](http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzFGp8O5sI/AAAAAAAAAEs/KV8xcGNYIFI/s1600-h/col09.jpg)

Belirsiz nesne (Anonymous object): Bir sınıftan türeyen her nesnenin aynı şekilde davranacağının düşünüldüğü tasarımlarda sınıftan türeyen nesne için isim kullanılmayabilir. Bu durumda nesne adı belirtilmez ve belirsiz nesne “:A” şeklinde sadece sınıf adı ile nitelenir.

[http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzFNP4Gt0I/AAAAAAAAAE0/-E2VhH-28PY/s320/col10.jpg](http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzFNP4Gt0I/AAAAAAAAAE0/-E2VhH-28PY/s1600-h/col10.jpg)

Kendine referans (Self-reference): Kendi üzerindeki bir metodu çağıran nesnedir. Çoğu durumda nesne, kontrol amaçlı olarak sahip olduğu metodları kullanabilir.

[http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzFRkzsmmI/AAAAAAAAAE8/hwgppiVa_S0/s320/col11.jpg](http://1.bp.blogspot.com/_1EPXNbAy3L0/SpzFRkzsmmI/AAAAAAAAAE8/hwgppiVa_S0/s1600-h/col11.jpg)

Zaman aşımı (Timeout): Belirli bir zaman cevap gelmeyince çağrılan mesaj tipidir. Aşağıdaki gibi gösterilir.

[http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzFWoJqV1I/AAAAAAAAAFE/vCP92Q8SGqw/s320/col12.jpg](http://3.bp.blogspot.com/_1EPXNbAy3L0/SpzFWoJqV1I/AAAAAAAAAFE/vCP92Q8SGqw/s1600-h/col12.jpg)

Çoklu nesne (Multi object): Bir nesen kendi içerisinde birden fazla nesne içeriyorsa bu nesne çoklu nesne olarak tanımlanırve aşağıdaki gibi gösterilir.

[](http://2.bp.blogspot.com/_1EPXNbAy3L0/SpzFcOPzyaI/AAAAAAAAAFM/YPcm2QPUXs4/s1600-h/col13.jpg)

Yorum: Diyagramın kim tarafından, ne zaman oluşturulduğunu ve kapsamını tutan bileşendir. Bilgi amaçlı olarak sağ üst köşeye konulabilir.

Karmaşık sistemler, tasarlanan görevlerine yerine getirmek için ortaklaşa çalışan bir çok nesne içermektedir. UML içindeki işbirliği diyagramları temel olarak bu nesneler arasındaki ilişkiye yoğunlaşırken dizge diyagramları mesaj sırası ile ilgilenmektedir. Her iki diyagram da pek çok noktada ortaklık gösterir. Aralarındaki farklara bir göz atmek istersek:

1. Dizge diyagramları, zamana bağımlı olarak nesneler arasındaki mesajlaşmayı gösterir. İşbirliği diyagramları ise zamandan bağımsızdır.

2. İşbirliği diyagramları ile tek bir diyagram üzerinde dinamik yapıyı ve nesnelerin statik ilişkisini göstermek mümkündür. Dizge diyagramları ise sadece dinamik yapıyı temsil eder.

3. Dizge diyagramları ilk analiz safhasında oluşturulurken işbirliği diyagramları daha çok tasarım aşamasında oluşturulur.

4. Dizge diyagramlarından mesajlaşma sıraları, işbirliği diyagramlarında ise ilişkiler daha kolay gözlemlenir. Büyük diyagramlar için işbirliği diyagramlarını takip etmek zorken dizge diyagramlarında bu iş daha kolaydır. Ancak işbirliği diyagramları daha az alanda daha çok bilgiyi gösterebilir.

5. Belirli bir amacı yerine getirmek üzere bir araya gelen nesneler arasındaki ilişkiler işbirliği diyagramları ile daha rahat görülür. Yine genel resmi görmek işbirliği diyagramları ile daha kolay olmaktadır.

6. Dizge diyagramlarındaki bir bağlantı bir nesneden diğer bir nesneye giden mesajı gösterirken işbirliği diyagramlarındaki bir bağlantı nesneler arasındaki bir ya da daha fazla mesajı gösterebilir.

7. Dizge diyagramlarında nesnelerin yaratılışı ve sonlandırışı daha kolay görülür. İşbirliği diyagramında ise aynı iş bu kadar kolay gerçekleştirilemez.

8. İşbirliği diyagramlarında zaman kavramı olmadığı için bir nesnenin ne zaman aktif ne zaman pasif olduğunu anlamak zordur. Bunu anlamak içn mesaj tiplerini incelemek gerekmektedir. Fakat dizge diyagramlarında bunu kavramak çok kolaydır.

9. İşbirliği diyagramları daha karışıuk dallanmalara (branching) ve eş zamanlı akış denetimine (concurrent flow of control) izin verirken dizge diyagramları doğası gereği çok daha basit dallanmalara müsaade etmektedir. Bu yönüyle işbirliği diyagramları daha karışık sistemleri modelleyebilmektedir.