YMT 312-Yazılım Tasarım Ve Mimarisi

Nesneye Yönelik Çözümleme ve Tasarım

Bölüm-6

Fırat Üniversitesi Yazılım Mühendisliği Bölümü

Bu Haftaki Konular

- Nesneye Yönelik Kavramlar.....5
- Nesne Yönelimli Sistemleri Destekleyen Kavramlar.....10
- Nesneye Yönelik Çözümlemenin Temelleri.....15
- Gereksinim Belirleme Çalışmaları.....18
- Nesneye Yönelik Çözümleme Teknikleri.....24
- Alt Sistem Modellemesi.....28
- Nesneye Yönelik Tasarım.....39
- Nesneye Yönelik Tasarım Metodolojileri.....47
- NY Metodolojilerinin Tasarım Yöntemleri.....49
- Genel Sistem Tasarımı.....57

Amaçlar

Nesneye Yönelik Kavramlar Hakkında Bilgilenmek

Kalıtım ve Çoklu Kalıtım Kavramlarını Öğrenmek

Nesne Yönelimli Sistemleri Destekleyen Kavramları Bilmek

Gereksinim Belirleme Çalışmalarını Öğrenmek

Nesneye Yönelik Çözümleme Tekniklerini Öğrenmek

Alt Sistem Modellemesini Kavramak

Nesneye Yönelik Tasarım Katmanlarını Öğrenmek

Nesneye Yönelik Metodolojilerinin Tasarım Yöntemlerini Öğrenmek

Genel Sistem Tasarımı Yapmak

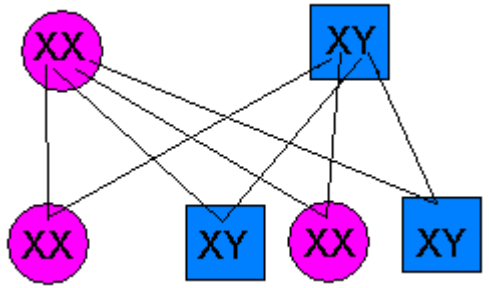
Giriş

- Yapısal tekniklerle güçlenen yazılım mühendisliğine, programlama düzeyinden yukarıya doğru bir gelişme ile Nesneye Yönelik (NY) yaklaşımlar kazandırıldı.
- NY programlama dillerinin gördüğü büyük ilgi, çok geçmeden bu yaklaşımın tasarım ve çözümleme konularına da yansıması ve daha sonra tüm süreç modeli ve metodoloji uyarlamaları ile izlendi.
- Bugün yeni projeler için geleneksel/NY tartışması, NY lehinde kararlarla sonuçlanma yönünde değişmektedir.
- Geçerli neden olmadığı durumlarda artık NY ortamlarda geliştirme çalışması yapmak gerekmektedir.
- Bu bölümde önce NY kavramlar kısaca özetlenecek ve sonra çözümleme teknikleri sunulacaktır.
- Geleneksel yaklaşımların geldiği noktaya, NY yaklaşımlar çabucak gelerek, kabul görmüş metodolojiler yaygınca bilinen standartlar halinde gelişmiştir.
- Hatta bu yaklaşımlardaki benzerlikler ve piyasanın da motivasyonu sonucunda değişik yaklaşımlar arasında uyum çalışmaları yapılmıştır.
- Fusion ve UML, bu birleştirme/standartlaşma çabalarının başlıca iki örneğini oluştururlar.

Nesneye Yönelik Kavramlar

Doğal çevremizdeki nesnelerin özellikleri ve ilgili oldukları süreçler bir arada düşünülür. Bir nesnenin özellikleri olduğu gibi onun yapacağı işlemler de vardır. NY modellerin ilk özelliği olarak, veri ve süreçlerin bir nesne için bir arada değerlendirilmesi tanımlanır. NY kavramların temeli olarak, aşağıdaki dört özelliği sıralayabiliriz:

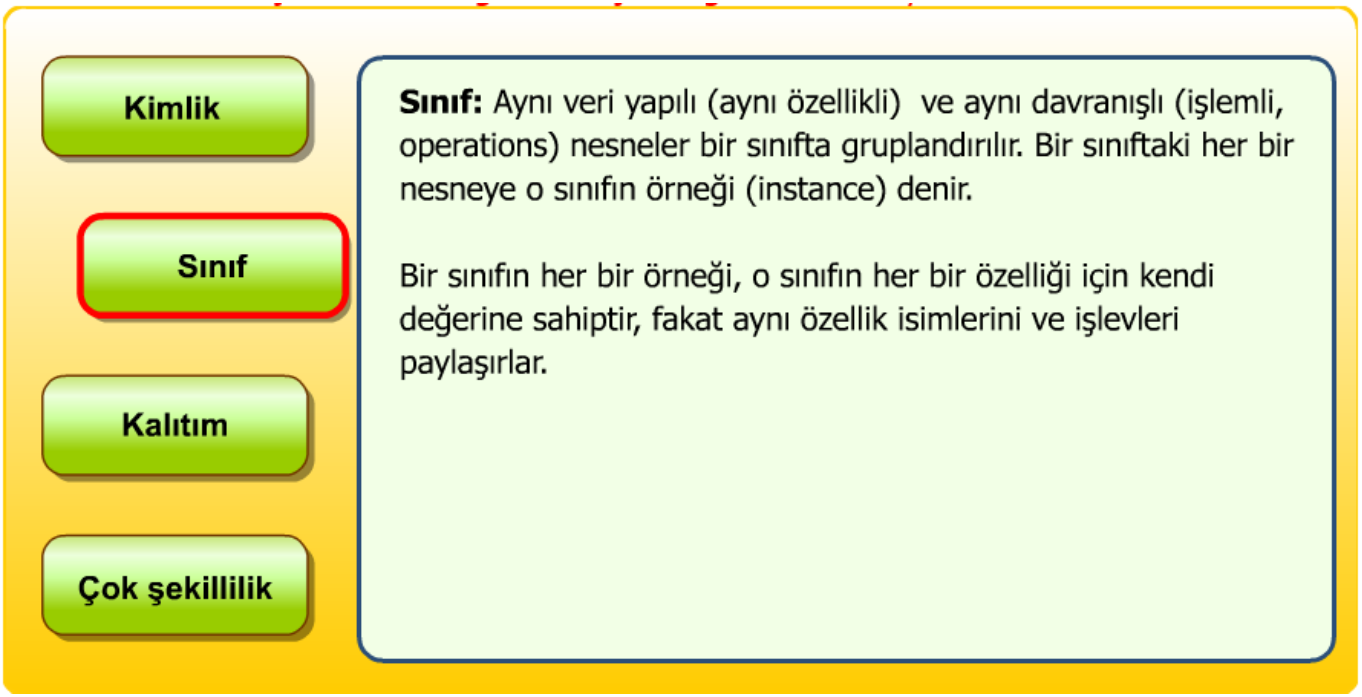
1. Kimlik (Identity)
2. Sınıflama (Classification)
3. Kalıtım (inheritance)
4. Çok Şekillilik/Biçimlilik(Polimorphism)



Kimlik


Kimlik
Sınıf
Kalıtım
Çok şekillilik
Kimlik: Nesne denilen, farklı, ayırt edilebilen varlıkların tanımlanmasıdır. Her nesnenin kendi kimliği vardır. Örneğin; Ali'nin bisikleti, renkli TV.
Poligon Nesneleri
Poligon Sınıfı
Özellikler (Attributes)
İşlemler (Operations)
üçgen
dikdörtgen
yamuk
çerçeve rengi
dolgu rengi
kenar sayısı
çiz
sil
taşı

Sınıf


Kimlik
Sınıf
Kalıtım
Çok şekillilik
Sınıf: Aynı veri yapılı (aynı özellikli) ve aynı davranışlı (işlemli, operations) nesneler bir sınıfta gruplandırılır. Bir sınıftaki her bir nesneye o sınıfın örneği (instance) denir.
Bir sınıfın her bir örneği, o sınıfın her bir özelliği için kendi değerine sahiptir, fakat aynı özellik isimlerini ve işlevleri paylaşırlar.

Kalıtım

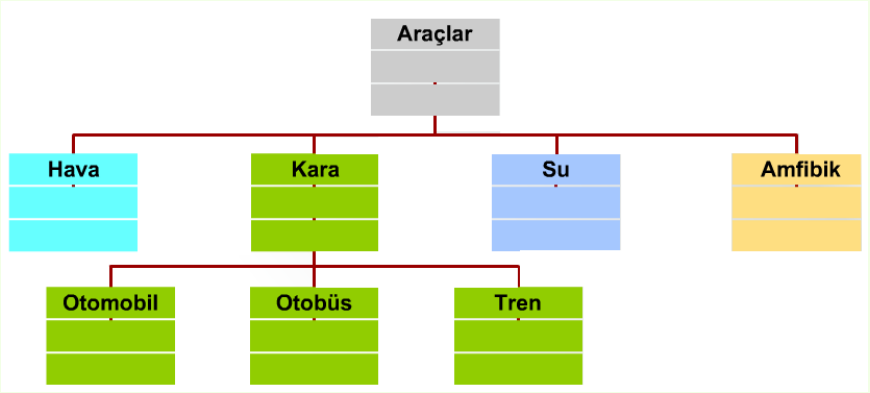
Kimlik

Sınıf

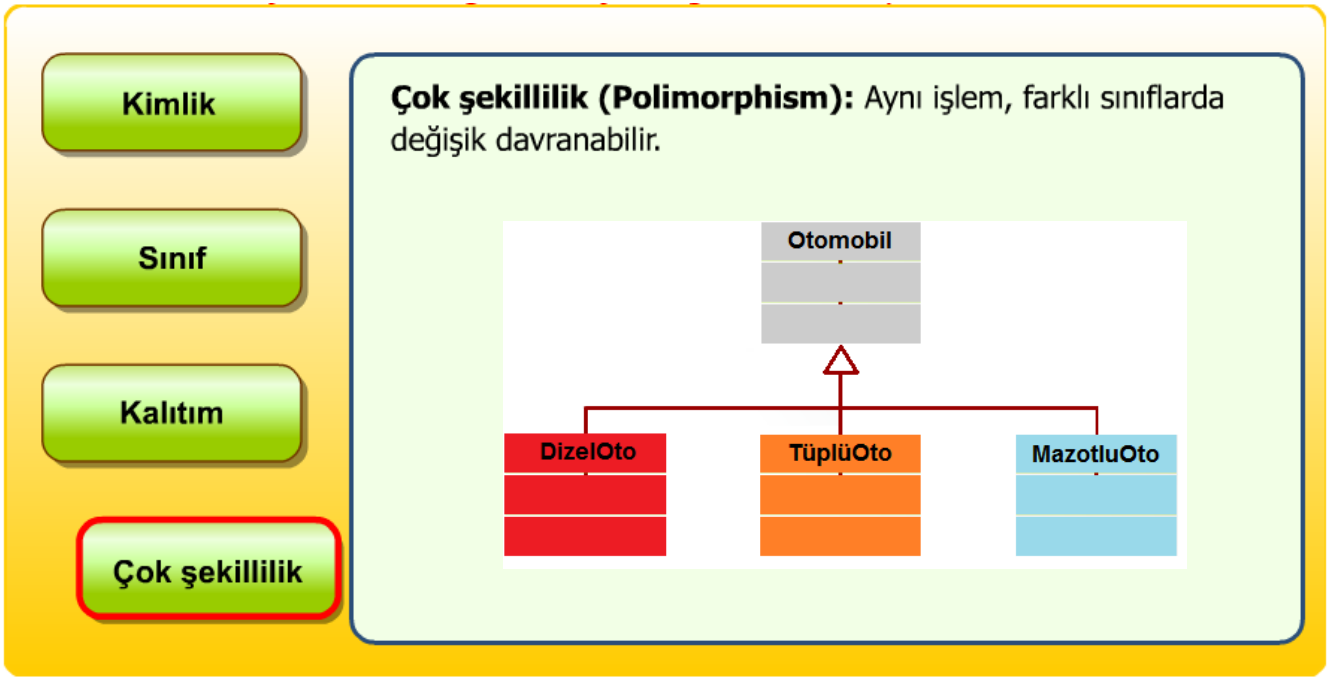
Kalıtım

Çok şekillilik

Kalıtım (Inheritance): Hiyerarşik ilişkiye dayanan sınıflar arasında özellik ve işlevlerin paylaşılmasıdır.



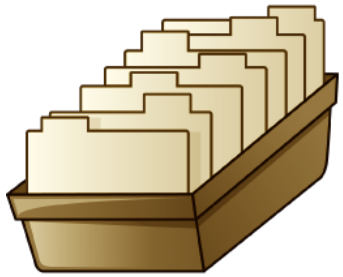
Çok Şekillilik

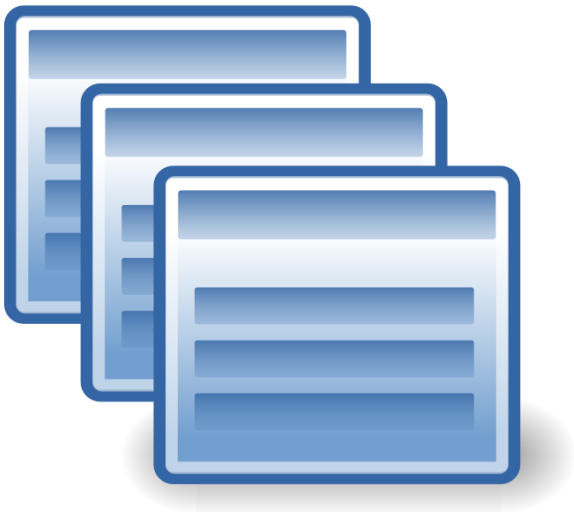


Nesne Yönelimli Sistemleri Destekleyen Kavramlar

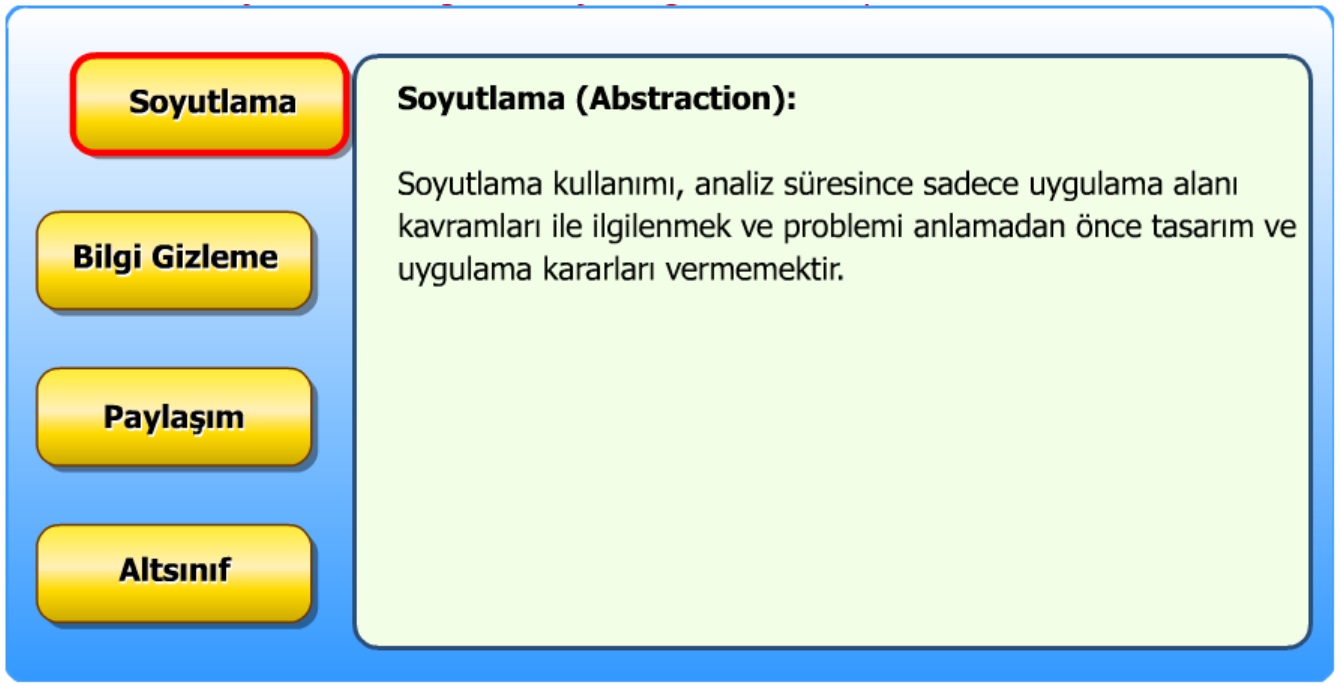
Sadece nesne yönelimli sistemlerde olmayan ama nesne yönelimli sistemleri destekleyen kavramlar;

- Soyutlama (Abstraction),
- Bilgi Gizleme (Encapsulation),
- Paylaşım (Sharing),
- Altsınıf (Subclass) ve "Aggregation" dır.

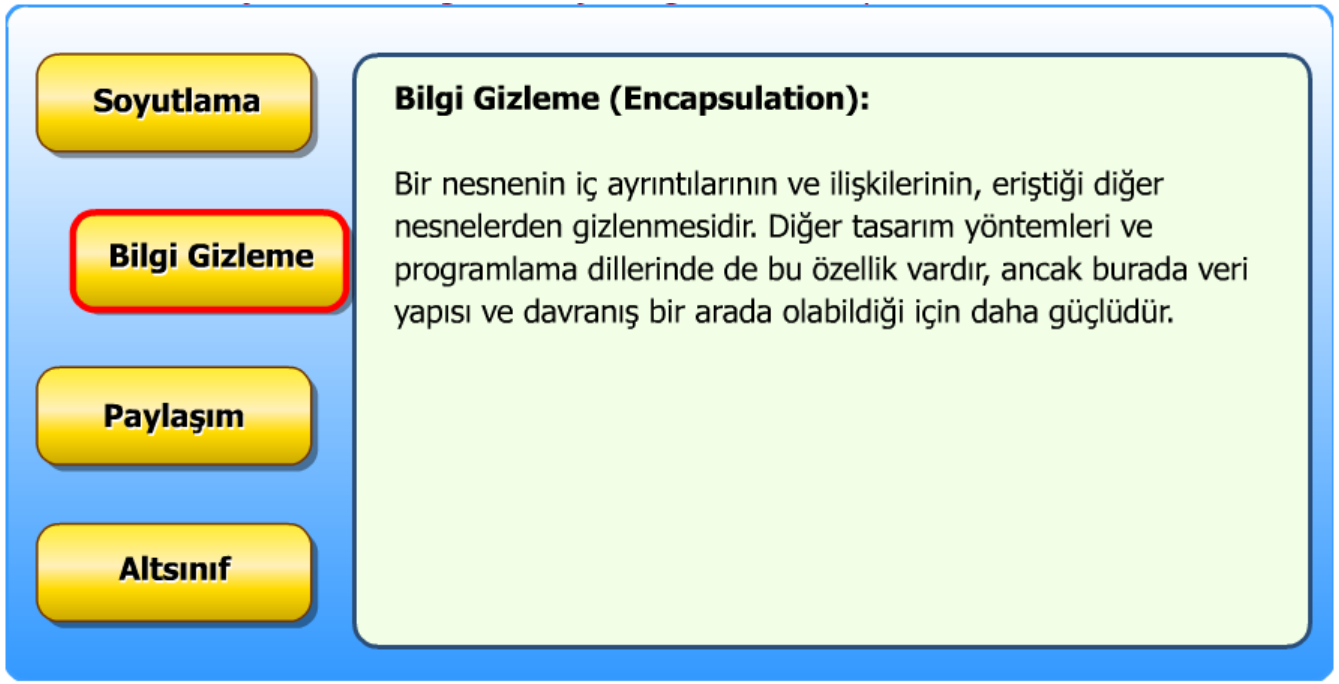




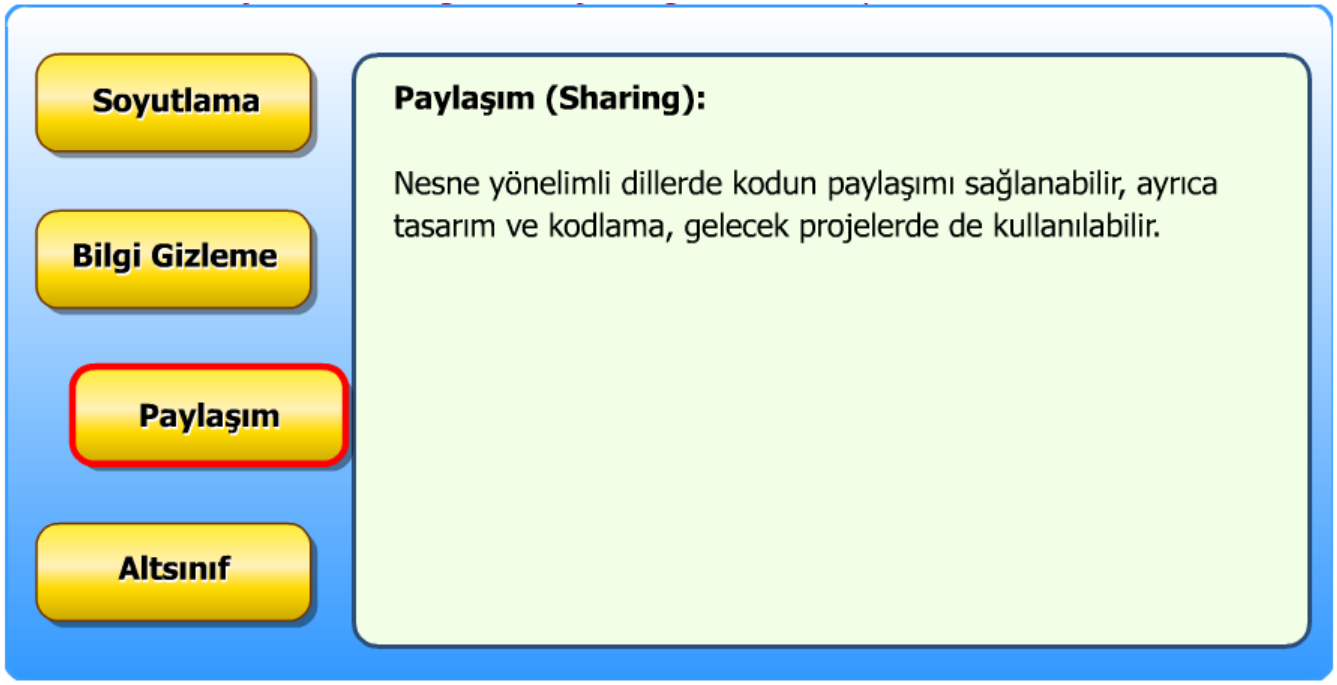
Soyutlama


Soyutlama
Bilgi Gizleme
Paylaşım
Altsınıf
Soyutlama (Abstraction):
Soyutlama kullanımı, analiz süresince sadece uygulama alanı kavramları ile ilgilenmek ve problemi anlamadan önce tasarım ve uygulama kararları vermemektir.

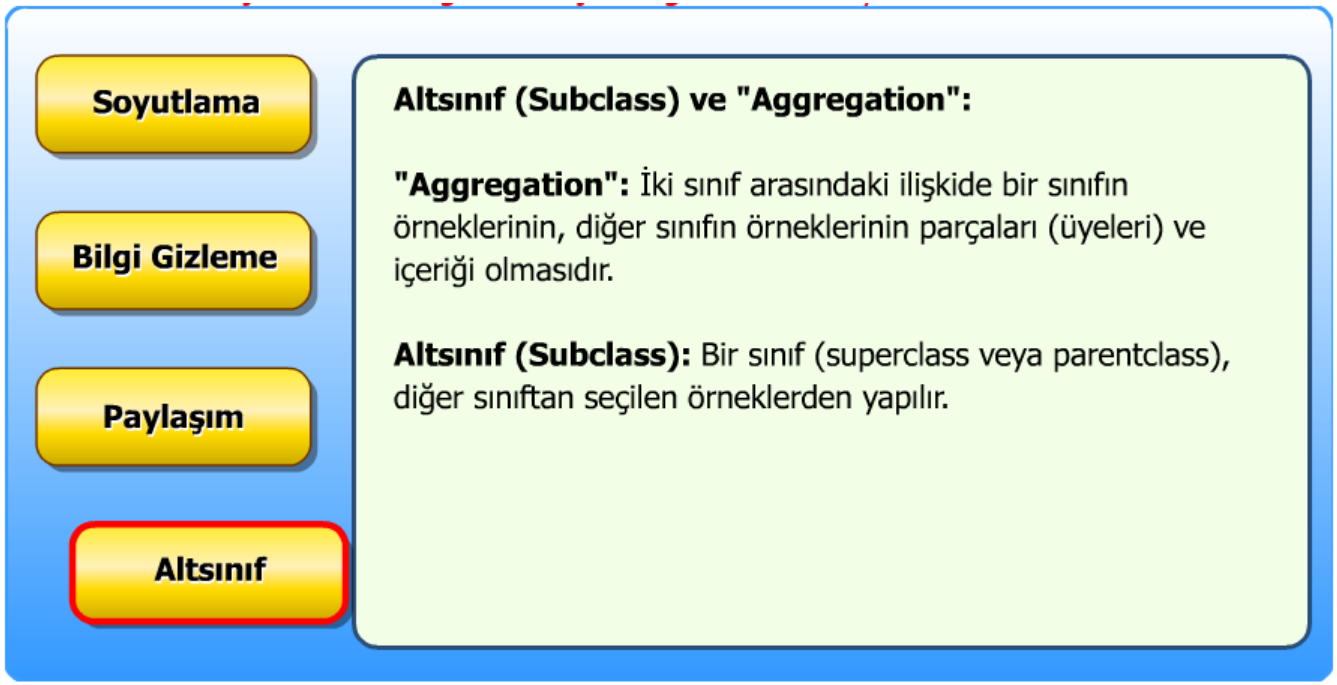
Bilgi Gizleme



Paylaşım


Soyutlama
Bilgi Gizleme
Paylaşım
Altsınıf
Paylaşım (Sharing):
Nesne yönelimli dillerde kodun paylaşımı sağlanabilir, ayrıca tasarım ve kodlama, gelecek projelerde de kullanılabilir.

Altsınıf


Soyutlama
Bilgi Gizleme
Paylaşım
Altsınıf
Altsınıf (Subclass) ve "Aggregation":
"Aggregation": İki sınıf arasındaki ilişkide bir sınıfın örneklerinin, diğer sınıfın örneklerinin parçaları (üyeleri) ve içeriği olmasıdır.
Altsınıf (Subclass): Bir sınıf (superclass veya parentclass), diğer sınıftan seçilen örneklerden yapılır.

Nesneye Yönelik Çözümlemenin Temelleri

Çözümleme: Bir şeyi anlayabilmek için parçalarına ayırmak.

Sistemi anlamaya yönelik çalışmalardan ve üst düzey planlama eylemlerinden oluşur.

- Uygulama/problem alanının anlaşılması.
- Kullanıcı gereksinimlerinin anlaşılması.
- Koddaki sınıflar ve nesneler ile bunların arasındaki üst düzey etkileşimlerin belirlenmesi: Çözümleme modelinin oluşturulması.

“Bir sorunu anlamadan çözemezsiniz.”

Uygulama Alanının Çözümlenmesi (Domain Analysis)

- Amaç, uygulama alanını anlamak ve elde edilen bilgileri analiz modeline taşımaktır.
  - Uygulama alanı hakkında bilgi edinilebilecek kaynaklar:
  - Teknik literatür
  - Mevcut uygulamalar
  - Müşteri anketleri
  - Uzman tavsiyeleri
  - Mevcut ve gelecekteki gereksinimler
- Problem alanı hakkında bilgi edinmeden “müşterinin dilinden konuşamazsınız”.

Genel Olarak NY Metodolojiler

- Sözü geçen değişik modeller, temelde genellenebilecek ortaklıklar göstermektedir.
- Bu özellikten faydalanmak için önce Fusion ve sonra da UML metodolojileri ortaya çıkmıştır.
- İki girişimde de önceki yöntemlerin iyi taraflarının alınarak, standart ve daha iyi bir metodolojiye ulaşılmak isteği yer almıştır.
- Genellenmiş şekilde işlemleri aşağıdaki gibi özetleyebiliriz:
  1. Gereksinimler belirlenir (kullanım durumları ve senaryolar kullanılır).
  2. Sınıf ve nesneler belirlenir.
  3. Özellikler ve işlemler tanımlanır.
  4. Sınıfları ve nesneleri organize edecek hiyerarşiler ve yapılar belirlenir.
  5. Nesneler arası ilişkiler modellenir.
  6. Nesneler arası etkileşim modellenir.
  7. Ortaya çıkan modeller, kullanım örnekleri ve senaryolar ile denenir ve iyileştirilir.

Gereksinim Belirleme Çalışmaları

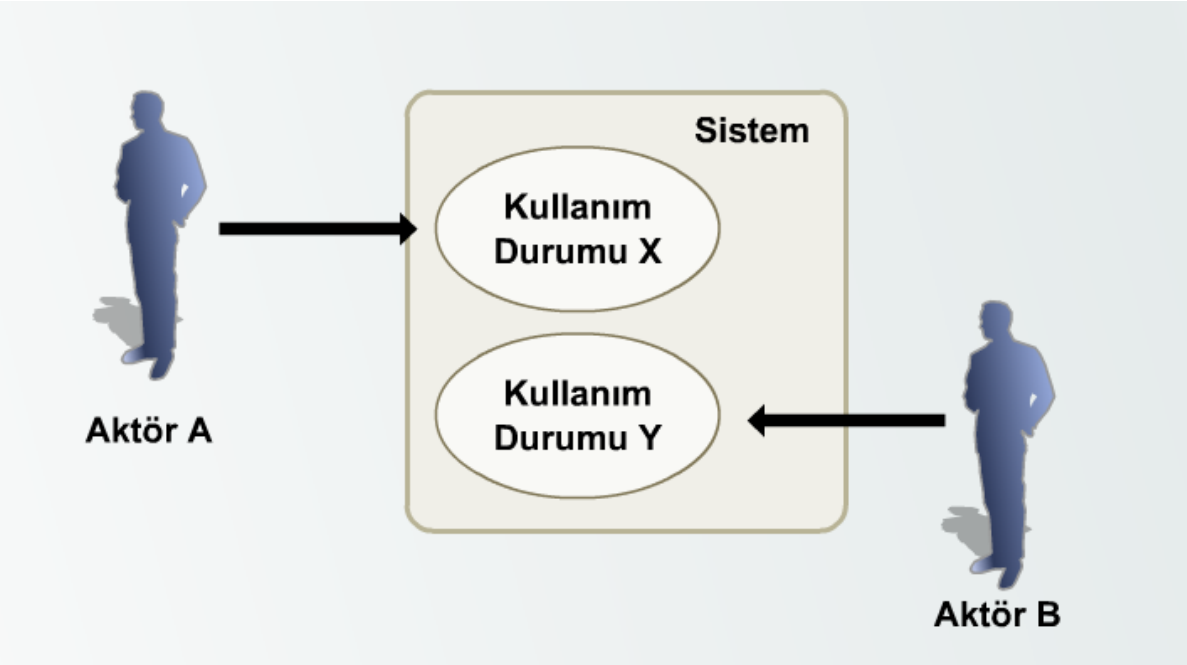
- Bu bölümde sözü geçecek olan iki teknik;
  - Kullanım Durumları ve
  - Sınıf Sorumluluk İşbirlikçi modelleridir.

- Kullanım Durumları, NY olmayan ortamlara da uyarlanabilir.
- Ancak bu tekniğin ortaya çıkışı NY metodolojilerle birlikte olmuştur.

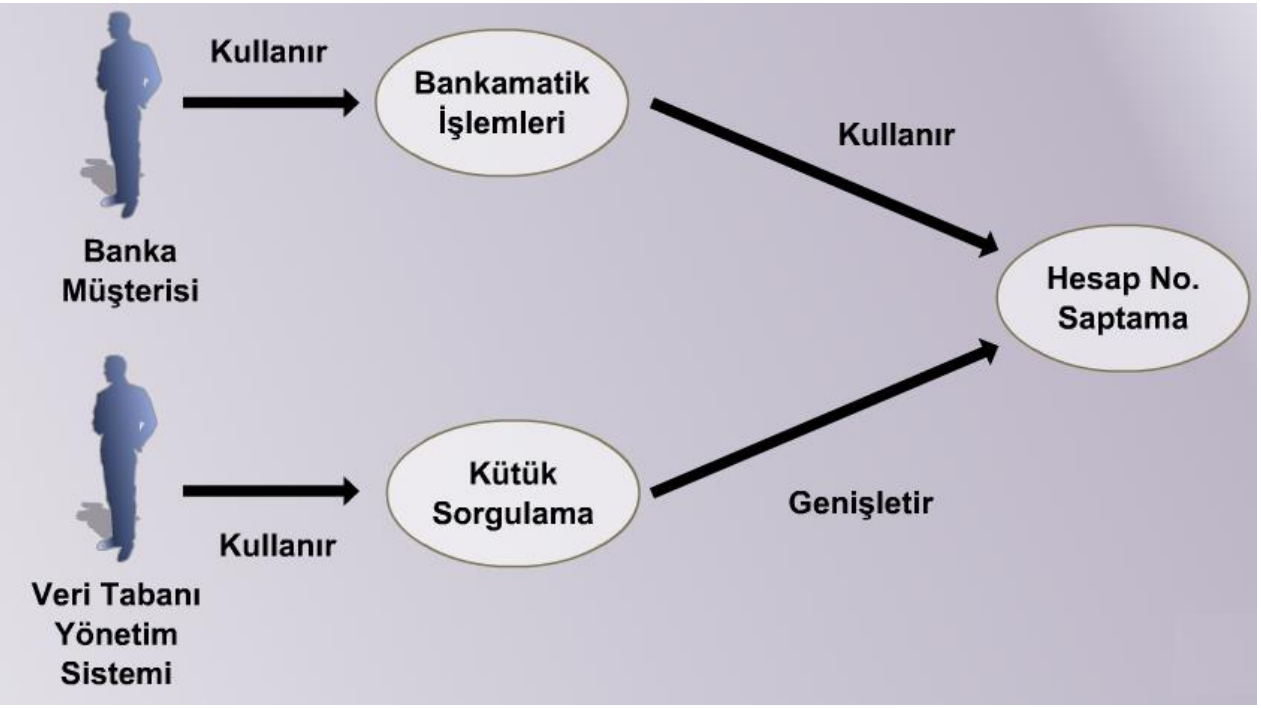
Kullanım Durumları

- İlk gereksinim derleme çalışmalarından sonra, sistemin kullanımıyla ilgili senaryoları denemeye yönelik modeller çizilir. Bunlarda 'aktörler' sistemin işlemesinde temel bazı süreçlerin harekete geçirilmesini sağlarlar.
- Aktörler, insan veya sistemin bir öğesi olabilir (işletim sistemi, bir dış etken vb.).
- Yalnızca bir süreci başlatmak için değil, sistem ile bir işlem boyunca herhangi bir etkileşimde bulunmak üzere de yer alabilirler.

Kullanım Durumu Diyagramı



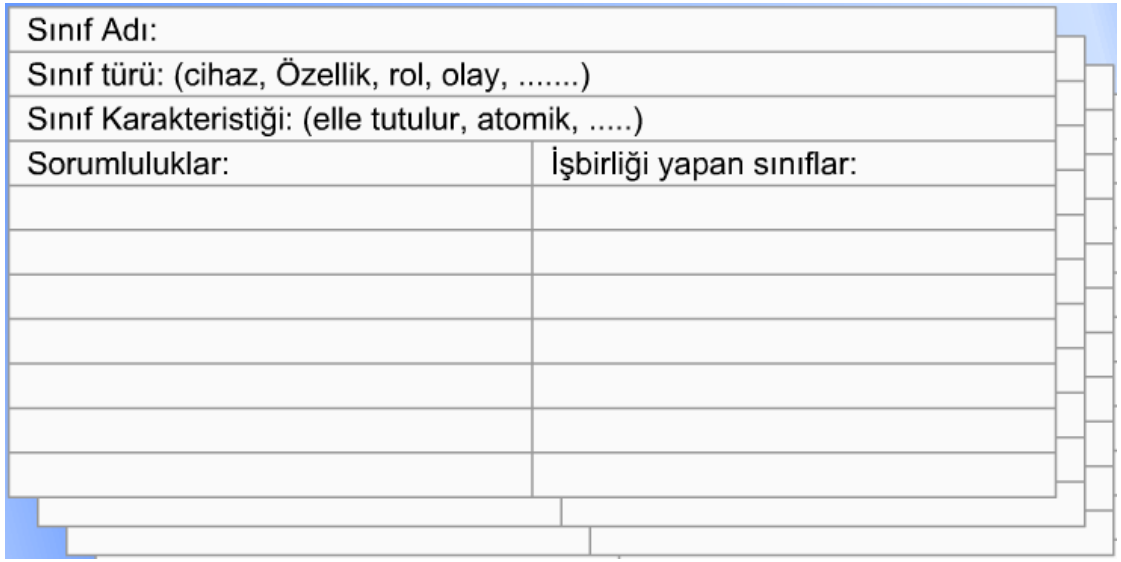
Bankamatik Sistemi Örneğinde Kullanım Diyagramı Modellemesi



Sınıf/Sorumluluk/İşbirlikçi Modeli

- Kullanım senaryoları irdelendikten sonra sistemi oluşturan temel sınıflar, bunların sorumlulukları ve etkileşimleri ortaya çıkarılmalıdır.
- Sınıf/Sorumluluk/İşbirlikçi (SSİ) modeli, sınıfları temsil eden kartlar kümesidir.
- Bu kartların başına sınıfın adı yazılır.
- Altta ise sınıfın sorumlulukları, solda ve sağda da işbirliği yapacak diğer sınıfların adları yazılır. Kartlar sanal olarak da tutulabilir.
- Sınıflar, dış varlıklar, doğal nesneler, olaylar, roller, kurum yapı birimleri, yerler ve yapılar gibi değişik kavramları temsil edebilirler.
- Doğal dil ile yapılmış sistem tanımından ilk önce dilbilimce isim olarak nitelendirilebilecek olanlar sınıf olmaya adaydırlar.
- Genelde bir seçimin sınıf olması için altı özelliği bulundurması istenir.

Sınıf/Sorumluluk/İşbirlikçi Modeli Örnek



Sınıf Adı:	
Sınıf türü: (cihaz, Özellik, rol, olay, ...)	
Sınıf Karakteristiği: (elle tutulur, atomik, ...)	
Sorumluluklar:	İşbirliği yapan sınıflar:

Bu özellikler;

- Bilgi saklama gereği,
- İşlem beklentisi,
- Birden çok özellik,
- Ortak özellikler,
- Ortak işlemler ve
- Temel ihtiyaç olma

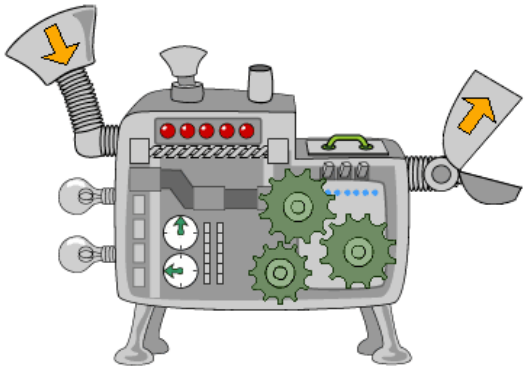
biçimindedir.

Nesneye Yönelik Çözümleme Teknikleri

- Önceki sayfalarda söz edilen teknikler, bu bölümde biraz daha ayrıntılı bir şekilde incelenmektedir.

- Ancak örnek olarak verilecek diyagramların değişik metodolojilerde farklılıklar gösteren çizimlerinin olacağı hatırlanmalıdır.

- Bu ayrılıklar konunun anlatımı açısından çok da önemli değildir.



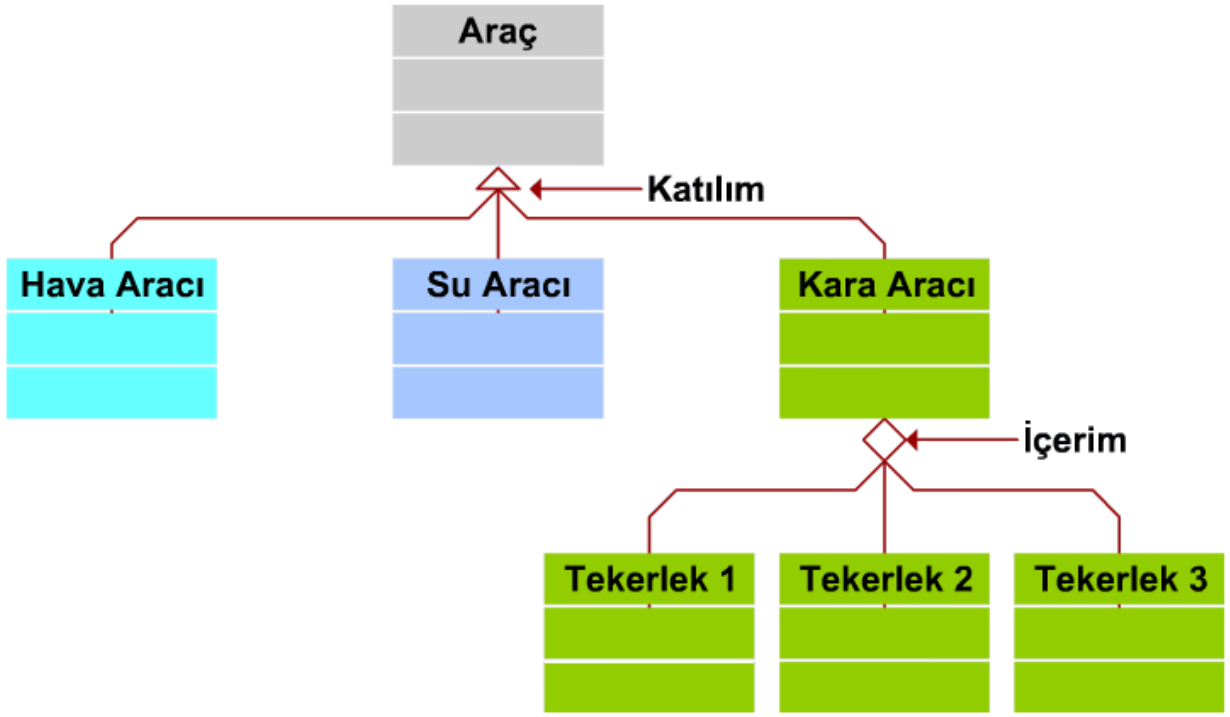
Yapısal İlişkiler

- Gereksinim çalışması sırasında sistemi tanımlamada kullanılabilecek bir çok yazılı belge oluşacaktır.
- Bu belgelerde, nesne olmaya aday olan varlıklar ve sınıf olmaya aday olan varlıklar önce belirlenir ve bunların arasında yapısal ve ilişkisel bağlantılar kurularak modelleme işlemi başlar.
- Yalnızca sınıflar ya da yalnızca nesneler ile başlamak zorunlu değildir.
- Yapısal ilişkiler, başlıca kalıtım ve içerim kavramlarına dayalıdır.
- Sistemde bulunacak nesne sınıflarında genelleştirme ve özelleştirme ilişkileri aranarak bunlar modele kalıtım bağlantısı olarak yansıtılır.

Yapısal İlişkiler Örnek

- Araçlar daha genel olup, su veya hava ortamlarında da yol alabilirler, yük taşıma amacıyla kullanılabilirler.
- Bir aracın taşıma kapasitesi, boş ağırlığı ve boyutları, onun durağan özellikleri olarak modellenebilir. İşlemler olarak da yükleme ve gitme gibi örnekler verebiliriz.
- Sözü geçen özellik ve işlemler ile sarmalanmış olan araç sınıfının bir özelleştirmesi olarak otomobil sınıfını tanımlayabiliriz.
- Otomobil sınıfı, sözü geçen özellik ve işlemleri kalıtım yolu ile edindiğinden kendisininmişçesine içerir. Ayrıca otomobil sınıfına özel eklemeler yapılabilir.
- Örneğin dört tekerlek gibi. Bu dört tekerlek, basit veri yapısı olarak tanımlanabileceği gibi, birer nesne olarak da modellenebilir.
- Otomobil sınıfının özelliği olarak yer alan tekerlek nesneleri, aslında 'içerim' yolu ile yapısal bir bağ kurmuşlardır.
- Bir otomobil, araç sınıfına bağlı olduğu gibi tekerlek sınıfına da bağlı olabilir (programcılar bu kolaylığa kaçıcı yolu bazen yeğlerler), ancak bu, modelleme açısından doğru bir yaklaşım olmaz.
- Çünkü kalıtım, bir genelleme/özelleşme ilişkisidir.
- Otomobiller araçtırlar, ama tekerlek değildirler. Bir sonraki slaytta Kalıtım ve İçerim bağlarını birlikte gösteren bir örnek içermektedir.

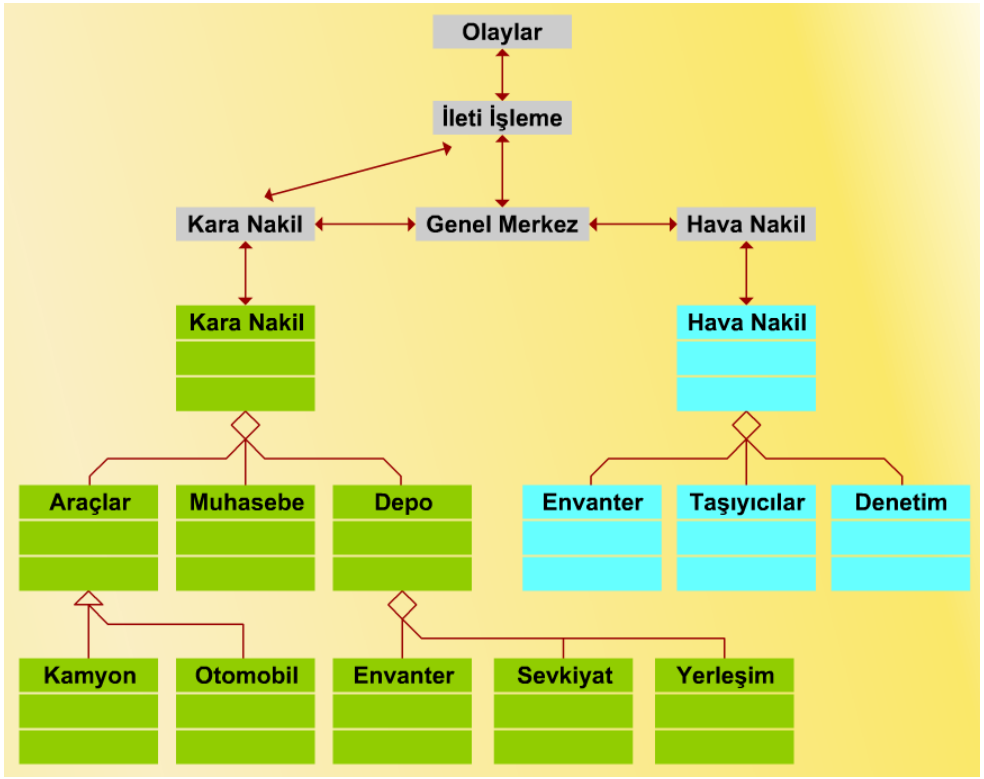
Kalıtım ve İçerim Örnek



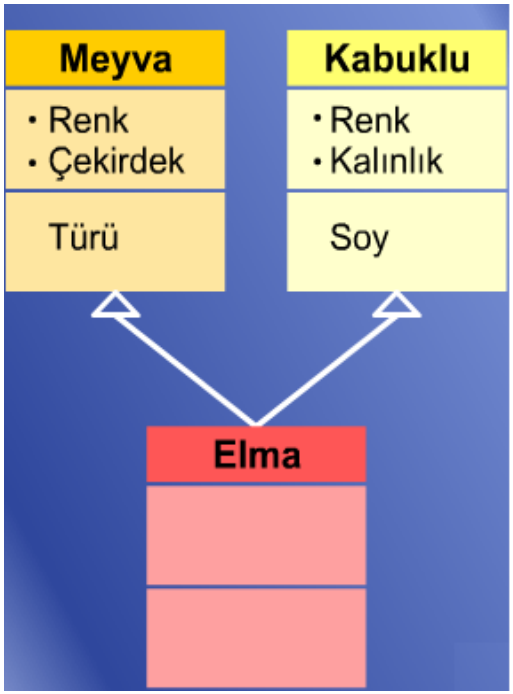
Alt Sistem Modellemesi

- Çözümleme aşamasında modelin mantıksal olarak anlaşılabilirliği için bu anlamsal bağların doğru taşınmasında yarar vardır.
- Tasarım aşamasında ise, verimlilik gibi bazı etkenlerin çok önemli olmaları durumunda bazı değişiklikler sonradan yapılabilir.
- Sözü geçen yapısal ilişkiler, sistem modelinin sıra düzensel olarak ayrıştırılması kavramına karşı düşerler. Bu önemli kavramı destekleyen bir üçüncü araç ise 'alt sistem' öğeleridir.
- Yapışıklık prensibinden hareketle bir işlem grubunun çalıştırılmasında emeği geçecek nesnelerin bir araya toplanarak yapay bir büyük nesne gibi değerlendirilmeleri, bir alt sistemin tanımlanmasıdır.
- Bazı yöntemlerde alt sistemlere 'özne' veya 'öğe' gibi isimler verilmiştir. NY tekniklerde alt sistem anlayışı ve içerim kavramının da kullanımının yanı sıra en ağırlıklı olarak kalıtım kavramı ile çözümleme modelleri oluşturur.

Alt Sistem Modellemesi Örnek



Çoklu Kalıtım

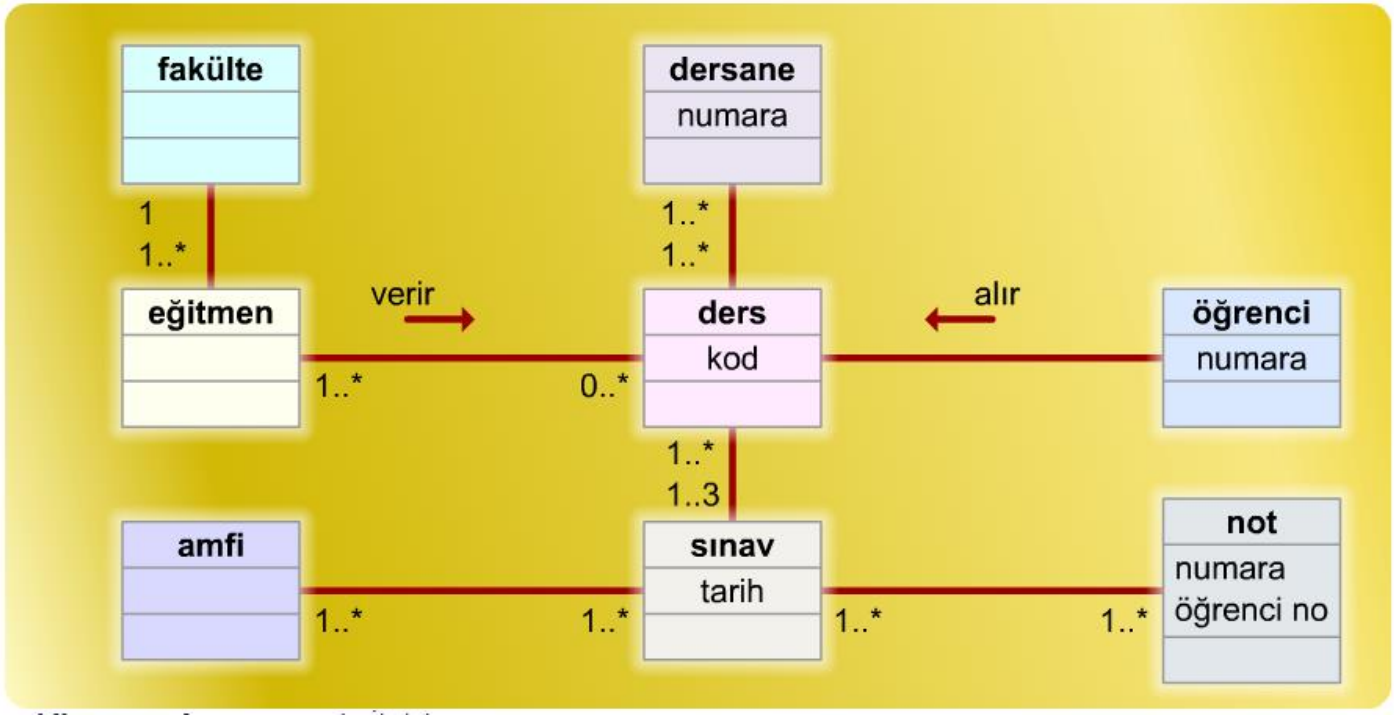


- Bazı durumlarda birden çok sınıfın özelliklerini kalıtım yolu ile almak uygun olabilir.
- Kalıtımı alan sınıfa, özelliklendirilmiş sınıf, kalıtım alınanlara da genel sınıflar denebilir.
- Bu durumda özelliklendirilmiş sınıf, kavramsal olarak her iki genel sınıfın içinde de bulunabilmelidir (bir elmanın hem meyva, hem de kabuklu cisim sınıflarına girebileceği gibi.)
- Genelde kalıtım için geçerli olan kural, bu durumda iki kere uygulanabilir olmalıdır:
- "Özelliklendirilmiş sınıf, genel bir sınıfın üyesidir." ifadesi geçerli ise kalıtım anlamlıdır.
- Bu ifadeyi elma örneğine uygularsak, "Elma bir çeşit meyvedir." ve "Elma bir çeşit kabuklu cisimdir." ifadeleri anlamsal olarak geçerli ifadelerdir.
- Yan tarafta çoklu kalıtım örneği modellenmektedir.

Çoklu Kalıtım İlişkilerin Gösterimi

- Geleneksel çözümleme tekniği olarak karşımıza çıkmış olan Nesne İlişki Diyagramları, NY teknolojisine uyarlanarak az değişiklikle kendisini kabul ettirmiştir.
- Yine nesneler arası ilişkiler isimleri konarak ve çoğullama belirtimi kullanılarak çizilmektedir.
- Genelde ikili ilişkiler kullanılmakla birlikte üçlü (üç nesne arasında bir ilişki) ve daha çoklu olanlarını da tanımlamak mümkündür. NY tekniklerinde ilişkiler yalnızca bir isim olmanın yanında, ayrı bir nesne olarak da tanımlanabilir. İlişkilerin iç özellik ve işlemleri olabilir.
- Ayrıca dönüşlü ilişkiler de tanımlanabilir.
  - Örneğin 'personel' sınıfı olarak modellenen sınıf içerisinde bir amir/memur ilişkisi tanımlanabilir ve bu, diyagramda aynı sınıftan çıkıp aynı sınıfta sonlanan bir oklu çizgi üzerine 'memur' ilişki tanımı yazılarak yapılabilir.

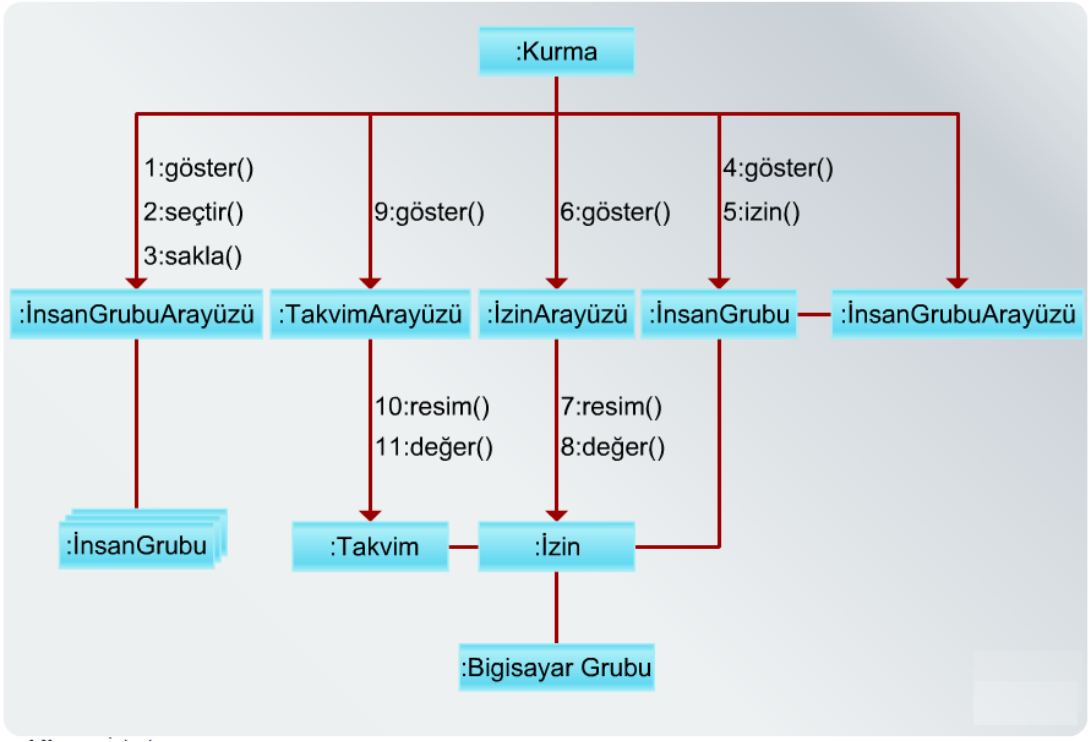
Sınıf Diyagramında İlişkiler



İşbirliği Diyagramları

- Kullanım durumu diyagramları ile irdelenen sistem düzeyindeki işlemlerin gerçekleştirilmeleri için birlikte çalışması gereken nesnelerin bu işbirliklerini yansıtır.
- İlk tekniklerdeki genelde sınıf ve nesne diyagramı ile yola çıkılma anlayışında nesne diyagramları, ileti trafiğini yansıtmakta idi.
- Bu nesne diyagramlarına karşı düşen yeni tekniklerde ise işbirliği diyagramı kullanılmaktadır.
- Biraz daha organize olarak hangi sistem işleminin yansıtıldığı sınırları ile belirtilerek ve kullanım örneği, SSİ gibi tekniklerle modellenmiş gereksinimlere dayandırılarak çizilir.
- Modeller arsındaki iletilerin sıralandırması ve biçemi belirtilerek sistem işleminin gerçekleştirilmesi modellenir.

İşbirliği Diyagramı



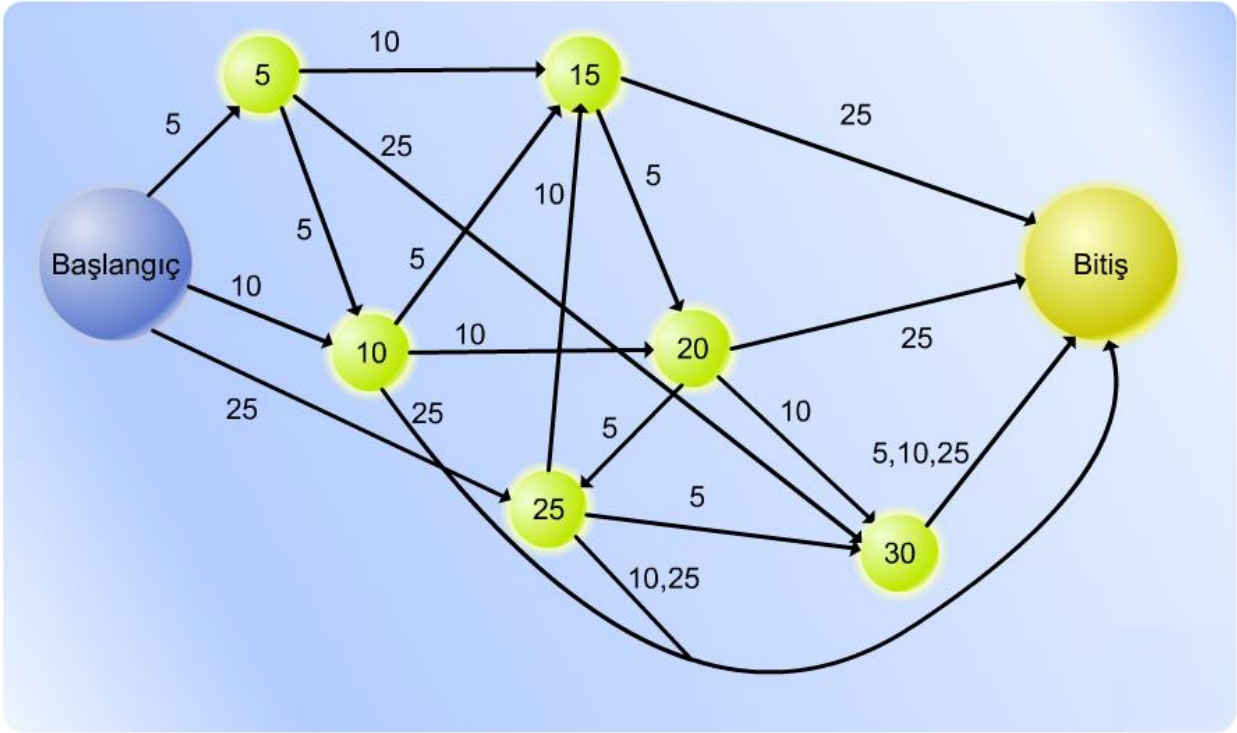
Davranış Modellemesi

- Çözümleme sırasında ayrıntıya girildikçe tasarımı ilgilendiren düşünceler söz konusu olur.
- Sistem davranışı, işbirliği diyagramları ile üst düzeyde ortaya konmuştur.
- Ancak bu işbirliği diyagramlarının bazı karmaşık kontrol gerektiren durumlarda ileti sıralandırması gibi kararları neye dayanarak yaptığının belirtilmesi istenebilir.
- Nesneler, daha önceki işlemlerin sonucu olarak bir duruma gelirler ve davranışları bu durumlara bağlı olarak ortaya konulur.
- Bu noktada 'durum modellemesi' konusuna girilmesi gerekir.
- Bazı metodolojiler, durum modellemesi konusunun çok ayrıntılı olduğunu ve daha çok tasarıma bırakılacak bir uğraşı olduğunu savunurlar.
- Durum sayısı artınca model karmaşıklaşır ve anlaşılması zorlaşır.

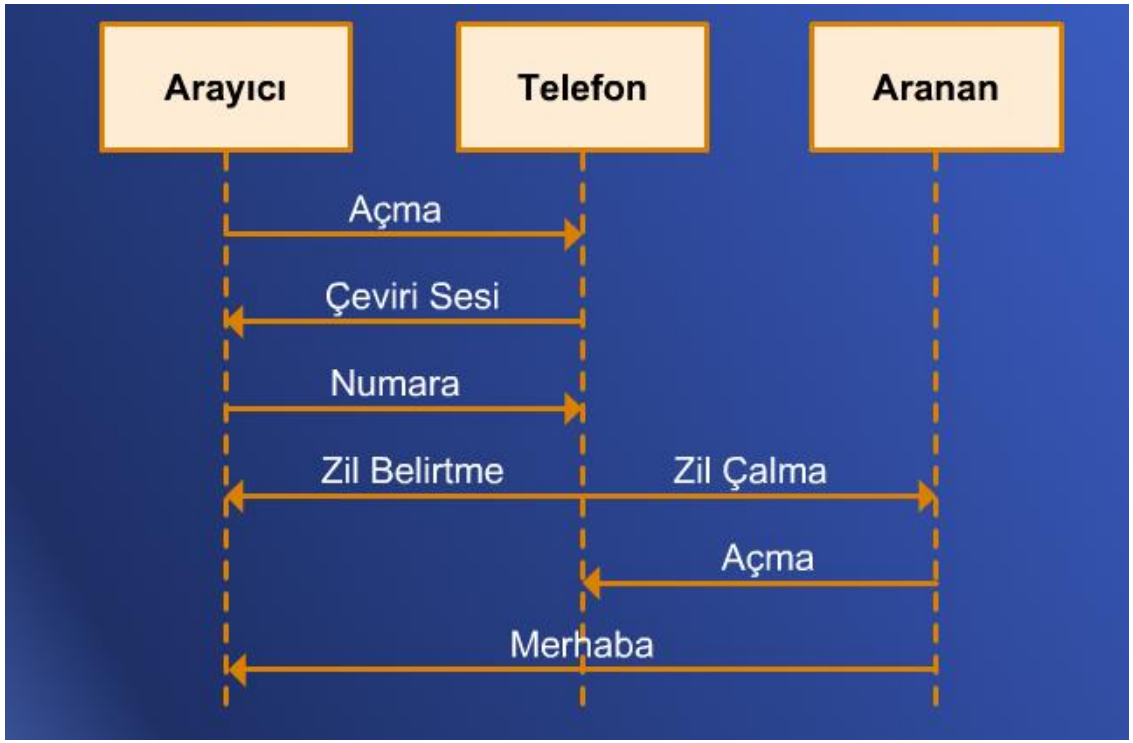
Davranış Modellemesi

- 'Durum Makinesi' modelleri, sistemin tümü için çok karmaşık olacaktır.
- Bu yüzden çok işlemliliği de yansıtabilen 'Petri Net' gibi modeller, yapısal teknolojiler zamanında sıkça kullanılmışlardır.
- Genelde de çözümleyicinin bu gibi diyagramları çizmesini istemektense bir şekilde serbestçe tanımladığı bilgilerden BDYM araçlarınca iç belirtim olarak kullanılmak üzere otomatik olarak elde edilmişlerdir.
- Ancak çok işlemliliği sağlamanın bir yolu da bilinen durum makineleri tabanlı modelleri, değişik nesneler içerisinde ayrı ayrı oluşturup (her biri tek işlemli modeller olarak) bunların işbirliğine imkan tanımaktır.
- Durum diyagramlarına destek olacak ardışık diyagramlar gibi araçlar da kullanılmaktadır.
- Sonraki slaytlarda durum diyagramı ve ardışık diyagramı örnekleri verilmektedir.

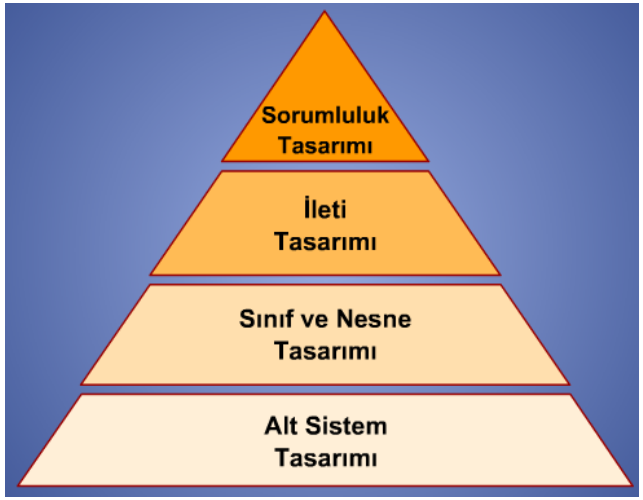
Durum Diyagramı



Ardışık Diyagramı



Nesneye Yönelik Tasarım



Nesneye Yönelik Tasarımda Katmanlar

- Nesneye Yönelik (NY) Tasarım, NY Çözümleme ile elde edilen modelin, yazılımın oluşturulması için yeterli tanımı sağlayacak şekile dönüştürülmesidir.
- Kavram olarak geleneksel tasarım ile benzerlikler gösterir.
- İki yaklaşımda da veri ve süreç belirtimi benzer şekilde yapılabilir.
- Algoritmik düzeyde süreç tanımları da farklı değildir.
- NY teknikler, yazılım tasarımının ilkeleri olan soyutlama, bilgi saklama, işlevsel bağımsızlık ve modülerliğe çok yatkındırlar.
- NY Tasarımı yanda görüldüğü gibi dört katman olarak inceleyebiliriz:
- Ayrıca bütün bu katmanlar bir 'saha nesneleri' bilgisinin oluşmasına dayalıdır.
- Belirli bir konuda yazılım geliştirilirken o konu ile ilgili genel modeller ve bunların uzantısı olan bir nesneler kitaplığı kurulmuş olmalıdır.
- Böyle bir yapı hazır değil ise eldeki projeler için gerekli altyapı olarak bir taraftan da saha altyapısının tasarımı yapılır.

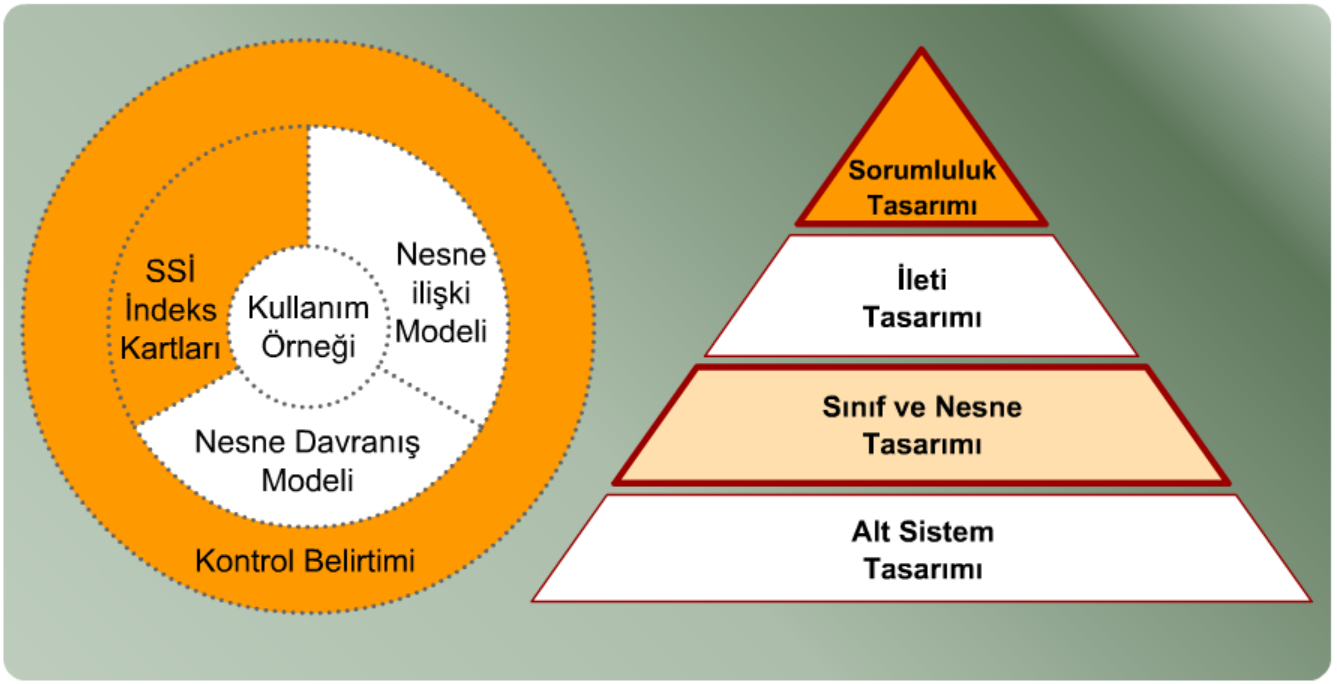
Nesneye Yönelik Tasarım

- Geleneksel tasarımın tersine, NY tasarımda NY çözümlemeden devir alınacak yapılar oldukça sadık kalınarak kullanılır.

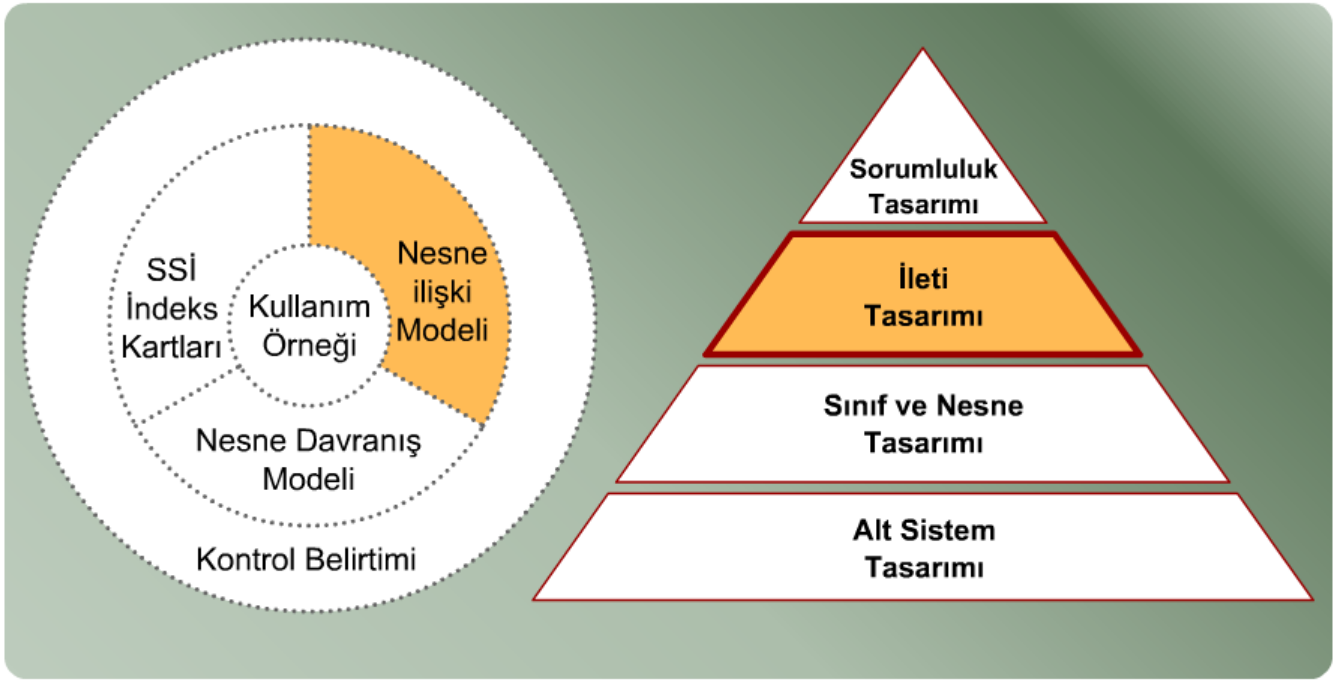
- Ancak ihtiyaçlar çalışmasında kullanılan NY olması gerekmeyen modeller bir şekilde NY yapılara yansımalıdır.

- Sonraki slaytlarda NY çözümlemeden NY Tasarıma geçişte kullanılan kavram ve ortamları gösterilmektedir.

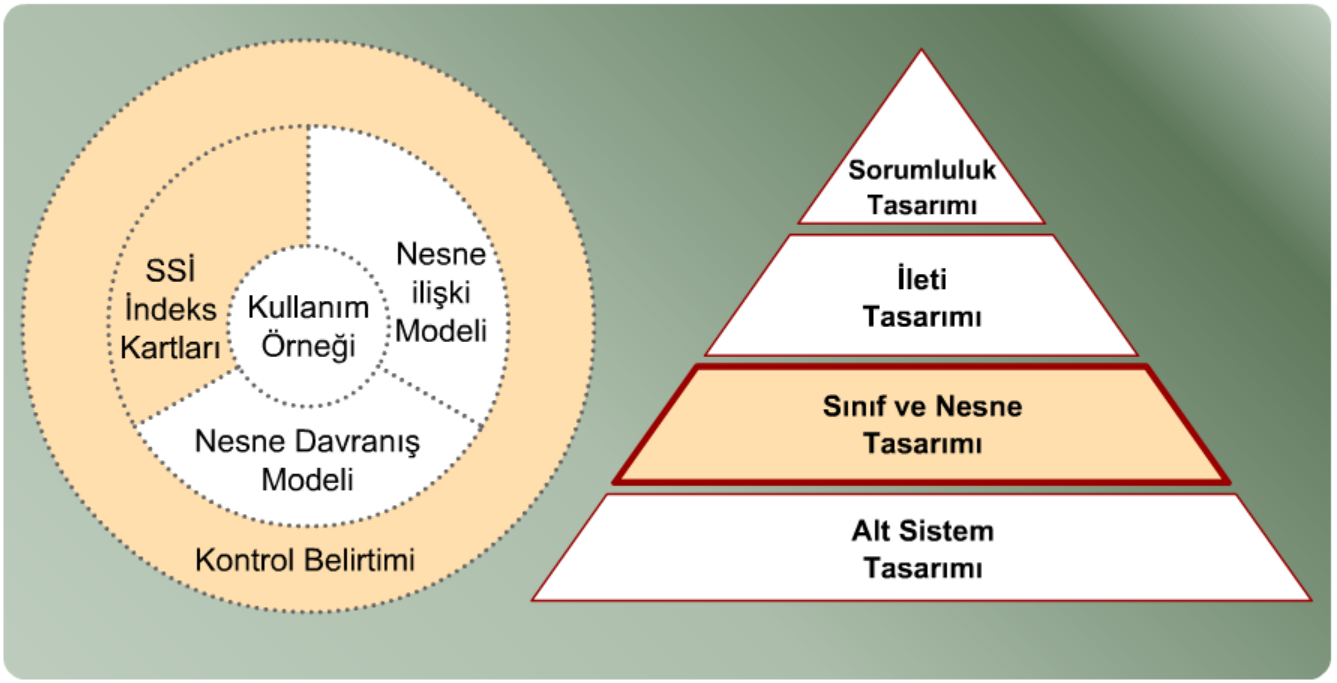
Nesneye Yönelik Yaklaşımda Çözümlemeden Tasarıma Geçiş



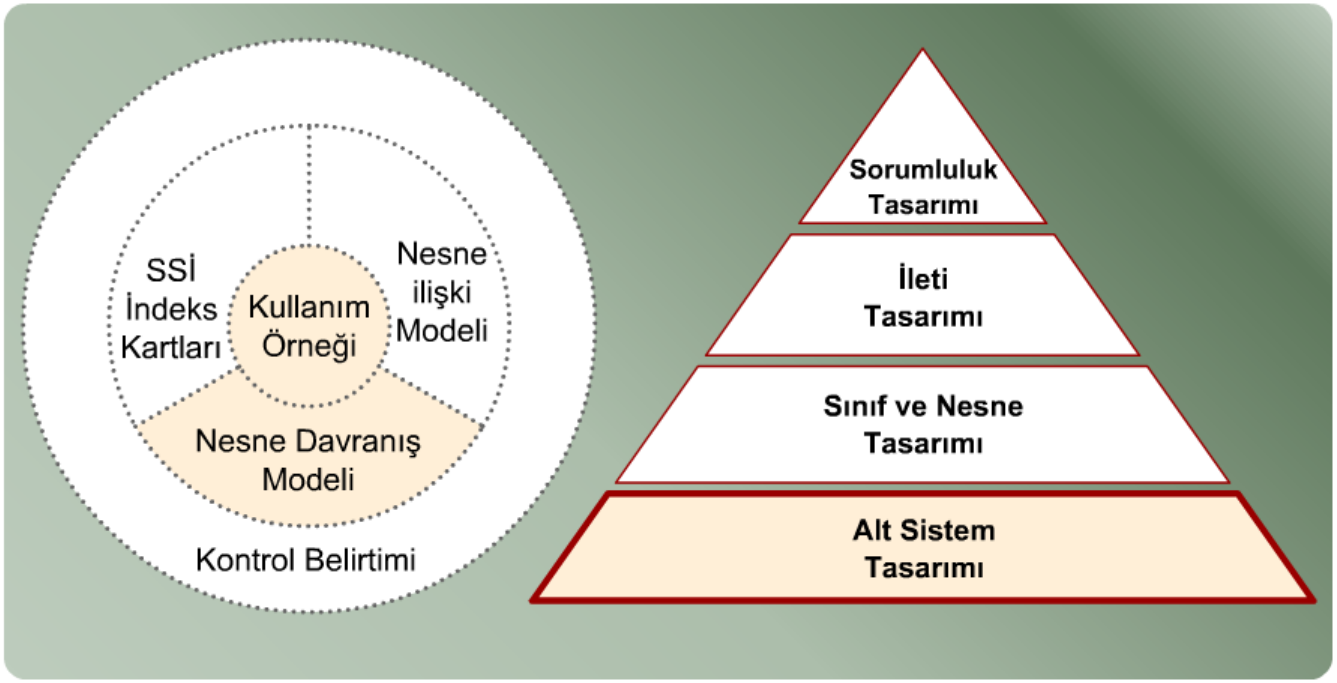
Nesneye Yönelik Yaklaşımda Çözümlemeden Tasarıma Geçiş



Nesneye Yönelik Yaklaşımda Çözümlemeden Tasarıma Geçiş



Nesneye Yönelik Yaklaşımda Çözümlemeden Tasarıma Geçiş



Nesneye Yönelik Yaklaşımda Çözümlemeden Tasarıma Geçiş

- Geleneksel ve NY Tasarım yaklaşımlarında en büyük fark, yapısal modellemelerde ortaya çıkar.
- Bütüncül bir yaklaşımla geleneksel tasarımda merkezi yapı ve kontrol hiyerarşisi kurulurken NY yaklaşımda dağıtık bir yapı söz konusudur.
- Sistem, etkileşimleri belirtilen bağımsız nesnelerin tümü olarak yapılanır.
- Kontrol kavramı da benzer şekildedir.
- Sistemin durumu ve dolayısıyla kontrol mekanizması, geleneksel yaklaşımda merkezidir.
- NY yaklaşımda ise iletilerle bağlantı kurulan dağıtık nesneler, sistemin genel durumunu belirler ve kontrol de dağıtık olarak sağlanır.

Nesneye Yönelik Yaklaşımda Çözümlemeden Tasarıma Geçiş

Birbirinden farklı olan geleneksel ve NY tasarım metodolojilerinin karşılaştırılmasında kullanılmak üzere 10 özellikten söz edilebilir:

1. Modül sıradüzen gösterimi
2. Veri tanımları belirtimi
3. İşlem mantığı belirtimi
4. Sondan sona süreç sıralamalarının gösterimi
5. Nesne durumları ve durum geçişlerinin gösterimi
6. Sınıf ve sıradüzen tanımları
7. Sınıflara işlemlerin tayin edilmesi
8. İşlemlerin detaylı tanımı
9. İleti bağlantılarının belirtimi
10. Sistem işlemlerinin tanımlanması

Nesneye Yönelik Tasarım Metodolojileri

Çözümlemede olduğu gibi tasarımda da bir çok farklı gibi görünen yöntem kullanılmaktadır. Aslında metodolojiler genelde çözümleme ve tasarım safhalarını diğer safhalar ile birlikte içerirler. Modüler yapının tasarım için öneminden daha önce söz edilmişti.

Bir tasarım metodunun modülerlik ve ilişkili NY kavramlarını desteklemesi için beş kriter ortaya atılmıştır:

1. **Çözünürlük:** Büyük bir problemin, çözülmesi daha kolay olan küçük problemlere ayrıştırılması yöntemidir.
2. **Bütünleştirme:** Geliştirilecek modüllerin ne kolaylıkla değişik sistemlerce kullanılabileceğidir.
3. **Anlaşılırlık:** Bir bileşenin, diğerlerine başvurmadan ne kolaylıkla anlaşılabildiğidir.
4. **Devamlılık:** Programda yapılacak değişikliğin yalnızca bir veya birkaç modülü etkilemesidir.
5. **Koruma:** Programda yapılacak değişikliğin yalnızca bir veya birkaç modülü etkilemesidir.

Nesneye Yönelik Tasarım Metodolojileri

Aynı şekilde modüler yapının sağlanması için de beş prensipten söz edilebilir:

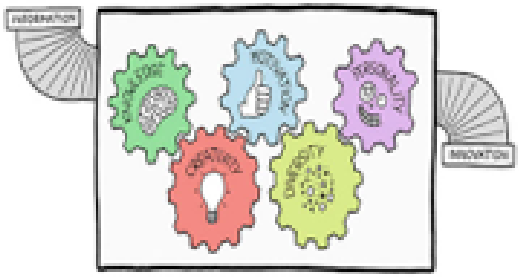
1. Modüler yapıyı destekleyici dil ögelerinin bulunması,
2. Az arayüz,
3. Arayüz içinde sınırlı bilgi akışı,
4. Arayüzün bağımsız olarak açıkça anlaşılabilmesi (örneğin merkezi yapılara başvurulmaması),
5. Bilgi saklama.

- Her ne kadar bu prensipler geleneksel tasarım için de geçerli ise de NY yaklaşımların daha etkili olarak bu prensipleri elde ettiği görülmektedir.
- Özellikle arayüz tanımları, bileşen tabanlı yaklaşımlarda iyice önemli olmaktadır.

NY Metodolojilerinin Tasarım Yöntemleri

Bazı Nesne Yönelimli metodolojilerinin tasarım yöntemlerini aşağıdaki şekilde maddelersek bunlar;

1. Booch Metodu
2. Coad ve Yourdon Metodu
3. Jacobson (OOSE) Metodu
4. Rambaugh (OMT) Metodu



Booch Metodu

Yapısal Planlama:

- Benzer nesneler ayrı kısımlarda toplanır.
- Nesneler soyutlama düzeylerine göre katmanlara ayrılır.
- Senaryolar tanımlanır.
- Tasarım öntipi oluşturulur.
- Öntip, kullanış senaryoları ile sınanır.

Taktik Tasarım:

- Özellik ve işlem kullanım kurallarının tanımını,
- Bellek idaresi, hata mesajları gibi altyapı işlevleri ile ilgili kuralların tanımını,
- Kuralların anlamını tanımlayan bir senaryonun geliştirilmesini,
- Her kural (politika) için bir öntipini,
- Öntip iyileştirmesini,
- Her politikanın, yapısal vizyonu iletmesi açısından gözden geçirilmesini kapsar.

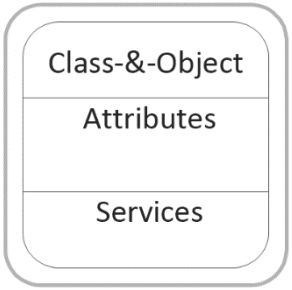
Coad ve Yourdon Metodu

Problem Ortamı Bileşeni

- Ortamı oluşturan bütün sınıflar gruplandırılır.
- Uygulama sınıfları için hiyerarşi oluşturulur.
- Uygun düştükçe kalıtım basitleştirilir.
- Verimlilik için tasarım uyarlanır.
- Alt düzey nesneleri, gerektikçe eklenir ve iyileştirilir.
- Tasarım gözden geçirilir ve çözümlemeye yapılacak ilaveler sorgulanır.

İnsan Etkileşim Bileşeni

- Kullanıcılar belirlenir.
- Görev senaryoları geliştirilir.
- Kullanıcı komutları sıradüzeni tanımlanır.
- Kullanıcı etkileşim sıralandırması iyileştirilir.
- İlgili sınıflar ve sıradüzenleri tasarlanır.
- Grafik arayüz sınıfları ile bütünleştirilir.



Coad ve Yourdon Metodu

Görev Yönetimi Bileşeni

- Görev çeşitleri (olay veya saat ile tetiklenen gibi) tanımlanır.
- Öncelikler oluşturulur.
- Bir görev, diğerlerinin yöneticisi olarak atanır.
- Her görev için uygun nesneler tasarlanır.

Veri Yönetimi Bileşeni

- Veri yapıları tasarlanır.
- Veri yapılarının idaresi için gerekli servis işlemleri tasarlanır.
- Veri idaresi için araçlar tanımlanır.
- Uygun sınıf ve hiyerarşiler tasarlanır.

Jacobson (OOSE) Metodu

1. Çözümleme modelinde gerçek dünyaya uygunluk ayarlamaları.
2. Temel tasarım nesnesi olarak ‘öbek’lerin tanımlanması:
   - İlgili çözümleme nesnelerinin uygulanacağı bir blok tanımlanır.
   - Arayüz, varlık ve kontrol blokları tanımlanır.
   - İşletim sırasında blokların nasıl haberleşeceğim açıklanır.
   - Bloklar arası uyarı işaretleri ve bunların haberleşeceği açıklanır.
3. Bloklar arası uyarı iletilerini gösteren etkileşim diyagramları çizilir.
4. Bloklar, alt sistemler altında düzenlenir.
5. Tasarım gözden geçirilir.

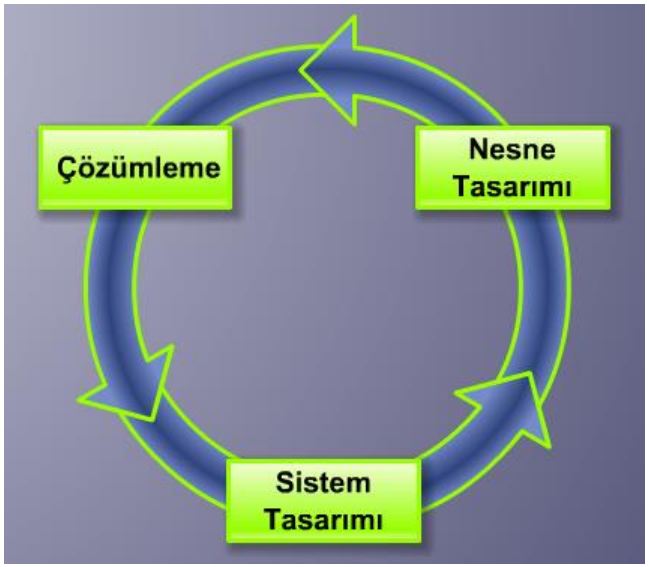
Rambaugh (OMT) Metodu

1. Sistem Tasarımı
   - Çözümleme modeli alt sistemlere ayrılır.
   - Problemin gerektirdiği eşzamanlılıklar tanımlanır.
   - Altsistemler ve süreçleyiciler görevlere dağıtılır.
   - Veri idaresi için temel strateji seçilir.
   - Ortak kaynaklar ve bunlara erişim kontrol edecek yapılar tanımlanır.
   - Sistem için uygun kontrol mekanizması tasarlanır,
   - Sınır koşulları göze alınır.
   - Parametreler arası çıkar çatışmaları gözden geçirilir.

2. Nesne Tasarımı
   - Çözümleme modelinden işlemler seçilir.
   - Her işlem için yordam tanımlanır.
   - Yordamlar için uygun veri yapıları tanımlanır.
   - İç sınıflar tanımlanır.
   - Nesne düzenlemesi veri ulaşımı ve hesaplama verimi açısından gözden geçirilir.
   - Sınıf özellikleri tasarlanır.

3. Sistem tasarımındaki kontrol mekanizmaları uygulanır.
4. Sınıf yapıları kalıtımı kuvvetlendirici yönde ayarlanır.
5. Nesne ilişkileri için ileti tasarımı yapılır.
6. Sınıflar modüller altında toplanır.

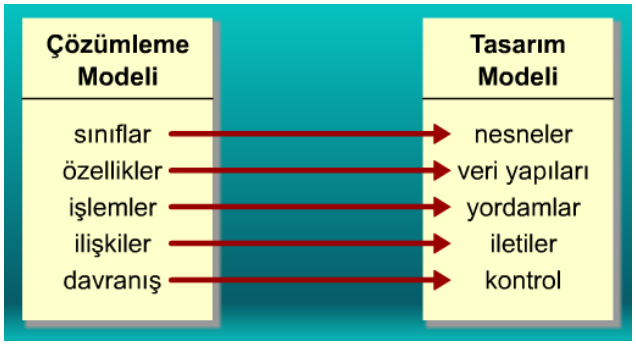
Genel Olarak NY Metodolojiler



Nesneye Yönelik Çözümlemeden Tasarıma Geçiş Çevrimi

- NY çözümlemede olduğu gibi, NY tasarımda da değişik yaklaşımlar olmasına rağmen benzer işlemler uygulamaktadır.
- NY yaklaşımlarda bazen çözümleme ve tasarım arasındaki ayırımı yapmak zorlaşır.
- NY Çözümleme, bir sınıflandırma işlemidir.
- Problem yapısı, nesne sınıfları olarak organize edilir ve nesne ilişkileri ile problemin davranışı modellenir.
- Çözümlemeden tasarıma geçiş süreci, genel hatlarıyla yanda gösterildiği gibi bir çevrim içerir.
- Tasarım ise tanımlanmış sınıflardan çıkarılacak nesneleri tanımlar ve bu nesnelerin birbirleri ile ilişkisini gösterir.
- Ayrıca nesnelerin davranışları ve haberleşmeleri belirtilir.

Genel Olarak NY Metodolojiler



Nesne Tasarımında
Çözümleme Modelinden Geçiş

- Tasarımcı, önce sistemin bütününü ele almalıdır.
- Kullanıcı istekleri ve bunların gerçekleştirilmeleri için gerekli alt sistem karakteristiklerinin tanımlanması gerekir.
- Alt sistemlerin tasarımı sırasında Coad Yourdon metodolojisinde sözü geçen dört önemli tasarım bileşeni tanımlanır.
- Daha sonra nesne tasarımına sıra gelir.
- Yanda gösterildiği gibi, Sınıf Sorumluluk İşbirliği (SSİ) modelindeki öğeler, tasarıma dönüştürülür.

Genel Sistem Tasarımı

- Değişik yaklaşımlardan söz edilmiş ise de sistem tasarımı konusunda Rambaugh yaklaşımı burada tercih edilmiştir.

- Daha önce tanımlanan sistem tasarımı adımlarında yapılacak ayrıntılı işlemler, bu bölümde açıklanmaktadır.

- Çoğu yaklaşım, önce sistem/alt sistem ayrıştırması ve bu ayrıştırılan bileşenler arası arayüzün tanımı ile tasarıma başlar.

Çözümleme Modelinin Ayrıştırılması

- Alt sistemleri tanımlarken, sistemin sınıf, ilişki ve davranışlarını 'yapışık gruplar' olarak bir araya getirmek gerekir.
- Bir alt sistem içerisinde bulunacak bileşenlerin ortak yanı bulunmalıdır.
- Aynı sistem işlemini yerine getirmek, aynı donanım birimi içerisinde bulunmak veya aynı kaynakları idare etmek gibi.
- Sistem açısından, alt sistemler sorumluluklarına göre tanımlanırlar.
- Bu tanımlama yapılırken de aşağıdaki tasarım kriterleri göz önünde bulundurulur:
  - Sistemin gerisi ile yapılacak haberleşmenin tümü, iyi belirlenmiş bir arayüz üzerinden olmalıdır.
  - Sınıflar, yalnızca aynı alt sistem içerisinde işbirliği yapmalıdırlar (Yalnızca küçük sayıdaki haberleşme sınıfları dışında).
  - Alt sistem sayısı küçük tutulmalıdır.
  - Karmaşıklığı azaltmak üzere alt sistemler kendi içlerinde bölünebilirler.
  - Haberleşen alt sistemler istemci/sunucu gibi bağlantı protokollerine sahip olabilirler.

Eşzamanlılık ve Alt Sistem Belirleme

- Nesne davranış modeli açısından, iki nesne aynı zamanda aktif değilse bir eşzamanlılık yoktur.
- Bu iki nesne aynı donanım (süreçleyici) üzerinde uygulanabilirler.
- Ancak eğer iki nesne aynı zamanda ve asenkron olarak olaylara karşılık veriyorlarsa eşanlılık vardır.
- Eşanlı alt sistemler için iki yerleştirme seçeneği vardır:
  1. Ayrı süreçleyicilere yerleştirme,
  2. Aynı süreçleyiciye yerleştirme ve işletim sisteminin eşzamanlılık desteğini kullanma.

Görev Yönetimi Bileşeni

- Eşzamanlı görevleri yöneten nesnelerin tasarımı için Coad ve Yourdon aşağıdaki stratejiyi önermektedirler:
  1. Görevin özellikleri ortaya çıkarılır:
     - başlatılma (olay veya saat ile tetiklenme..)
     - öncelik (görevlerin birbirinden önce başlatılması)
     - önemlilik (kritik bir görevin kesilmemesi)
  2. Bir yönetici 'görev' ve ilişkili nesneler tanımlanır.
  3. Yönetici ve diğer görevler bütünleştirilir.

- Görevler, standart tasarım modeline girmeden önce bir şablon ile şekillendirilirler.

- Burada görevin ismi, açıklanması, önceliği, sorumlu olduğu işlemler, nasıl başlatıldığı ve haberleşmesindeki girdi/çıktı veri değerleri bulunur.

Veri Yönetim Bileşeni

- Sistem açısından bakıldığında tek bir veri tabanı yönetim sistemi ile her türlü ihtiyaç karşılanmaktadır.
- Ancak sistem düzeyindeki ihtiyaçlar ile veri yapılarının işlemleri için gereken alt düzeydeki ihtiyaçların ayırt edilmesi gerekir.
- Nesnelerin özelliklerinin kaydedilebilmesi ve yeniden okunabilmesi gerekir.
- Nesneye yönelik veritabanı yönetim sistemleri artık ticari ürünler olarak piyasaya sürülmüşlerse de henüz fazlaca kullanılmamışlardır.
- Çoğu zaman uygulamada nesneler bir şekilde ilişkisel veri tabanlarında saklanmaktadır.
- Nesnelerin kayıt ve yeniden okunma işlemlerini yönetmek üzere önlemler alınmalıdır.

Kaynak Yönetimi Bileşeni

- Genel olarak, alt sistemler kaynaklar için yarışırlar.
- Kaynakların kontrolsüz bir şekilde, isteyen birime atanması sorun çıkarır.
- 'Gözetleyici Nesneler' ile bir kaynağın kullanımı için hangi alt sistemin nasıl ve ne zaman erişime kavuşabileceği düzenlenebilir.
- Alt sistemler önce bu gözetleyiciler ile haberleşerek izin aldıktan sonra kaynağa erişebilirler.
- Bu erişim politikası çizilmeli, gözetleyici nesnelerin bu politikalar doğrultusunda işlemleri tanımlanmalıdır.

Kullanıcı Arayüzü Bileşeni

- Kullanım örnekleri gibi ortamlardan yola çıkarak kullanıcı ve sistem etkileşimlerini tanımlamak doğaldır.
- Roller ve kullanım senaryoları tanımlandıktan sonra, komut hiyerarşisi ortaya çıkarılır.
- Komut hiyerarşisi, bütün kullanım örnekleri kapsanana kadar ana menüden başlayarak alt menülerin ve daha alt kumanda yapılarının tanımlanması ile tasarlanır.
- Bu konuda yardımcı olabilecek hazır nesneler çoğu geliştirme ortamında mevcuttur; örneğin Visual BASIC veya Delphi.

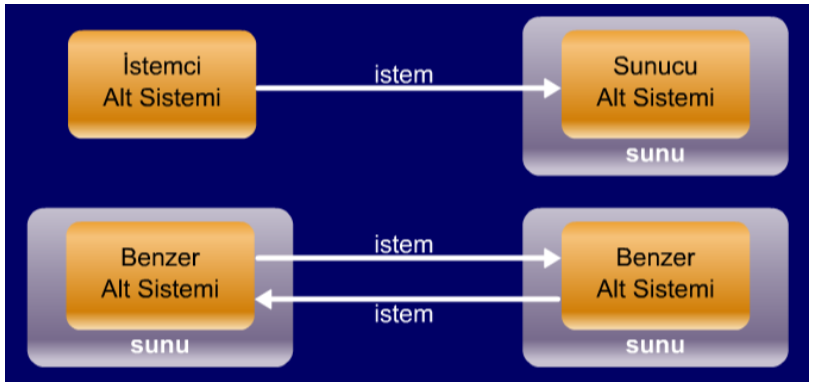
Alt Sistemler Arası İletişim

Alt sistemler tanımlandıktan sonra bunlar arasındaki iletişim yöntemlerinin de belirtilmesi gerekir. Alt sistemlerde genelde iki tür haberleşme tipi yaygındır. Bu haberleşme tipleri, istemci/sunucu veya benzerler arası bağlantı tiplerinden oluşur. Alt sistemler arasındaki haberleşmeyi de istem ve karşılığında sunulacak bir hizmet şeklinde genellersek, bir hizmet sunusu için aşağıdaki tasarım adımları uygulanmalıdır:

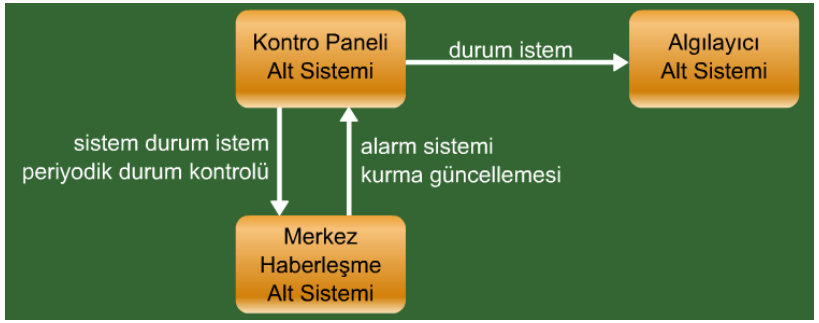
1. Alt sistem ile işbirliğinde bulunanlarca yapılabilecek istemler sıralanır. İstemler alt sistemlere göre gruplanır ve sunular ile ilişkilendirilir.
2. Her sunu için nesne işlemleri belirlenir ve bunlar alt sistem dışından olmamalıdır.
3. Her sununun yer alacağı Şekil 9.16'daki gibi bir tablo oluşturulur. Bu tablodaki sütunlar:

- Tip (İstemci/sunucu vb.)
- İşbirlikçiler: Sunu ile ilgili alt sistemler
- Sınıf: Sunu için gerekli 'nesne işlemleri'ni içeren sınıf isimleri
- İşlem: Belirtilen sınıf içerisinde sunuyu gerçekleştiren işlemler
- İleti biçemi şeklindedir. Eğer etkileşim karmaşık ise, bir çeşit '**olay akışı**' olan diyagram çizilir.

Alt Sistem İşbirliği



Alt Sistem İşbirliği Modeli Örnekleri



Alt Sistem İşbirliği İçin Olay Akış Diyagramı Örneği

Nesne Tasarımları-Nesne Tanımı

Tasarımda bir nesnenin tanımı için iki öğe vardır:

Protokol Tanımı
Nesnenin alabileceği her ileti ve sonucunda yapması gereken işlemin tanımlanması ile oluşan nesne ara yüzüdür.

Uygulama Tanımı
İleti sonucunda başlatılacak her nesne işlemi için süreç belirtimi yapılması ve nesne içi özel veri yapılarının (nesne özelliklerinin) ayrıntılarının belirtimidir.

Burada daha önce karşılaşılan 'ne' ve 'nasıl' soruları yine geçerlidir. Protokol tanımı, '**ne**' sorusuna, uygulama tanımı ise '**nasıl**' sorusuna cevap verir.

Nesne Tasarımları-
Yordam ve Veri Yapısı Tasarımı

- Veri yapıları ve yordamlar birlikte tasarlanırlar.
- Sistem düzeyinde yordam, bir bütünlük içerisindeki işlemi gerçekleştiren süreç olarak tanımlanabilir.
- Ancak alt sistem ve nesneler düzeyinde yordamlar biraz daha farklı ele alınırlar.
- Üst düzeydeki işlemlerin daha küçük işlemlere ayrıştırılarak gerçekleştirilmeleri söz konusu olur.
- Burada yine Rambaugh'dan alınan eniyileme önerilerinden söz etmek yerinde olur:
  - Nesne ilişki modelini verimlilik ışığında gözden geçirin, gerekirse tekrarlanan yapılar tanımlayın.
  - Nesne içi veri yapılarını ve işlem yordamlarını yine verimlilik ışığında gözden geçirin
  - Türetilen bilgiyi saklamak üzere ve fazla işlemi önlemek için gerekiyorsa ek özellikler tanımlayın.

Tasarım Kalıpları

- Diğer alanlarda da tasarımcılar 'kalıp' fikrini başarı ile uygulamaktadırlar.
- Örneğin ev mimarisinde sıkça kullanılacak bir kapı ve belirli bir mesafeden sonra bir pencere kümesi, bir kalıp olarak tanımlanabilir ve her oda tasarımında bu kalıp kullanılmağa çalışılabilir. Böylece yeniden kullanılabilirlik arttırılmış olur.
- Yerleşmiş bir kalıp kullanımı için kalıplar, dört özellikle tanımlanabilir:
  - Kalıbın ismi
  - Genelde kalıbın uygulandığı problem türü
  - Kalıbın karakteristiği (tasarımda hangi özelliklerin değiştirilmesi ile kalıbın değişik problemlere uyarlanacağı)
  - Kalıbın uygulanmasının etkileri
- NY tasarımda kalıpları oluşturmak için iki mekanizma yararlıdır.
- Kalıtım yolu ile bir kalıp tanımlandıktan sonra değişik problemlere uygulanabilir.
- Diğer mekanizma ise nesneleri bir araya getirerek oluşturulacak 'içerim'dir.
- Karmaşık bir işlemi (alt sisteme karşı düşebilir) bir nesneler bütünü ile uygulama yöntemidir.
- Seçenek olduğu taktirde içerim, kalıtıma karşı tercih edilmektedir.

Özet

- NY kavramların temeli olarak, aşağıdaki dört özelliği sıralayabiliriz:
  - Kimlik (Identity)
  - Sınıflama (Classification)
  - Kalıtım (inheritance)
  - Çok Şekillilik/Biçimlilik(Polimorphism)
- Kalıtım: Hiyerarşik ilişkiye dayanan sınıflar arasında özellik ve işlevlerin paylaşılmasıdır.
- Çok şekillilik: Aynı işlem, farklı sınıflarda değişik davranabilir mantığına dayanır.
- Nesne yönelimli sistemleri destekleyen kavramlar;
  - Soyutlama (Abstraction),
  - Bilgi Gizleme (Encapsulation),
  - Paylaşım (Sharing),
  - Altsınıf (Subclass) ve "Aggregation" dır.
- Uygulama Alanının Çözümlenmesinde Amaç, uygulama alanını anlamak ve elde edilen bilgileri analiz modeline taşımaktır.

Özet

Gereksinim Belirleme Çalışmalarında iki teknik vardır;

- Kullanım Durumları ve
- Sınıf Sorumluluk İşbirlikçi modelleridir.

Nesneye Yönelik Tasarımda Katmanlar şu şekildedir:

- Sorumluluk Tasarımı
- İleti Tasarımı
- Sınıf ve Nesne Tasarımı
- Alt Sistem Tasarımı

- NY tasarım metodolojilerinin karşılaştırılmasında kullanılmak üzere 10 özellikten söz edilebilir:
- Bir tasarım metodunun modülerlik ve ilişkili NY kavramlarını desteklemesi için beş kriter ortaya atılmıştır.
- Nesne Yönelimli metodolojilerinin tasarım yöntemleri; Booch Metodu, Coad ve Yourdon Metodu,Jacobson (OOSE) Metodu ve Rambaugh (OMT) Metodu'dur.

Sorular

1. Nesneye Yönelik Kavramlar nelerdir? Açıklayınız.
2. Kalıtım ve Çoklu Kalıtım arasındaki farkı açıklayınız.
3. Nesne Yönelimli Sistemleri Destekleyen Kavramlar nelerdir? Açıklayınız
4. Sınıf/Sorumluluk/İşbirlikçi Modelini açıklayınız.
5. Alt Sistem İşbirliğini açıklayınız.
6. Nesne Yönelimli metodolojilerinin tasarım yöntemleri kaç tanedir? Bunları açıklayınız.
7. Tasarımda Nesne kavramı nedir?

Kaynaklar

“Software Engineering A Practitioner’s Approach” (7th. Ed.), Roger S. Pressman, 2013.

“Software Engineering” (8th. Ed.), Ian Sommerville, 2007.

“Guide to the Software Engineering Body of Knowledge”, 2004.

” Yazılım Mühendisliğine Giriş”, TBİL-211, Dr. Ali Arifoğlu.

”Yazılım Mühendisliği” (2. Basım), Dr. M. Erhan Sarıdoğan, 2008, İstanbul: Papatya Yayıncılık.

Kalıpsiz, O., Buharalı, A., Biricik, G. (2005). Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı Nesneye Yönelik Modelleme. İstanbul: Papatya Yayıncılık.

Buzluca, F. (2010) Yazılım Modelleme ve Tasarımı ders notları (http://www.buzluca.info/dersler.html)

Hacettepe Üniversitesi BBS-651, A. Tarhan, 2010.

Yazılım Proje Yönetimi, Yrd. Doç. Dr. Hacer KARACAN

Ödev

Mimari Tasarım Hakkında Araştırma Yapınız.
Yazılım Mimarisinde Kullanılan Stilleri Araştırınız.