



YAZILIM MİMARİLERİ DERSİ

BİLGİSAYAR PROGRAMCILIĞI



Yazılım Nedir ?

Yazılım,

1. Komutlar : yürütüldüğü zaman istenen fonksiyon ve performansı sağlar,
2. Veri Yapıları : programa verileri yeteri kadar işlemek için izin verir,
3. Belgeler : operasyonları ve program kullanımını tanımlar...

Yazılım Nedir ?

- Yazılım = Mantık +
(algoritma)
Veri +
(test verisi, bilgi?)
Belge +
(dokümanlar)
İnsan +
(kullanıcı, geliştirici)
Program
(kod)

- “Bilgisayar sisteminin donanım bileşenleri dışında kalan her şey”

Yazılım Mühendisliği nedir ?

IEEE Bilgisayar Topluluğunun Yazılım Mühendisliği Tarifi:

“Mühendislik eylemlerinin, (Geliştirme, İşletme, ve Bakım), disiplinli, sistematik ve nicelikli bir şekilde yazılıma uygulanması” olarak tanımlamıştır...

Yazılım Uygulamaları

Sistem Yazılımları : Diğer programlara hizmet sağlayan programlar...(Derleyiciler, İşletim Sistemleri)

Gerçek – Zamanlı Yazılımlar : Anlık verileri izlemek için kullanılan programlar.(Un fabrikası üretim izleme programı)

Ticari Yazılımlar : Ticari verileri işleyebilen programlara denir. (Brill, Optimasyon vb..)

Bilimsel ve Mühendislik Yazılımlar : Bilimsel ve mühendislik alanlarında kullanılan programlar. (Autocad, Matlab vb.)

Yazılım Uygulamaları

Gömülü Yazılımlar : Genellikle donanımları kontrol etmek için kullanılan ve rom hafızaya sahip programlar. (Mikrodalga fırın tuş takımı programı gibi)

PC Yazılımları : Kullanıcıların işlemlerini yaptıkları genel amaçlı programlar. (Word, Excel vb.)

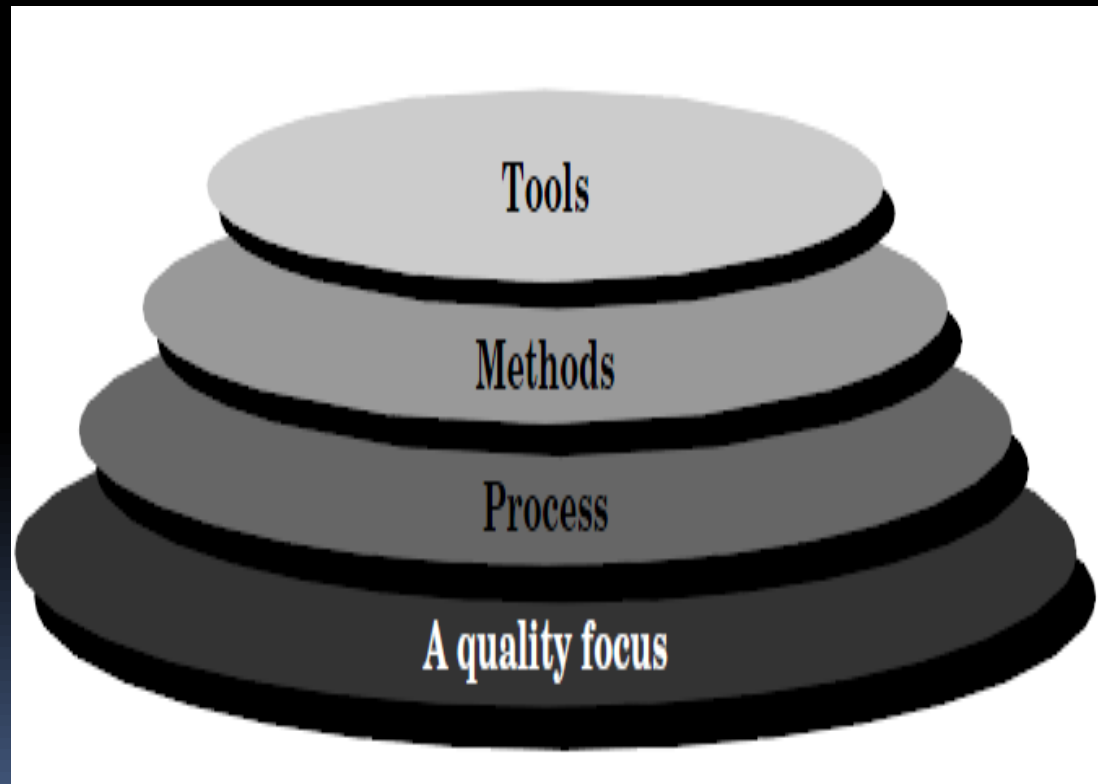
Web Tabanlı Yazılımlar : Web programcılığı ile kodlanmış yazılımlar. (Facebook)

Yapay Zeka Yazılımları : Kompleks problemlerin çözümü için kullanılan yazılımlar. (Uzman Sistemli Yazılımlar, Doku Tanımlama, Yapay Sinir Ağları vb.)

Yazılım Teknolojisi

Yazılım teknolojisi katmanlı bir yapıya sahiptir.

- Aygıtlar
- Metotlar
- İşlemler
- Kalite



Yazılım Problemi Çözülürken

- Çözülecek problem nedir ?
- Problem çözülürken ne kullanılacak ?
- Çözüm nasıl gerçekleştirilecek ?
- Çözüm nasıl oluşturulacak?
- Problem oluşturulup çözülürken hatalar nasıl ortaya çıkarılacak ?
- Kullanıcı tarafından istenilen değişiklikler, düzeltmeler ve adaptasyon süreçleri nasıl desteklenecek?



Yazılım Süreçleri

- Gereksinim Belirtilimleri (Requirements Specifications)
- Tasarım (Design)
- Kodlama (Coding)
- Test (Test - Validation)
- İşletim ve Bakım (Operation and Maintenance)
- Proje Yönetimi
- Kalite Yönetimi
- Düzenleşim (Konfigürasyon)/Değişim Yönetimi

Program Yazım Aşamaları

- **Tanımlama Aşaması :** Programcı başarılı bir sistem için ; kullanılacak verileri, hangi fonksiyon ve performans arzulandığını, umulan sistem davranışlarını, hangi arayüzlerin kullanılacağını, hangi tasarım kısıtları olduğunu, hangi tanımlama sorunları olduğunu tanımlar...

Program Yazım Aşamaları

- **Gelişim Aşaması** : Programcı gelişim aşamasında, verilerin nasıl yapılandırılacağını, yazılım mimarisinde fonksiyonların nasıl uygulanacağını, prosedürsel detayların nasıl uygulanacağını, arayüzlerin nasıl karakterize edileceğini, tasarımın program diline nasıl aktarılacağını ve yazılımın nasıl test edileceğini tanımlamaya çalışır.

Program Yazım Aşamaları

- **Destek Aşaması** : Programcı bu adımda genellikle hata düzeltme, adaptasyon gibi işleri halletmeye çalışır. Bu aşamada kendi içinde 4 aşamada tanımlanır.
 1. **Düzeltilme** : Yanlış ve eksikliklerin giderilmesi
 2. **Adaptasyon** : Başka ortamlara uyum sağlaması (Örn. Hem windows hemde linux ortamında çalışması)
 3. **Geliştirme** : İhtiyaçlar doğrultusunda programda gelişimler sağlanması
 4. **Korunma** : Programın kullanıcı veya işletim sisteminden doğacak hatalara karşı korunması ve varsayılan ayarlara dönebilme...

Yazılım Geliştirme Modelleri

Yazılım ihtiyaçlarının giderek büyümesi, yazılım geliştirme faaliyetlerinde kullanılmak üzere metodolojilerin gelişimini de ortaya çıkartmıştır. Yazılım teknolojilerinin gelişmesi ile, var olan model ve metodolojiler de gelişmekte ve yeni modeller ortaya çıkmaktadır. Uygun yazılım geliştirme modelleri kullanılması, yazılımın daha emniyetli, doğru, anlaşılabilir, test edilebilir ve bakım yapılabilir olarak geliştirilmesinde çok önemli rol oynar. Daha emniyetli yazılımların daha kısa sürede, daha az bütçeyle ve en önemlisi daha az hatayla geliştirilmesi için sürekli yeni teknolojiler ve modeller bulunmaya çalışılmaktadır.

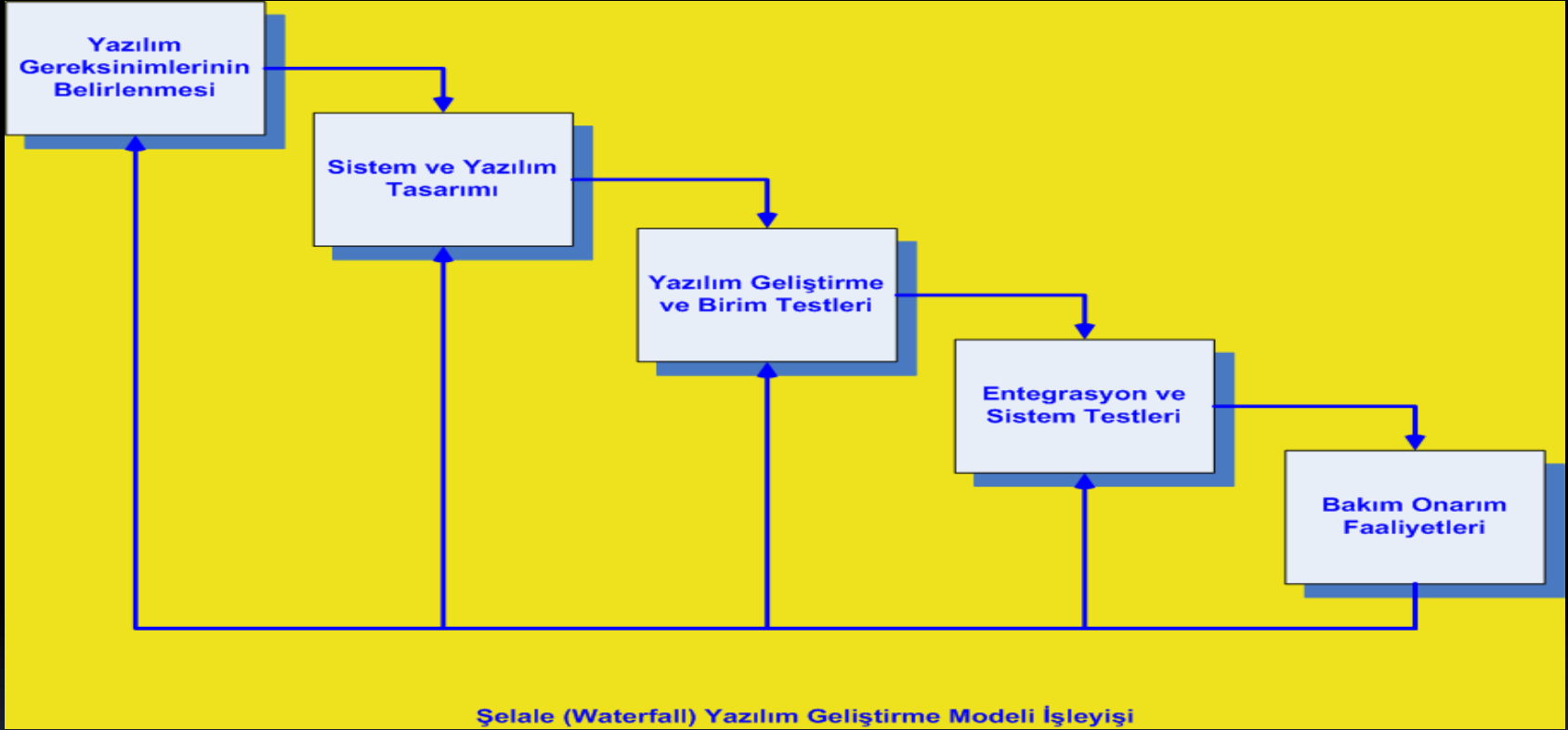
Yazılım Geliştirme Modelleri

- Şelale Modeli
- Spiral Modeli
- Artımlı Geliştirme
- Döngüsel Model
- Evrimsel Geliştirme

Şelale Modeli

- Şelalenin her basamağında yer alan aktiviteler eksiksiz olarak yerine getirilir. Bu bir sonraki basamağa geçme şartı budur.
- Her safhanın sonunda bir doküman oluşturulur. Bu yüzden şelale modeli doküman güdümlüdür.
- Yazılım süreci lineerdir, yani bir sonraki safhaya geçebilmek için bir önceki safhada yer alan aktivitelerin tamamlanmış olması gerekir.
- Kullanıcı katılımı başlangıç safhasında mümkündür. Kullanıcı gereksinimleri bu safhada tespit edilir ve detaylandırılır. Daha sonra gelen tasarım ve gerçekleştirme (kodlama) safhalarında müşteri ve kullanıcılar ile diyaloga girilmez.

Şelale Modeli



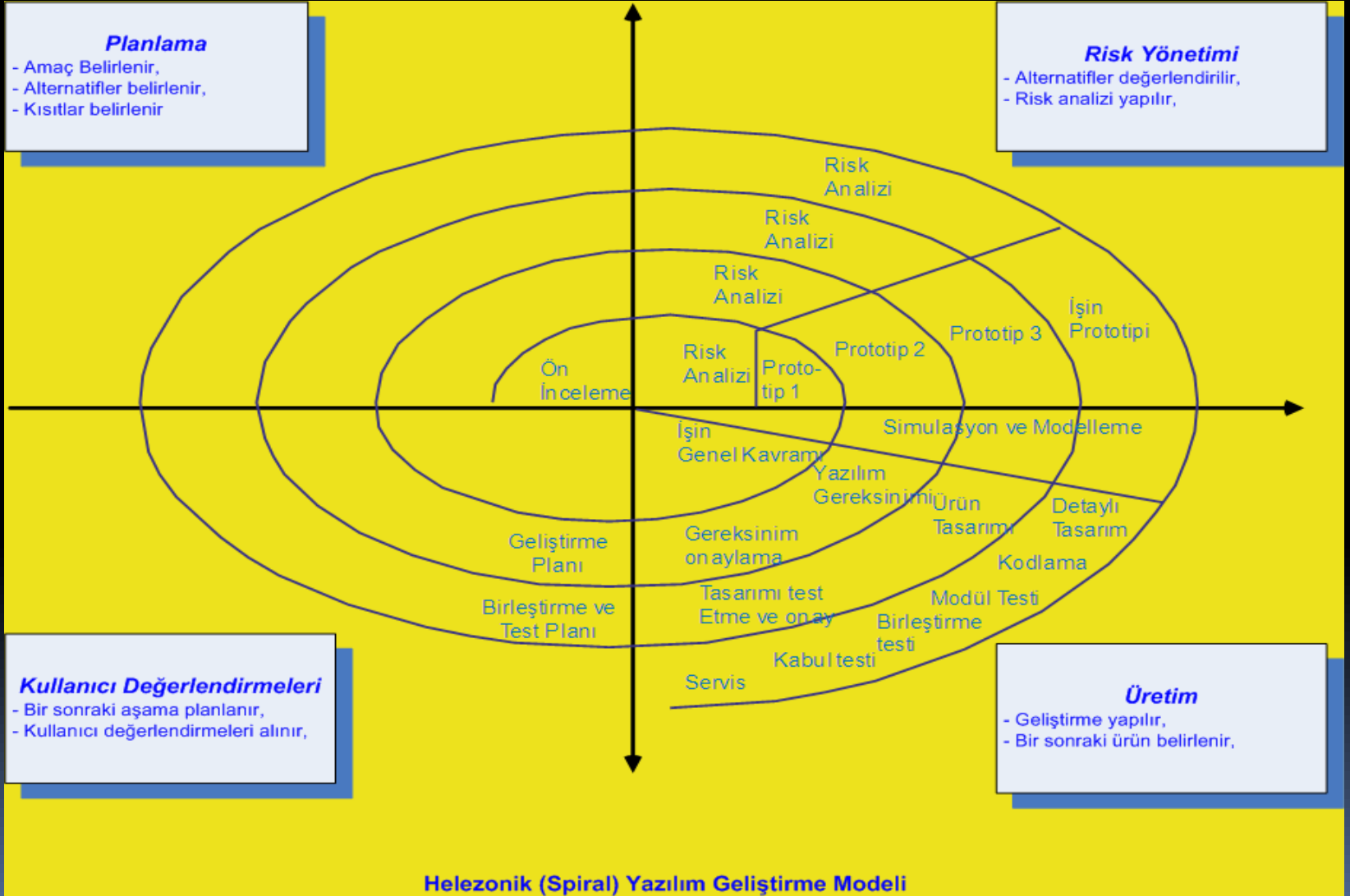
Şelale modelinde yazılım, aşamaları en az birer kez tekrarlanarak geliştirilir. Çok iyi tanımlanmış ve üretimi az zaman gerektiren projeler için uygun bir model olmakla birlikte günümüzde kullanımı gittikçe azalmaktadır.

Spiral Modeli

Spiral yazılım geliştirme modeli temel olarak dört ana bölüm içerir. Bunlar, planlama, risk yönetimi, üretim ve kullanıcı değerlendirmeleri olarak tanımlanabilir.

- ▣ Planlama, üretilecek ara ürün için işin planlanması, amaç ve kısıt ve alternatiflerin belirlenmesi, bir önceki adımda üretilmiş olan ürün ile tümleştirme yapılması faaliyetlerini içerir.
- ▣ Risk yönetiminde, alternatifler değerlendirilir ve risk analizi yapılır.
- ▣ Üretim, planlanmış ara ürünün geliştirildiği aşamadır.
- ▣ Kullanıcı değerlendirmesi kısmında, ara ürün hakkında kullanıcıların test ve değerlendirmeleri yapılır.

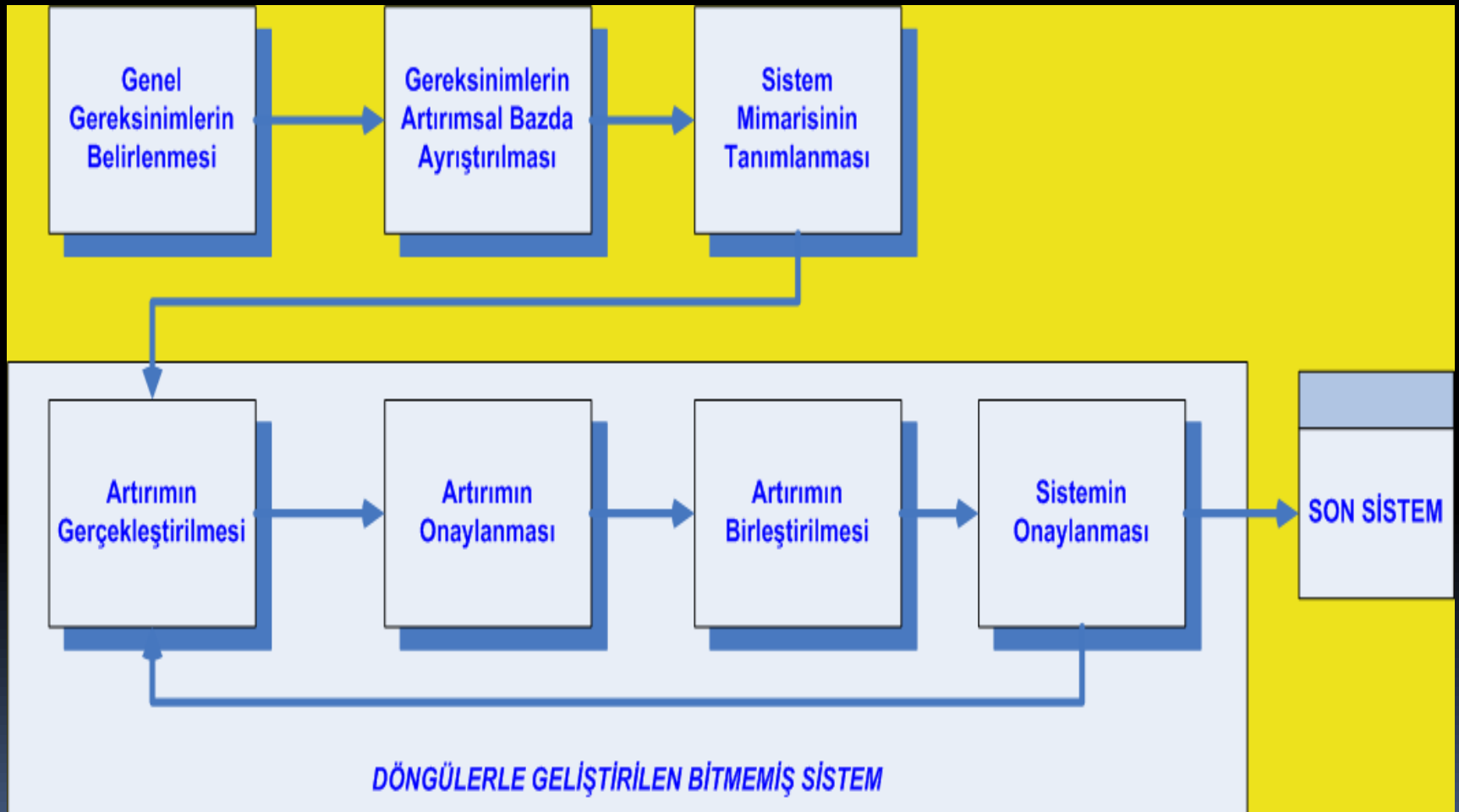
Spiral Modeli



Artımlı-Geliştirme Modeli

Artırımsal model, yazılımın küçük parçalara ayrılarak dögüsel olarak geliştirilmesi fikrine dayanır. Proje süresi, artırım (veya dögü) olarak tanımlanan küçük zaman dilimlerine bölünür. Proje bir çok dögünün gerçekleştirilmesi ile ilerler. Her dögünün sonunda, projeye ait planlanmış çıktılar elde edilir ve yazılıma yeni bir fonksiyonalitye eklenir. Bu sayede yazılım artırımsal olarak geliştirilir. Projenin bir dögüde henüz tümleştirme süreci sonlanmamışken , diğer bir dögünün dögünün tasarım süreci başlayabilir. Dolayısı ile, bu model yazılım geliştirmenin doğasına daha uygun olarak görünmektedir. Her dögüde yeni bilgi ve tecrübeler edinilir ve bunlar projenin geliştirilmesi aşamasında çok değerli katkılar yapar. Artırımsal modelin en önemli avantajlarından biri, projenin ilk safhalarında elde edilen çıktıların projenin ilerleyen aşamalarında değişikliğe uğraması halinde bile büyük bir maliyete neden olmadan bu değişikliklerin yapılabilir olmasıdır.

Artımlı-Geliştirme Modeli




Döngüsel Modeli

Döngüsel model, artımlı- geliştirme modeline benzemektedir. Bu modelde de programlama boyunca sürekli olarak gereksinimler alınıp kodlama yapılmakta ve test işlemleri yürütülmektedir. Böylece programda oluşacak eksiklikler ve hatalardan daha kolay kurtulma imkanına sahip olunmaktadır.



Evrimsel Geliştirme

Bir çok kaynakta bu model, döngüzel ve arttırımsal geliştirme modelinin birleşimi olarak tanımlanmaktadır.

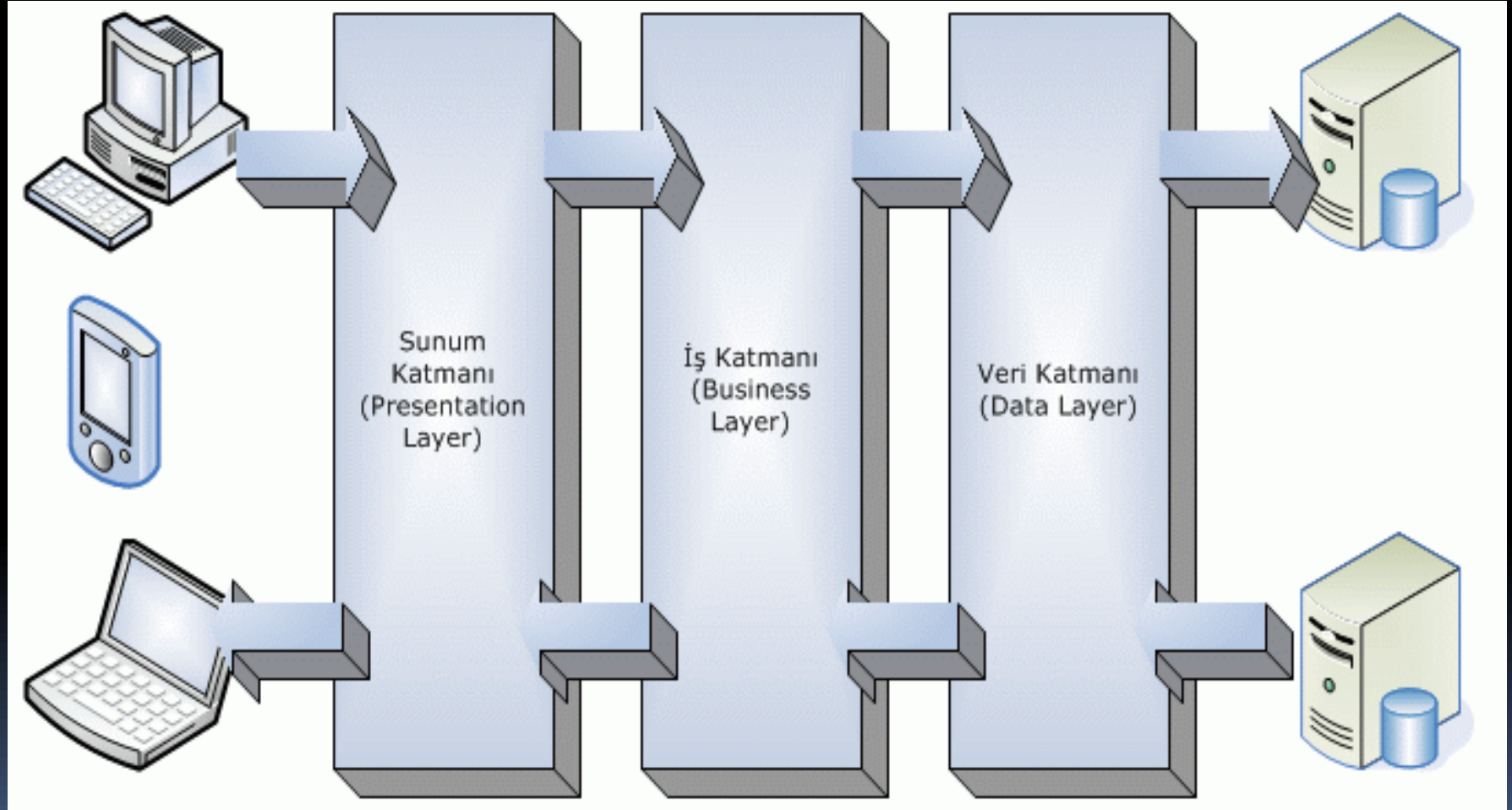


Yazılım Katmanları

Bir yazılım hazırlanırken, yazılımın kullanım amacına göre katmanlar oluşturulabilir veya tek katman üzerinde de çalıştırılabilir.

1. **Veri Katmanı** : Verinin veritabanı sisteminden getirilmesi ve veritabanı sistemine gelen verilerin eklenmesi için kullanılan katmandır.
2. **İş Katmanı** : Veri tabanından gelen verilerin işimize uygun olarak düzenlenmesi ve kontrol edilmesini sağlayan katmandır.
3. **Sunum Katmanı** : Kullanıcının göreceği, ve kullanıcıdan girdiği verilen alınacağı, yada daha önceden girilmiş verilerin bir şekilde kullanıcıya gösterilmesi için gereken bir katmandır.

Yazılım Katmanları



Tek Katmanlı Yazılımlar

Bu mimaride katmanlar tek bir bilgisayar ve yazılımda bulunmaktadır. Tek kullanıcılı bir sistem olduğu için hızlıdır fakat çok kullanıcıya destek sağlamadığı için kullanım oranı sınırlı olmaktadır.



İki Katmanlı Yazılımlar

İki katmanlı mimaride, kullanıcı arayüzü ve uygulama yazılımları bir katman, veriler ise ikinci katman olarak kullanılır. Kullanıcı arabirimi ve uygulama yazılımlarının olduğu bilgisayarlar istemci, verilerin olduğu bilgisayar ise



İki Katmanlı Yazılımlar

İki katmanlı mimaride, uygulama programları her bilgisayara ayrı ayrı yüklenmeli ve program değişikliği tekrardan tüm bilgisayarlara yeniden yüklenmelidir.

Veri tabanı tüm istemciler tarafından kullanıldığı için, kullanıcı ve iş yükü arttıkça kullanımdaki etkinlik azalabilmektedir.

Program ağdaki başka bir bilgisayara yüklenerek, yetki zaaflarına yol açabilir.

Üç Katmanlı Yazılımlar

Üç katmanlı mimaride, kullanıcı arayüzü, uygulama yazılımları, veriler ayrı bilgisayarlarda bulunmaktadır.

Bilgisayar

**Kullanıcı Ara
Birimi**

Bilgisayar

Uygulama Yazılımları

Bilgisayar

Veriler

Üç Katmanlı Yazılımlar

Üç katmanlı mimaride, web tabanlı uygulamalar ile gerçekleştirilmektedir. Bu uygulamaları kullanmak için bir adet web brovser olması yeterlidir. Uygulama yazılımlarındaki bir değişiklik aynı anda yazılımı kullanan tüm programlarda etkin olacaktır. Gereksiz yetkilendirmeler ortadan kalkacaktır. Veri tabanı ile bağlantı uygulama yazılımı tarafından yapılacağı için veri tabanı daha etkin kullanılmaktadır.

UML (Unified Modeling Language- Birleşik Modelleme Dili)

- UML, gelişen yazılım teknolojisi ve artan karmaşıklık karşısında endüstriyel olarak geliştirilmiş ve standartlaşmış bir evrensel modelleme biçimi ve dilidir.

Neden UML ihtiyaç duyarız?(1)

- Yazılım teknolojisi geliştikçe yazılan programların karmaşıklığı ve zorluğu giderek artmaktadır. Donanım ve yazılımın iç içe girdiği, büyük ağ sistemlerinin giderek arttığı bir dönemde doğaldır ki biz programcıların yazacağı programlarda büyüyecektir.

■ Neden UML ihtiyaç duyarız?(2)

- Yazacağımız programlar çok karmaşık olacağı için kod organizasyonu yapmamız zor olacaktır.
- Programımızın analiz ve dizayn aşamasında modellemeyi güzel yaparsak ileride doğabilecek birçok problemin çıkmasına engel olmuş oluruz.

Kullanımı

- UML daha çok nesneye dayalı programlama dilleri için uygundur. Problemlerimizi parçalara ayırabiliyorsak, ve parçalar arasında belirli ilişkiler sağlayabiliyorsak UML bizim için biçilmiş kaftan gibidir.
- UML 1997 yılında yazılımın, diyagram şeklinde ifade edilmesi için bir standartlar komitesi tarafından oluşturuldu. Daha önce hemen hemen her daldaki mühendislerin standart bir diyagram çizme aracı vardı. Ve şimdi de programcıların UML 'si var.

Avantajları(1)

- 1-) Öncelikle programımız kodlanmaya başlamadan önce geniş bir analizi ve tasarımı yapılmış olacağından kodlama işlemi daha kolay olur. Çünkü programdan ne beklediğimizi ve programlama ile neler yapacağımızı profesyonel bir şekilde belirleriz UML ile.
- 2-) Programımızda beklenmedik bir takım mantıksal hataları (bug) minimuma indirgemiş oluruz.
- 3-) Tasarım aşaması düzgün yapıldıysa tekrar kullanılabilen kodların sayısı artacaktır. Buda program geliştirme maliyetini büyük ölçüde düşürecektir.

Avantajları(2)

4-) UML diyagramları programımızın tamamını kapsayacağı için bellek kullanımını daha etkili hale getirebiliriz.

5-) Programımızın kararlılığı artacaktır. UML ile dökümanlandırılmış kodları düzenlemek daha az zaman alacaktır.

6-) Ortak çalışılan projelerde programcıların iletişimi daha kolay hale gelir.Çünkü UML ile programımızı parçalara ayırdık ve parçalar arasında bir ilişki kurduk.

DİYAGRAM TÜRLERİ(1)

CLASS DIAGRAM

Gerçek dünyada eşyaları nasıl araba, masa, bilgisayar şeklinde sınıflandırıyorsak yazılımda da birtakım benzer özelliklere ve fiillere sahip gruplar oluştururuz. Bunlara "Class"(sınıf) denir.

OBJECT DIAGRAM

Bir nesne(object) sınıfın (class) bir örneğidir. Bu tür diyagramlarda sınıfın yerine gerçek nesneler kullanılır.

DIYAGRAM TÜRLERİ(2)

STATE DIYAGRAM

Gerçek nesnelerin herhangi bir zaman içindeki durumunu gösteren diyagramlardır. Mesela, Can nesnesi insan sınıfının gerçek bir örneği olsun. Can 'nın doğması, büyümesi, gençliği ve ölmesi State Diagram 'larıyla gösterilir.

SEQUENCE DIAGRAM

Class ve Object diyagramları statik bilgiyi modeller. Halbuki gerçek zamanlı sistemlerde zaman içinde değişen inter aktiviteler bu diyagramlarla gösterilemez. Bu tür zamanla değişen durumları belirtmek için sequence diyagramları kullanılır.

DİYAGRAM TÜRLERİ(3)

ACTIVITY DIAGRAM

Bir nesnesinin durumu zamanla kullanıcı tarafından ya da nesnenin kendi içsel işlevleri tarafından değişebilir. Bu değişim sırasını activity diyagramlarıyla gösteririz.

USE CASE DIAGRAM

Programımızın davranışının bir kullanıcı gözüyle incelenmesi Use Case diyagramlarıyla yapılır. Gerçek dünyada insanların kullanacağı bir sistemde bu diyagramlar büyük önem taşırlar.

COLLABORATION DIAGRAM

Bir sistemin amacının yerine gelmesi için sistemin bütün parçaları işlerini yerine getirmesi gerekir. Bu işler genellikle birkaç parçanın beraber çalışmasıyla mümkün olabilir. Bu tür ilişkileri göstermek için Collaboration Diyagramları gösterilir.

DİYAGRAM TÜRLERİ(4)

COMPONENT DIAGRAM

Özellikle birden çok geliştiricinin yürüttüğü projelerde sistemi component dediğimiz parçalara ayırmak, geliştirmeyi kolaylaştırır. Sistemi öyle modellememiz gerekir ki her geliştirici ötekinden bağımsız olarak çalışabilsin. Bu tür modellemeler Component Diyagramlarıyla yapılır.

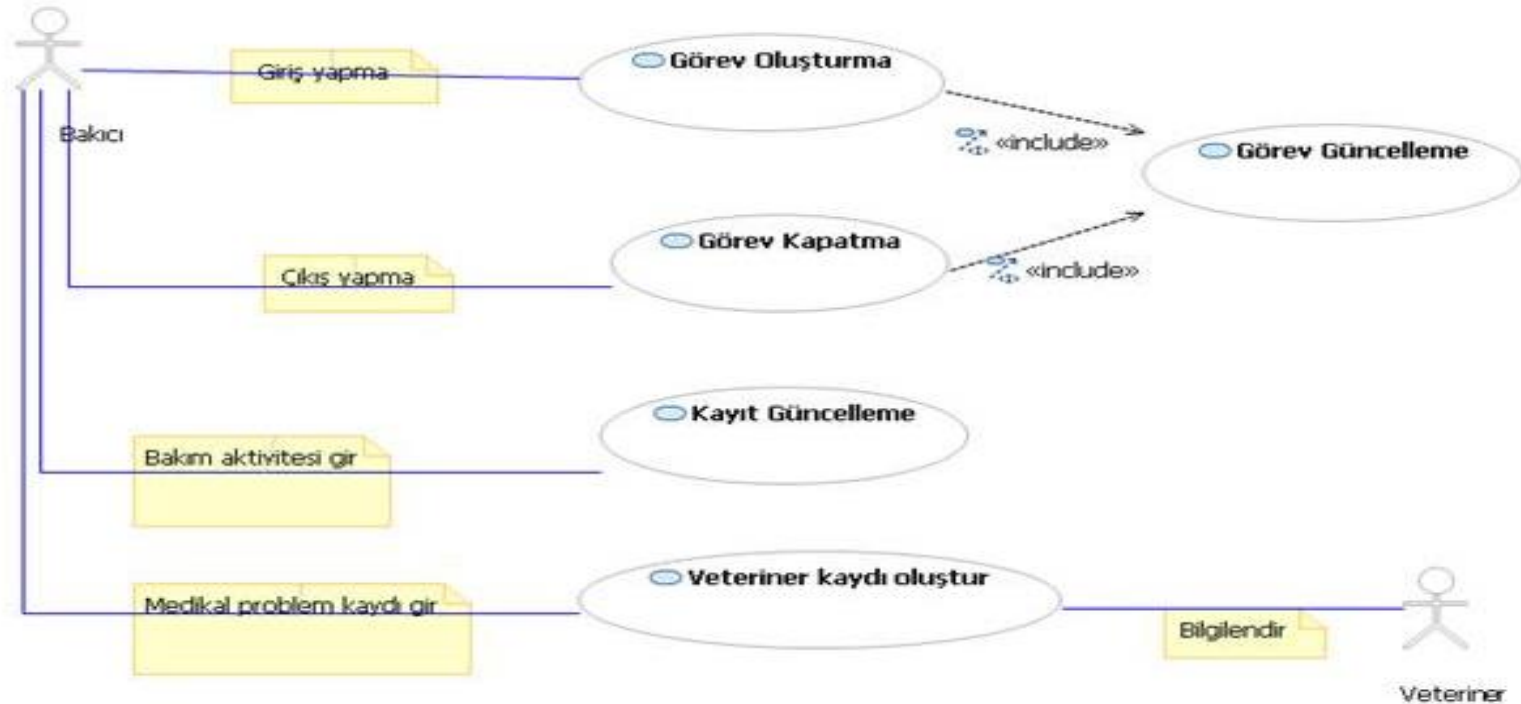
DEPLOYMENT DIAGRAM

Bu tür diyagramlarla sistemin fiziksel incelenmesi yapılır. Mesela bilgisayarlar arasındaki bağlantılar, programın kurulacağı makinalar ve sistemimizdeki bütün aletler Deployment Diyagramında gösterilir.

örnek1

Use Case Diyagramı

Sistemin fonksiyonel gereksinimlerini resmetmek için çok kullanışlıdır. Aşağıdaki grafik KYS'nin basitçe bir usecase diyagramını göstermektedir.

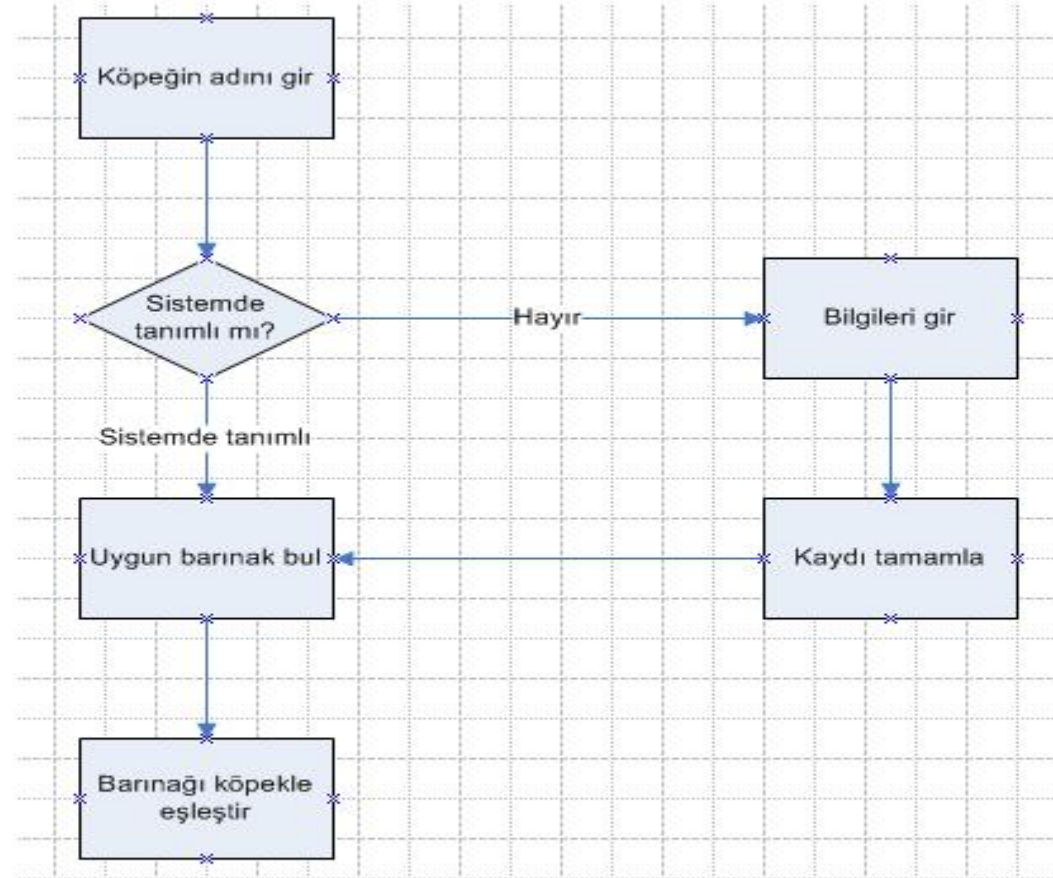


- Bakıcı bir giriş kaydı yaptığında KMS nasıl davranır?
 - o Sistem bir görev oluşturur.
- Bakıcı bir çıkış kaydı yaptığında KMS nasıl davranır?
 - o Sistem bir görev kapatır
- Bakıcının hangi aktiviteleri KMS'nin görev güncelleme işlemi yapmasına sebep olur?
 - o Giriş ve çıkış işlemleri
- Bakıcının hangi aktivitesi Veterineri ilgilendirir?
 - o Bir medikal kayıt girme işlemi

örnek2

Aktivite Diyagramı

Aktivite diyagramları bir fonksiyonel gereksinimin bir ana senaryo ve birkaç alternatif senaryoyu içeren detaylı davranışını resmeder. Bu, verilen bir fonksiyonalityi tam anlamıyla anlamınıza yardımcı olur. Aşağıda KYS'nin bir fonksiyonel gereksinimi için oluşturulmuş bir aktivite diyagramı görüntülenmektedir.

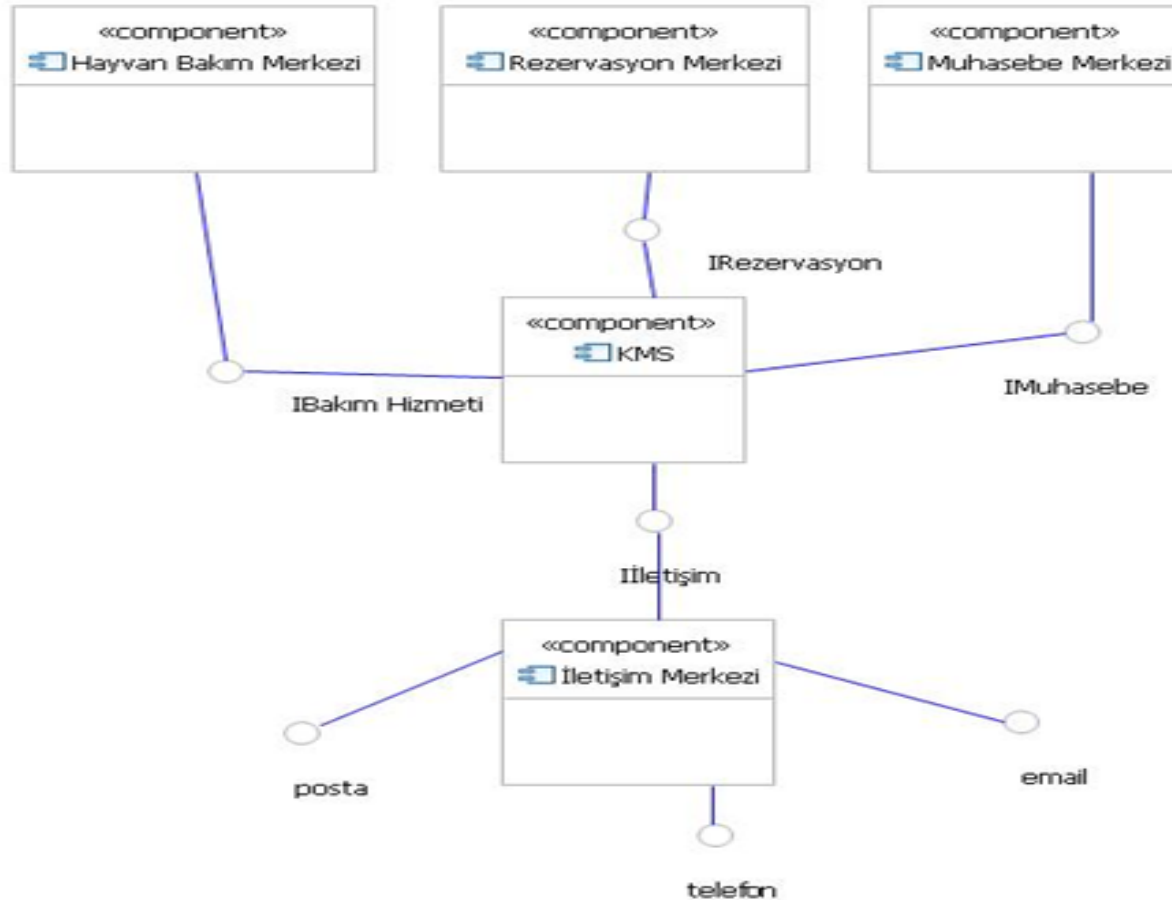


- Sistemde kayıtlı bir köpeğe barınak bulmak için sistemin izleyeceği adımlar nelerdir?
 - Köpeğin adını gir, uygun barınağı bul ve barınağı köpeğe assign et.
- Yeni bir köpek için sistemde ek olarak yapılması gerekenler nelerdir?
 - Köpeğin kişisel bilgilerini gir ve kayıt oluştur.

örnek3

Komponent Diyagramı

Komponent diyagramları, sistemin çalıştırılabilir bölümleri, komponentleri, veri depoları gibi deploy edilebilir birimlerini ve etkileşimde bulundukları interface'leri resmeder. Sistemin mimarisini göstermesi açısından faydalıdır.

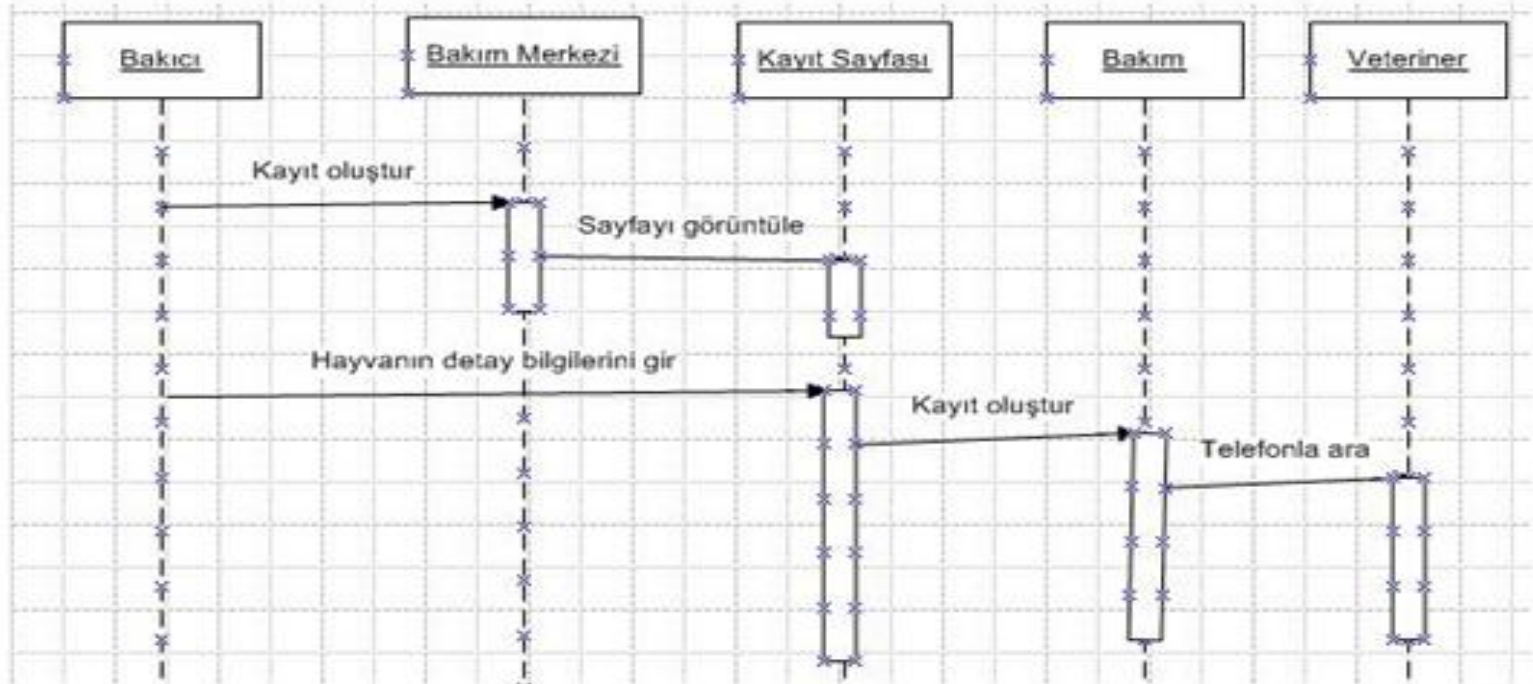


- Hayvan bakım merkezi, bakımının hayvanın bilgilerini girdiği bir web sayfasıdır. Peki KMS'ye veri sağlamak için hangi arayüz kullanılır?
 - IBakım Merkezi
- Diğer hangi komponentler hangi arayüzlerle KMS'yi besler?
 - Rezervasyon merkezi IRezervasyon arayüzüyle, muhasebe merkezi IMuhasebe arayüzüyle.
- İletişim merkezi komponentinden hangi tür iletişimler gerçekleştirilir?
 - Posta, telefon ve email.

örnek4

Sequence Diyagramları

Sequence diyagramları bir fonksiyonel gereksinimin bir senaryosu veya path'i boyunca, zaman içerisinde gerçekleştirdiği detaylı davranışları gösterir.

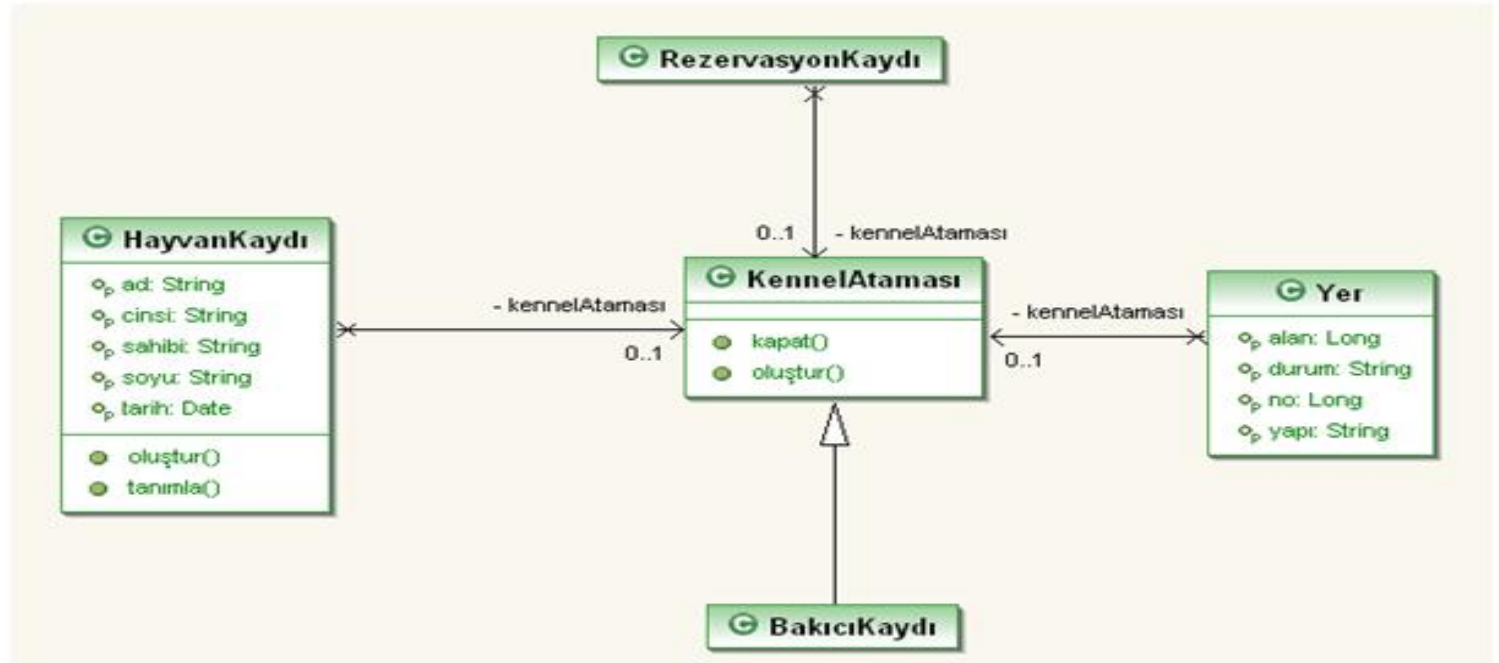


- Sistemdeki hangi nesneler yeni kayıt oluşturmaya ilgilidir?
 - Bakım merkezi, kayıt sayfası, IBakım merkezi ve iletişim arayüzleri
- Hangi aktörler yeni kayıt oluşturmaya ilgilidir?
 - Bakıcı ve veteriner
- Kayıt sayfası yeni bir kayıt için hangi arayüzü kullanır?
 - IBakım hizmeti
- Sistem veterinerle nasıl iletişim kurmaktadır?
 - Telefonla

örnek5

Sınıf Diyagramları

Sistem tasarımında, sınıfların ve interface'lerin birbirleri arasındaki ilişkiyi resmeder. Kodun nesne yönelimli yapısını tespit etmek açısından faydalıdır.

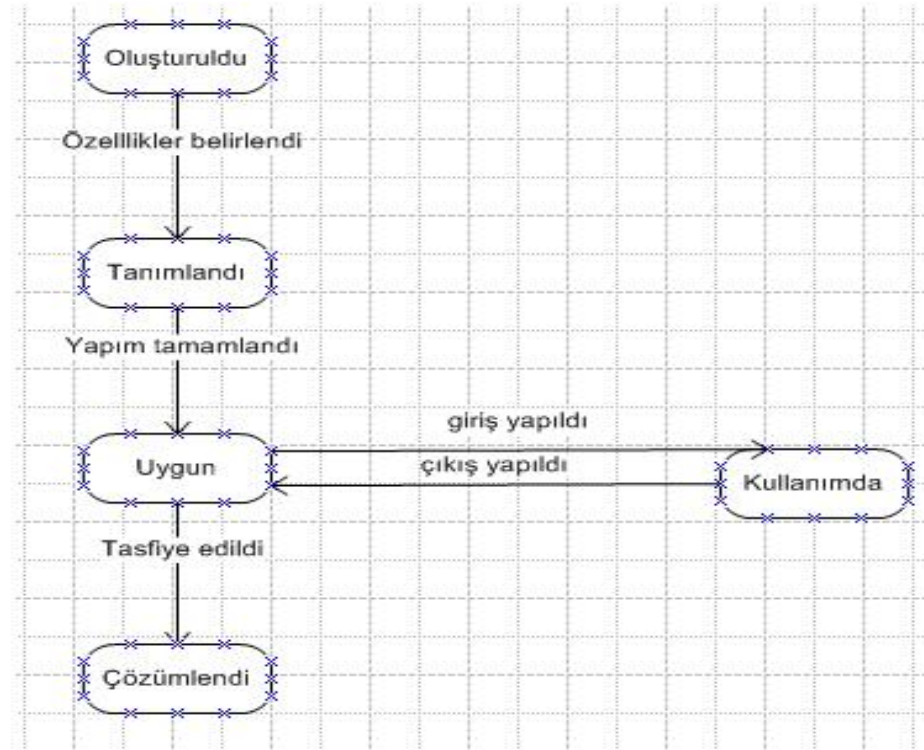


- Hangi sınıflar KennelAtaması sınıfıyla ilişkilidir?
 - HayvanKaydı, RezervasyonKaydı ve Yer
- KennelAtaması sınıfı hangi işlemleri yapabilir?
 - Oluşturma ve kapatma
- Hangi özellikler bir HayvanKaydı sınıfını tanımlar?
 - Ad, cins, sahip, soy ve tarih
- Hangi özellikler bir Yer sınıfını tanımlar?
 - Alan, durum, no ve yapı.

örnek6

Statechart Diyagramları

Dahili veya harici olayların tepkisine göre sistemin durum değişikliğini gösteren diyagramlardır.

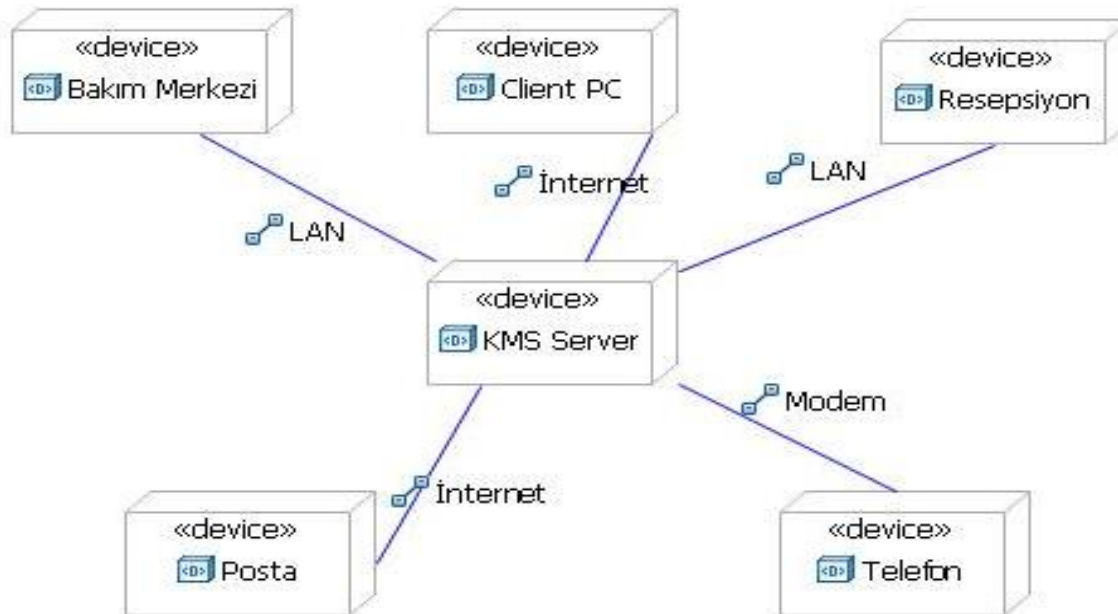


- Hangi işlemle tanımlandı statüsüne geçilir?
 - Özelliklerin belirlenmesi
- Hangi durumdan gelen hangi işlemle bir barınak uygun hale gelir?
 - Tanımlandı durumundan sonraki 'yapım tamamlandı' olayıyla, kullanımda durumundan sonraki 'çıkış yapıldı' olayıyla
- Barınak uygun durumdayken 'tasfiye edildi' olayı gerçekleştiğinde hangi duruma geçilir?
 - Çözümlendi
- Barınak Kullanımda durumundan Çözümlendi durumuna nasıl geçer?
 - Çıkış yapıldı olayı ve bunu takip eden 'tasfiye edildi' olayıyla.

örnek7

Deployment Diyagramları

Sistemin deploy edilebilir parçalarının bağlandıkları düğümler ile bu düğümlerin birbirleri ve farklı cihazlarla nasıl iletişimde bulunduklarını resmeder.

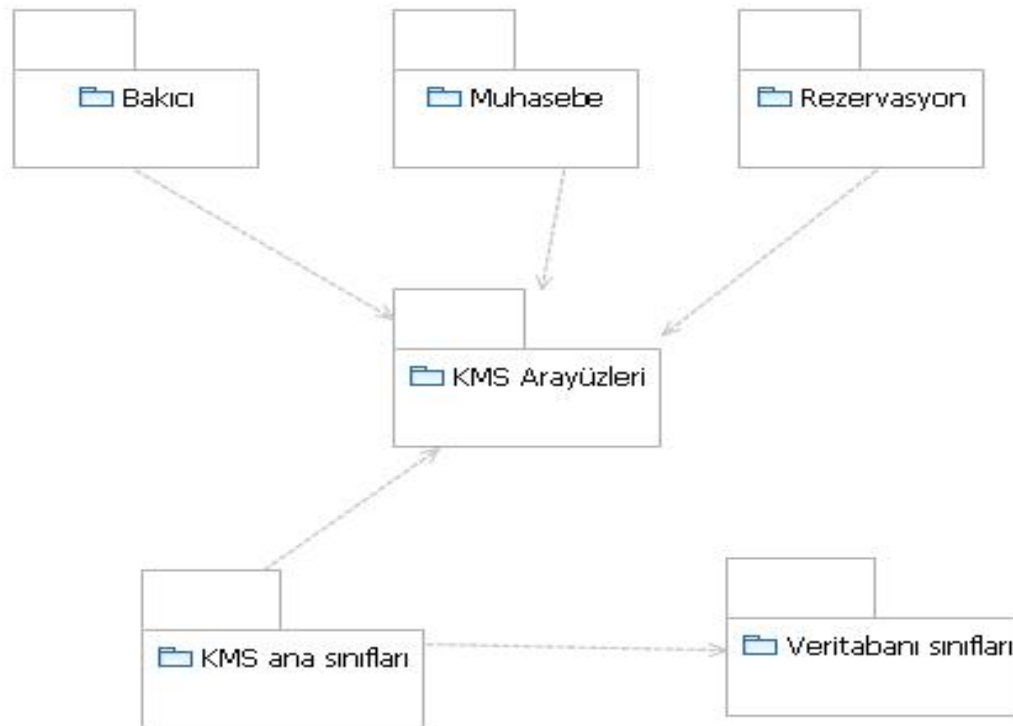


- KMS Server'ı telefon ile nasıl haberleşir?
 - Modem vasıtasıyla
- Kullanıcının PC'si KMS Server'ındaki bilgiye nasıl erişir?
 - İnternet yoluyla
- Veri, Bakım merkezinden resepsiyona nasıl gidebilir?
 - Diyagrama göre KMS Server'ı üzerinden.

örnek8

Paket Diyagramları

Paket diyagramları, tasarımınızdaki ilişkili elementlerin nasıl bir araya gelip gruplandığını ve bu grupların birbirine nasıl depend ettiklerini gösterir. Kompleks tasarımları parçalara ayırarak, yönetilebilir küçük parçalar haline getirilmesi açısından faydalıdır.



- Hangi paketler KMS arayüzleri sınıfının verilerini kullanır?
 - Bakıcı, muhasebe, rezervasyon ve kms ana sınıfları
- KMS arayüzleri sınıfı hangi sınıfların verisini kullanır?
 - Kms ana sınıfları ve veritabanı sınıfları