Nesne Yönelimli Analiz ve Tasarım

Sistem Tasarımı

Tasarım Hedeflerinin Belirlenmesi
Sistemin alt bileşenlerine ayrıştırılması
Yazılım mimarileri

Nesne Yönelimli Tasarım- Sistem Tasarımı

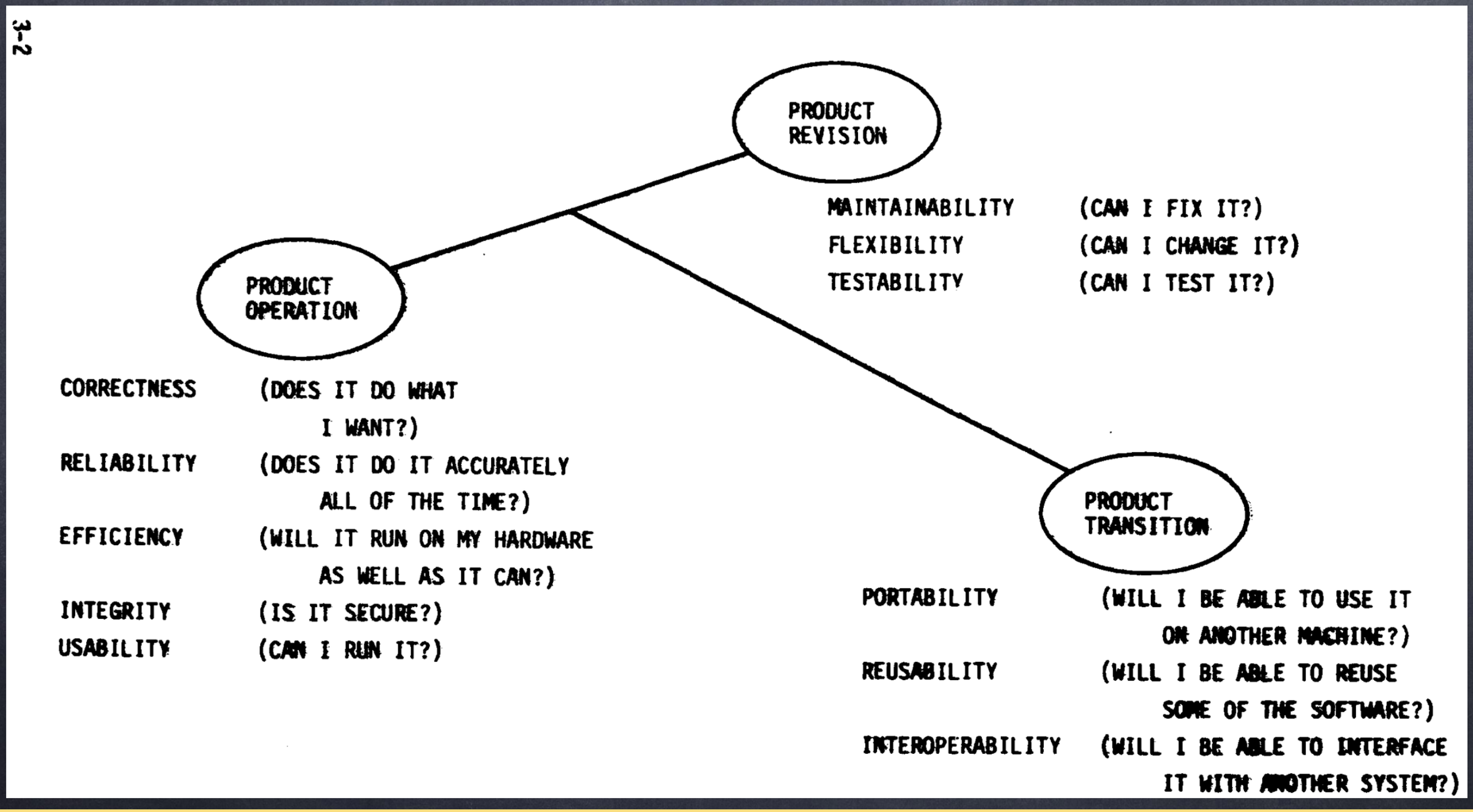
* Analiz aşaması uygulama alanına (application domain) odaklanır.
* Uygulama alanı gerçekleştirilecek yazılımın çalışacağı ortamı ifade eder.
* Tasarım aşaması Çözüm alanına (solution domain) odaklanır.
* Tasarım aşaması, sistem tasarımı ve nesne tasarımı olarak iki bölüme ayrılabilir.
* Sistem tasarımı kapsamında sistem alt bileşenlerine ayrılır. Sistemle ilgili tasarım hedefleri belirlenir ve kısıtlar ihlal edilmeden bu hedeflere ulaşmak için gerekli mimari oluşturulur.

Nesne Yönelimli Tasarım- Sistem Tasarımı

* Analiz aşaması:
  * gereksinimler, kısıtlar belirlenir
  * kullanım durumları tanılanır
  * analiz sınıf şeması/ nesne modeli, dinamik modeller (sıralama şeması, etkinlik şeması vb.) oluşturulur
* Sistem tasarım aşaması:
  * tasarım hedefleri (kalite ölçütleri) belirlenir
  * sistemin alt bileşenlerine ayrıştırılır
  * tasarım hedeflerine ulaşmayı sağlayacak yazılım mimarisi (mimari stiller), veri modelleri, erişim denetimi yöntemleri vb. belirlenir.

1. Tasarım Hedeflerinin Belirlenmesi / Yazılımların Kalitesi (McCall Kalite Üçgeni)

Bir yazılımın kalitesine etki eden faktörler üç sınıfa ayrılır.



McCall, J.A., Richards, P.K. and Walters, G.F. (1977) Factors in Software Quality. RADC TR-77-369, Rome Air Development Center, Rome.

Kalite Ölçütlerinin Birbirlerine Olumsuz Etkileri

* Integrity <-> Efficiency
  * Güvenlik denetimi ek kontrol ve fazladan yer anlamına gelir
  * bankacılık uygulaması
* Portability <-> Efficiency
  * tüm platformları desteklemesi için eklenen denetimler fazladan kaynak kullanımı ve gecikme anlamına gelir
* Reusability (Flexibility, Maintainability) <-> Efficiency
  * program daha küçük parçalar halinde yazılırsa hız azalır.
  * gömülü sistemler, gerçek zamanlı sistemler...
* Diğerleri neler olabilir?

2. Sistemin alt bileşenlerine ayrıştırılması

* Karmaşıklığı azaltmak için yazılım sistemi alt bileşenlerine (alt sistem) ayrılır.
* Alt sistemler, birbirleriyle yakından ilişkili; sınıflar, ilişkiler ve kısıtlardan oluşur.
* Alt sistemler sayesinde;
  * problem çözümü kolaylaşır
  * çok sayıda kişinin aynı projede aynı anda çalışabilmesi sağlanır
  * sistemin anlaşılması, anlatılması kolaylaşır
  * bakım kolaylaşır
  * modülerlik, kod tekrar kullanımı artar...
* Analiz aşamasındaki alt bileşenler
  * sınıf
  * paket (UML paket şeması)
* Tasarım aşamasındaki alt bileşenler için "UML Component" şeması kullanılabilir.

2. Sistemin alt bileşenlerine ayrıştırılması

İyi bir tasarım için

High Coherence

Modüller tek ve özel bir işi, mükemmel bir şekilde yapmalı. Alt sistemler içerisindeki sınıflar benzer işi yapmalı ve ilgili olmalı. "Cohesion" Bunun ölçüsüdür .

Low Coupling

Aradaki bağıntılar ne kadar fazla olursa nesne içerisinde yapılacak köklü değişiklik ona bağlı olan modülleride etkileyecektir.

2. Sistemin alt bileşenlerine ayrıştırılması- UML Paket Diagramı

* Yapısal gösterim şekillerindendir.
* Paketleri ve paketler arası ilişkileri göstermek için kullanılır.
* Paketler içerisinde; sınıf, paket, bileşen (component), kısıtlar ve bağımlılıklar olabilir.
* Java platformunda; package ifadesi paket oluşturmak için, import ifadesi ise kullanılacak paketleri içe aktarmak için kullanılır.

* import java.net.Socket;
* import java.io.PrintWriter;
* MusteriliSiparis.java
  * package cc.ders7.lab;
  * import cc.ders5.Musteri;
  * import cc.ders6.siparis.Siparis;

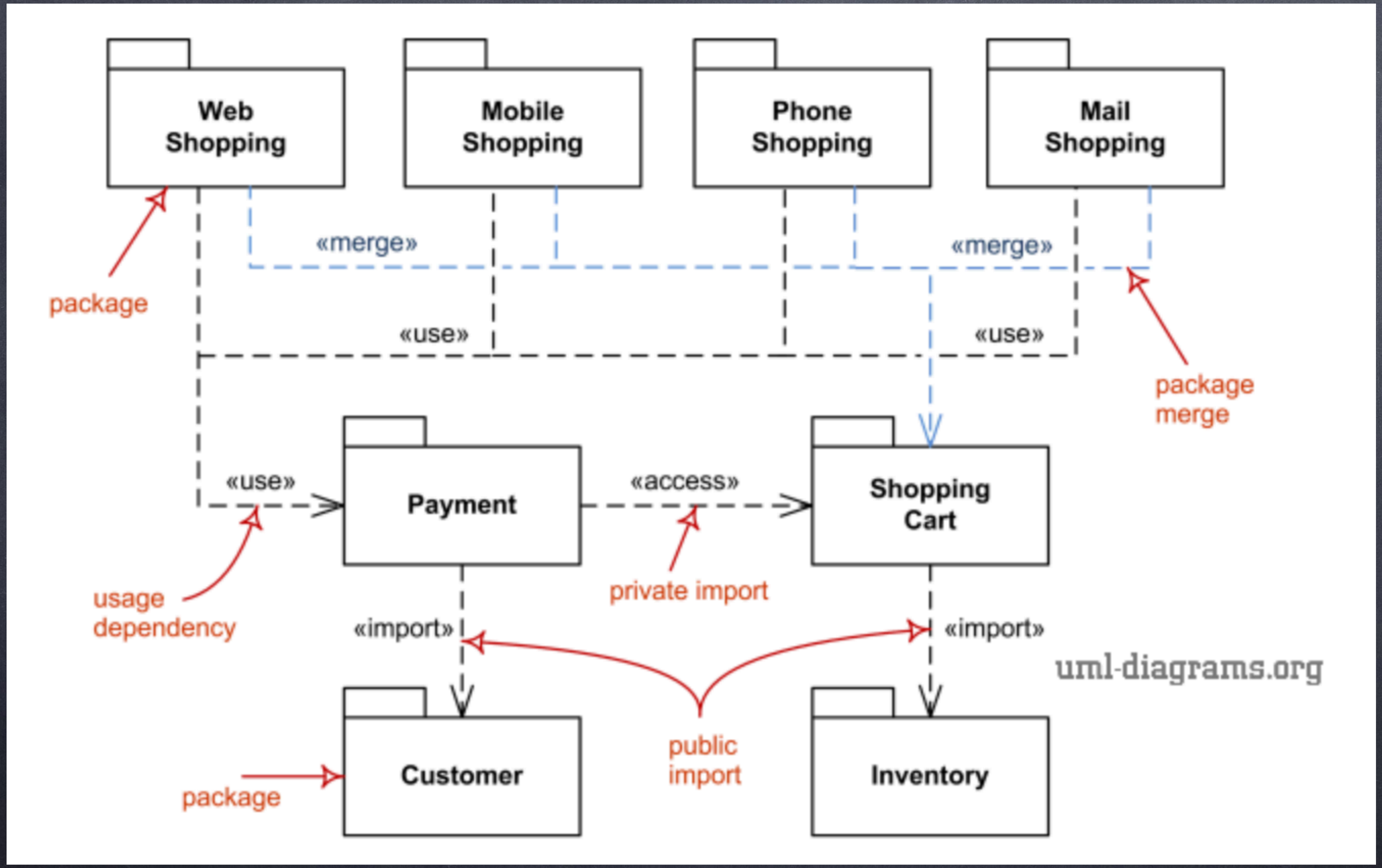
```java
package cc.ders5;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Uygulama {

    public static void main(String [] args){
```

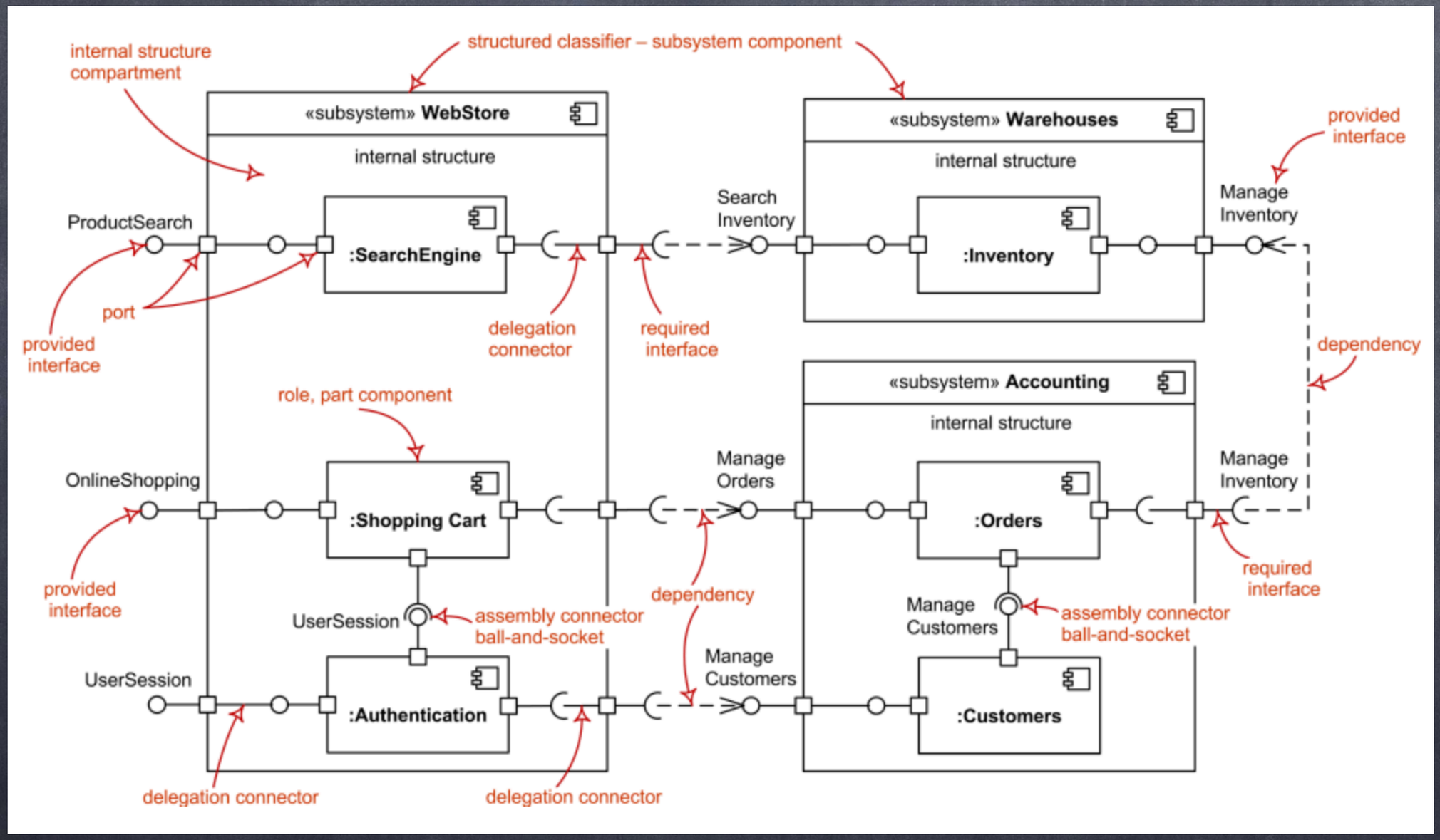
2. Sistemin alt bileşenlerine ayrıştırılması- UML Paket Diagramı



https://www.uml-diagrams.org/package-diagrams-overview.html

2. Sistemin alt bileşenlerine ayrıştırılması- UML "Component" Diagramı

* Yapısal gösterim şekillerindendir. "Component" (Bileşen), sistemin diğer bölümleriyle haberleşmeyi sağlayan arayüzlere (API-Application Programming Interface) sahip alt sistemler ya da sistemlerdir. Ortak amaca hizmet eden sınıfların bir araya getirilmesiyle oluşur.



* Şekilde bağıntılı üç alt sistem (WebStore(Web mağazası), Warehouse(depo, ambar), Accounting(Muhasebe)) bulunmaktadır. Alt sistemler içerisinde başka alt sistemlerde bulunmaktadır.
* "SearchEngine" sağladığı "ProductSearch" arayüzüyle ürünlerin aranmasını sağlar. Bu işlem için "Inventory" bileşeninin sağladığı "SearchInventory" arayüzünü kullanır.
* "Shopping Cart" bileşeni çevrimiçi alış veriş arayüzü sağlar. Alış veriş için kullanıcının oturum açması gerekir. Siparişlerin yönetimi için "Orders" bileşeninin "Manage Orders" arayüzünü kullanır.
* ...

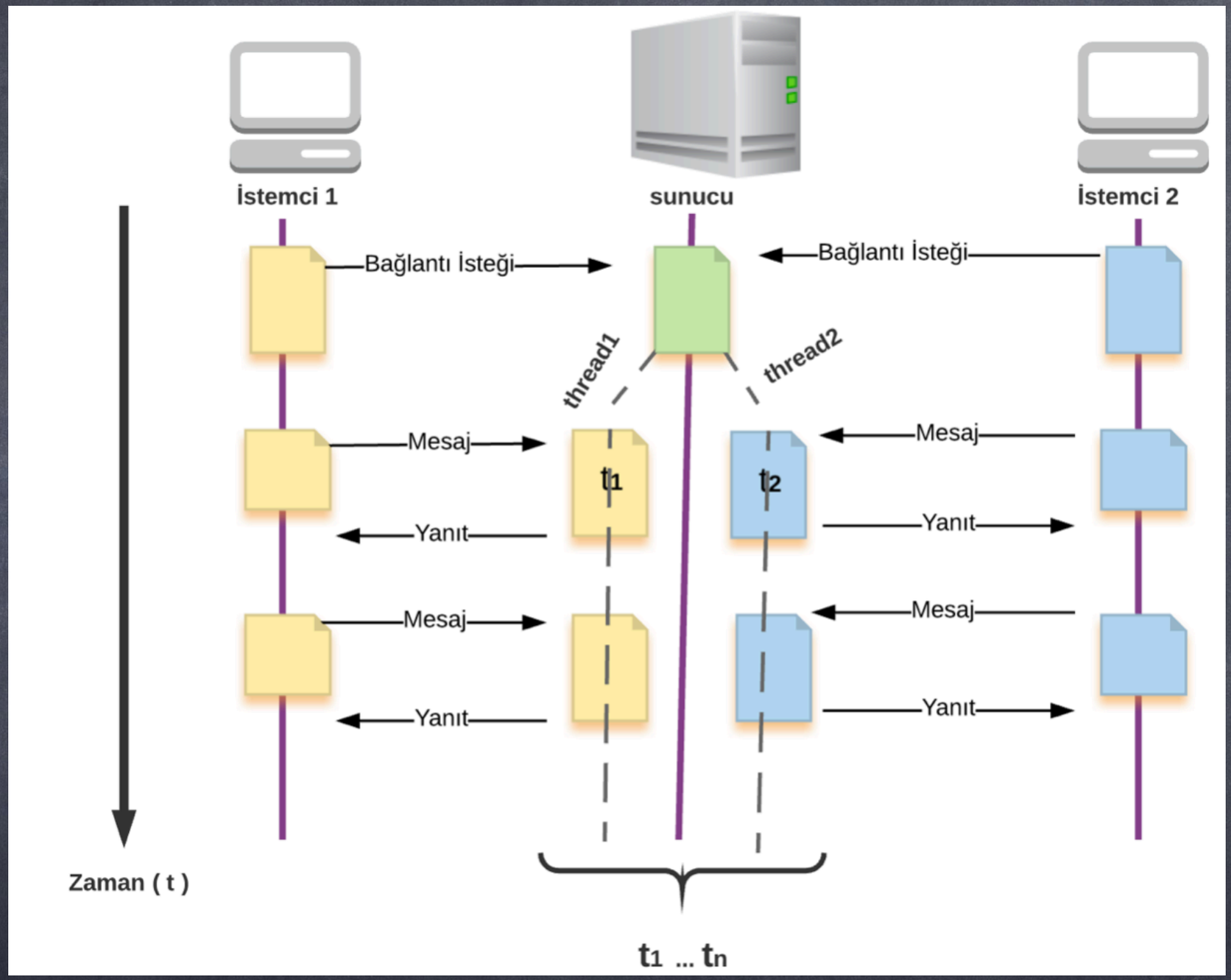
https://www.uml-diagrams.org/component-diagrams.html

3. Yazılım mimarileri (mimari stiller)

* İstemci/Sunucu (Client/Server)
* Üç Katmanlı Web Mimarisi (Three-tier Web Architecture)
* Servis Yönelimli Mimari (Service-Oriented Architecture (SOA))
* Model/Görünüm/Denetleyici (Model/View/Controller(MVC))
* Peer-To-Peer

https://www.uml-diagrams.org/package-diagrams-overview.html

3. Yazılım mimarileri (İstemci/Sunucu)

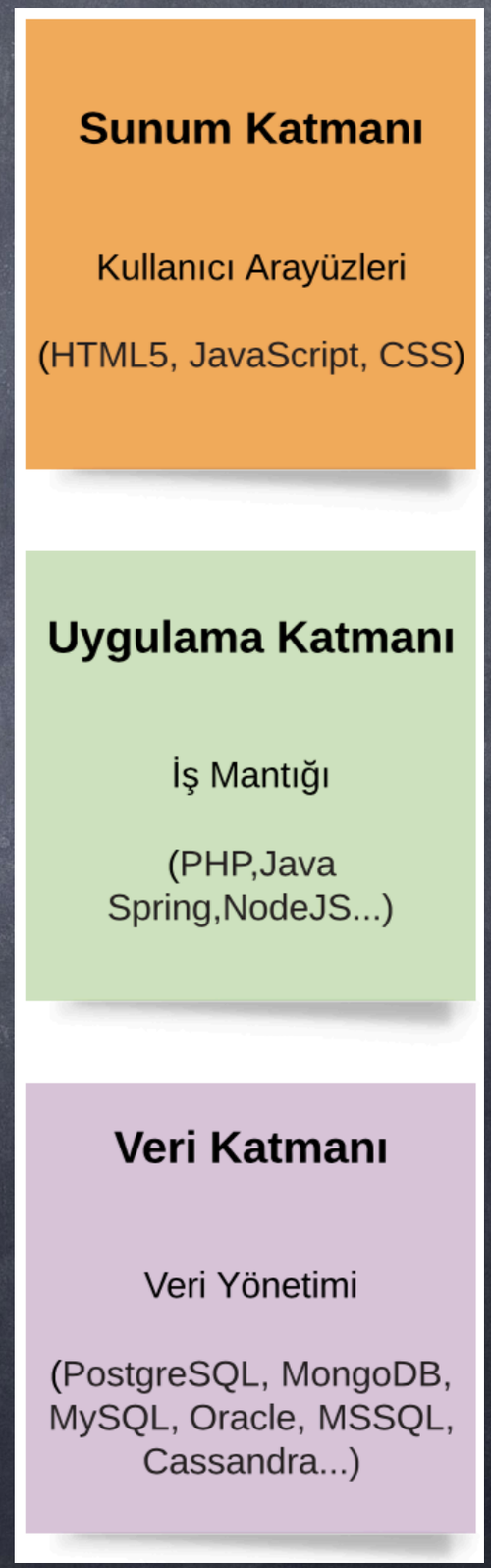


İstemci/Sunucu Örneği

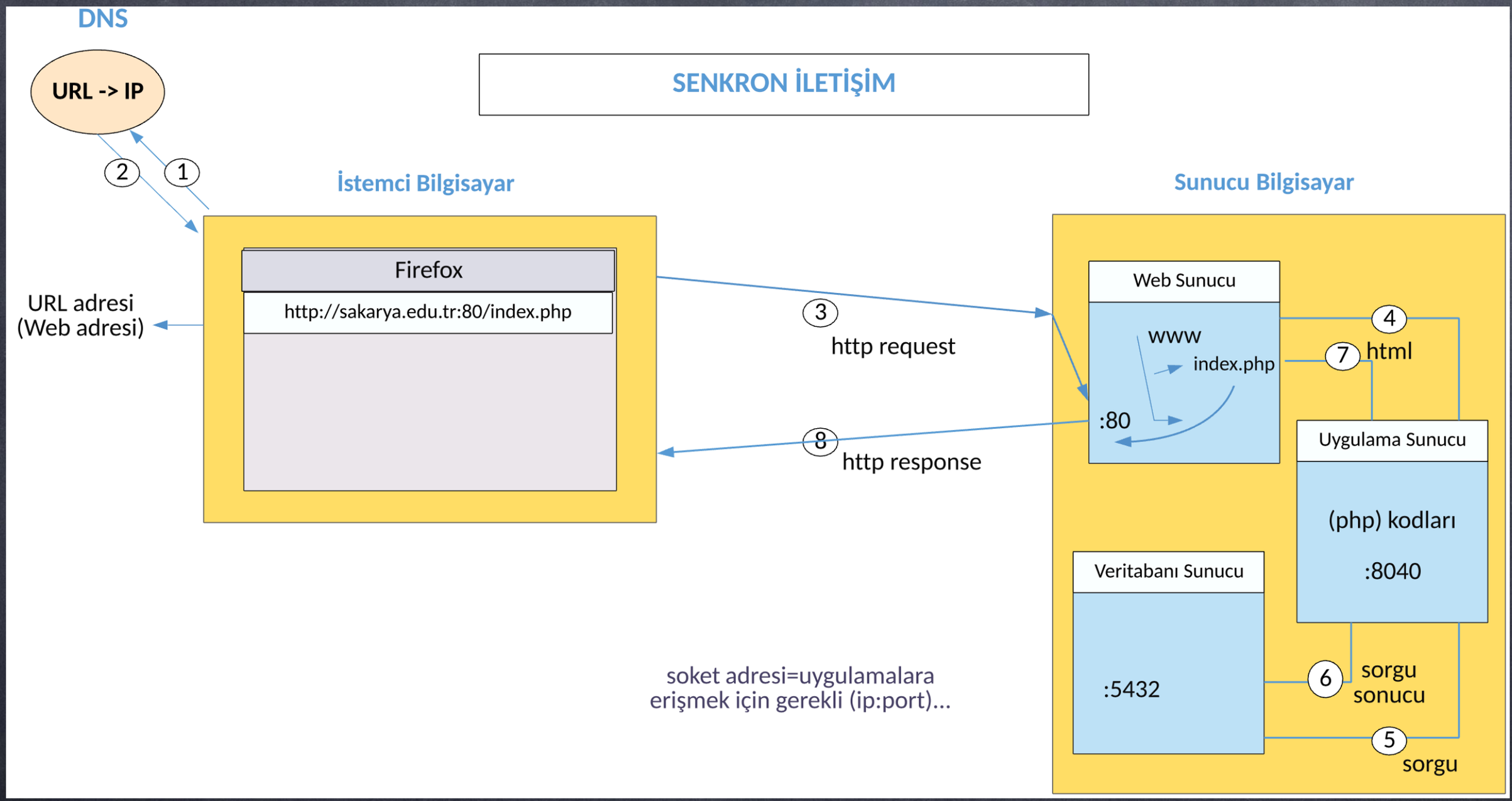
https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders8/IstemciSunucuMimarisi

3. Yazılım mimarileri (Üç Katmanlı Web Mimarisi (Three-tier Web Architecture))

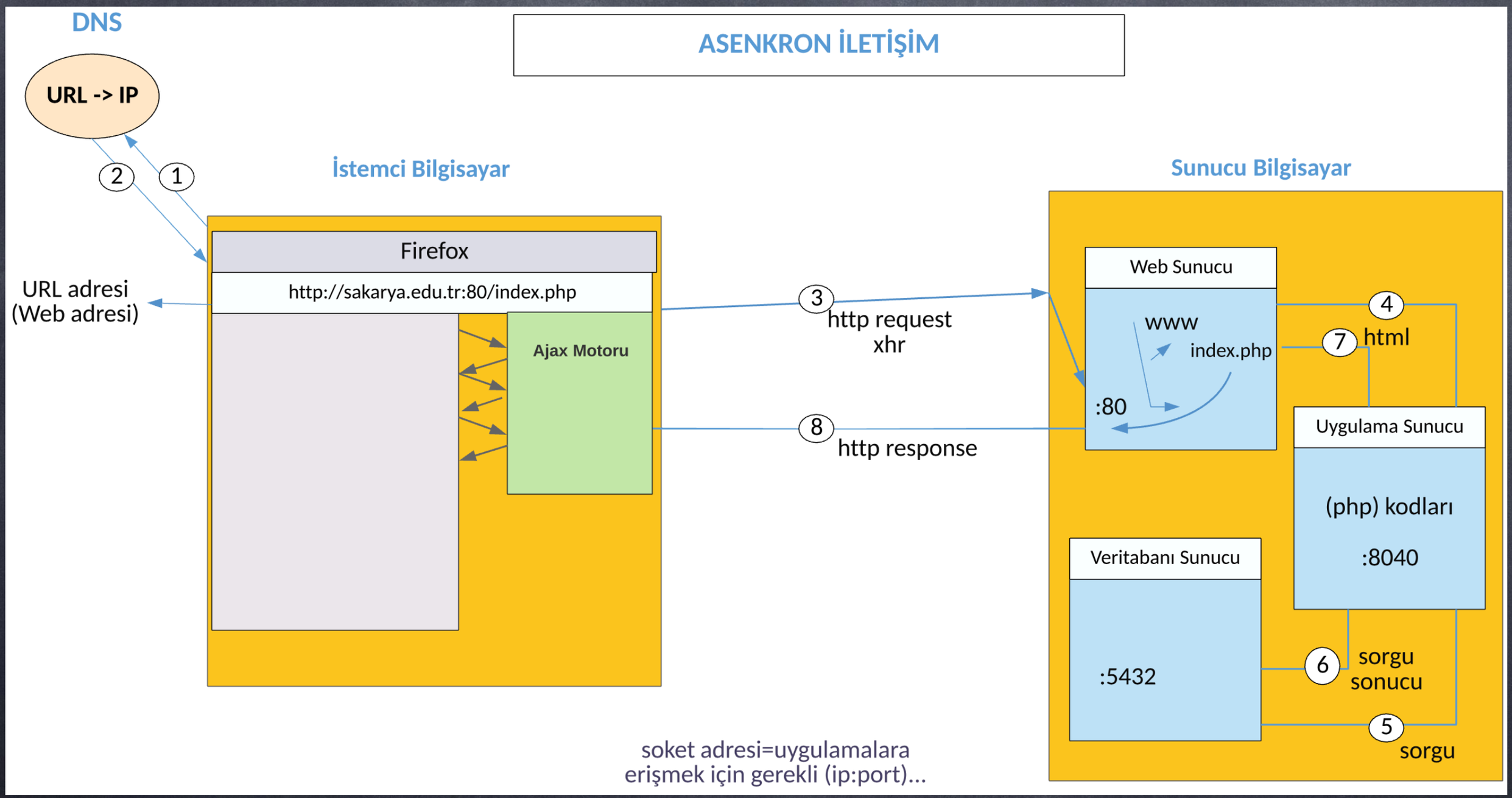
* Sunum, iş mantığı ve veri yönetimi fonksiyonlarının fiziksel olarak ayrıldığı İstemci/Sunucu mimarisidir.
* Modülerlik, başarım, bölümler birbirlerinden bağımsız geliştirilebilir



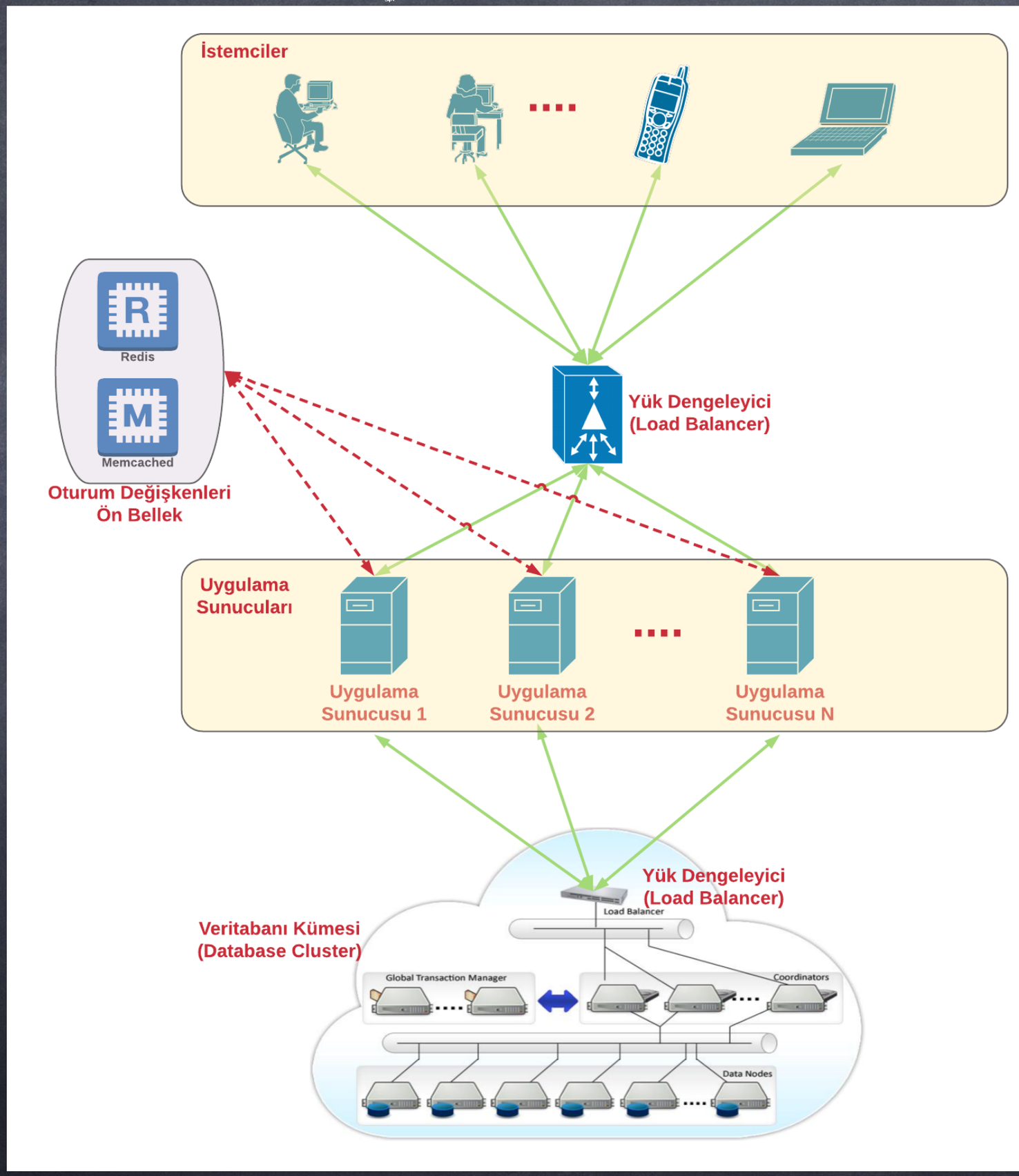
3. Yazılım mimarileri (İstemci/Sunucu(Web Uygulama Mimarisi))



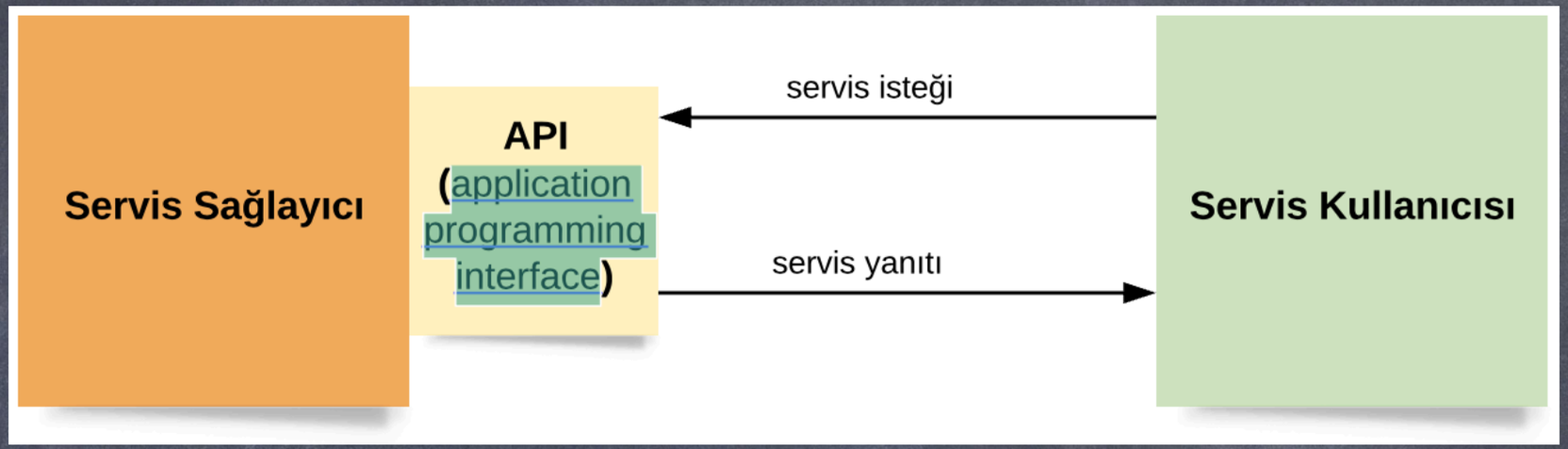
3. Yazılım mimarileri (İstemci/Sunucu(Web Uygulama Mimarisi))



3. Yazılım mimarileri (İstemci/Sunucu(Ölçeklenebilir Web Uygulama Mimarisi))



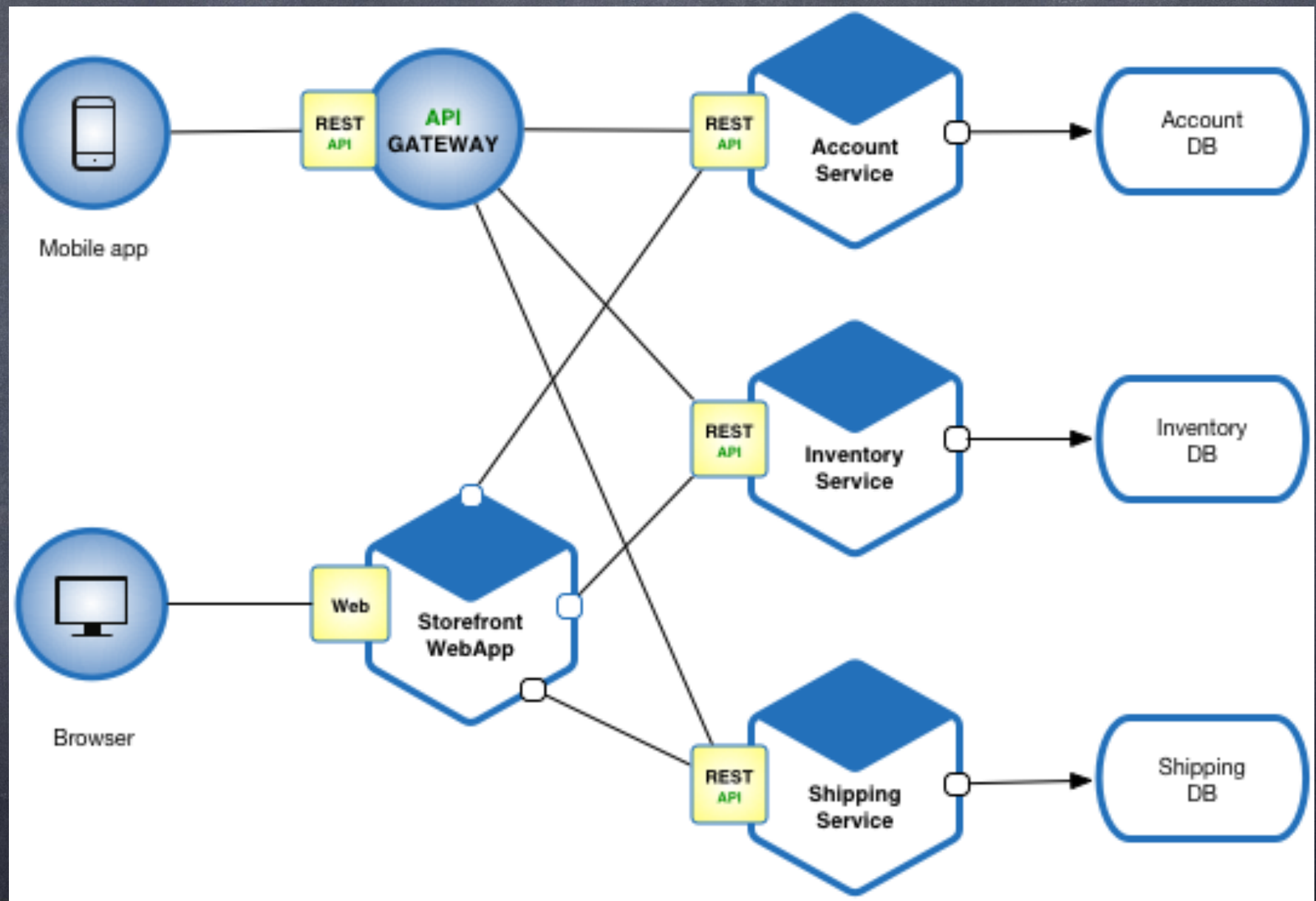
Servis Yönelimli Yazılım Mimarisi (Service-Oriented Architecture (SOA))



* Sistemlerin birlikte çalışmasını sağlar
* Servisler birbirlerinden bağımsızdır.
* RESTful, gRPC, SOAP, XML-RPC, Jini, COBRA...
* RESTful ile json desteği

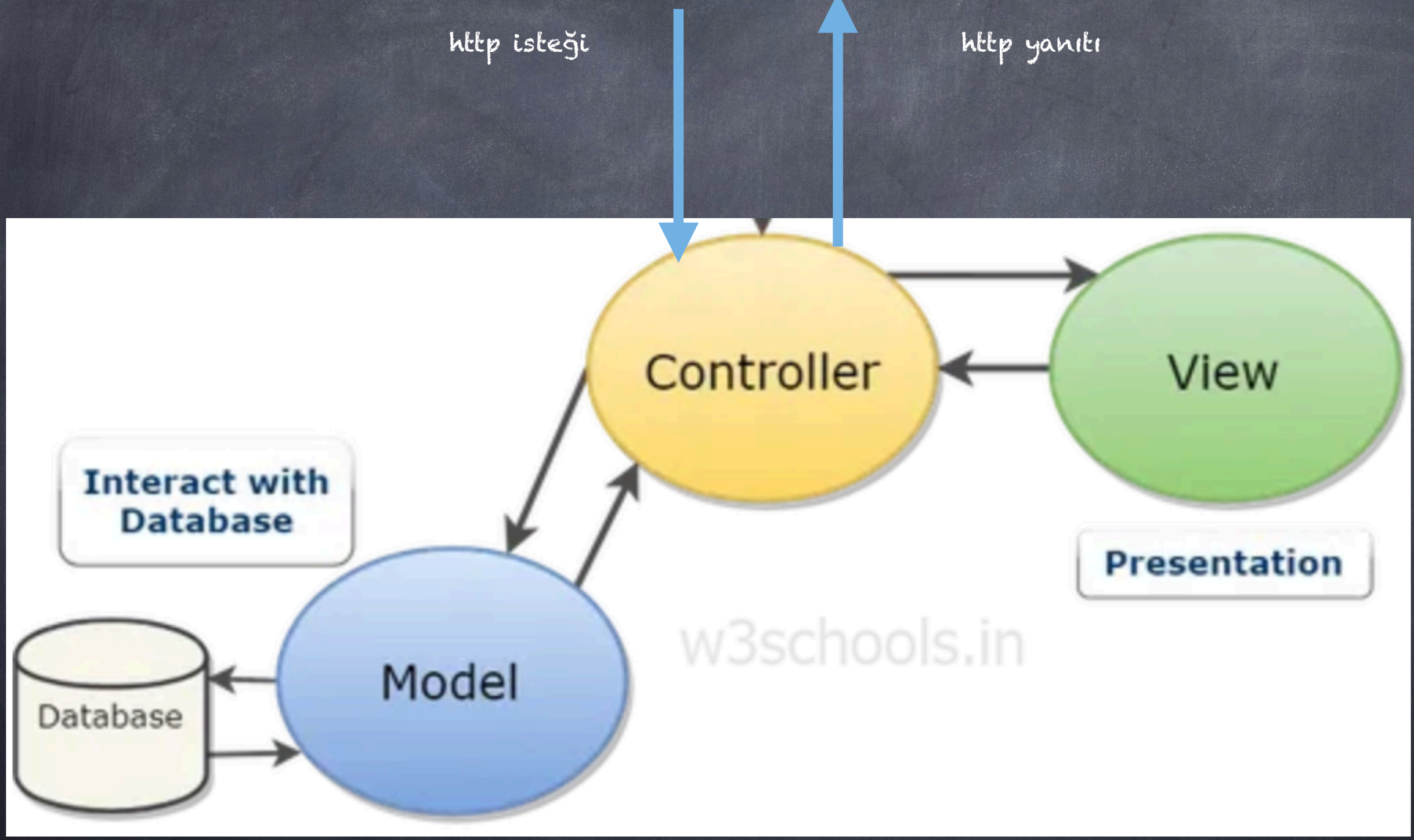
Mikro Servis Mimarisi

* servis yönelimli mimarinin modern bir türevidir.
* uygulamalar çok sayıda servisin bir araya getirilmesiyle oluşturulur.
* servislerin; bakımı ve testi kolaydır, bağımsız olarak konuşlandırılırlar/ geliştirilirler.



https://microservices.io/

3. Yazılım mimarileri (MVC)



https://www.w3schools.in/mvc-architecture