Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ 313 Yazılım Mühendisliği

12. Sistem Mimarisi

Tasarım

Yazılım Geliştirmede Tasarım

Belirli bir dizi gereksinim için, yazılım geliştirme ekibi bu gereksinimleri karşılayacak bir sistem **tasarlamalıdır**.

Bu tasarımın birçok yönü bulunmaktadır:

- sistem mimarisi
- program tasarımı
- güvenlik
- performans

Tasarım kullanılabilirlik ile ilgili derslerinin önemli bir parçasıydı.

Uygulamada, gereksinimler ve tasarım birbiriyle ilişkilidir. Özellikle, tasarım üzerinde çalışmak genellikle gereksinimleri netleştirir. Bu geri bildirim, yinelemeli ve çevik yazılım geliştirme yöntemlerinin bir gücüdür.

Yaratıcılık ve Tasarım

Yaratıcılık ve tasarım

Sistem ve program tasarımı, kullanıcı arayüzleri gibi yazılım geliştirmenin özellikle yaratıcı bir parçasıdır.

Her şeyden önce **sadelik** için çabalayın. Amaç, karmaşık gereksinimleri uygulamak için basit yollar bulmaktır. İnsanların tasarımlarınızı "zarif", "uygulaması, test edilmesi ve bakımı kolay" olarak tanımlayacağını ummalısınız.

Bir zanaat olarak yazılım geliştirme

Yazılım geliştirme bir **zanaattır**. Yazılım geliştiricilerin tasarımda kullanılabilecek çeşitli **araçları** vardır, ancak belirli bir uygulama için uygun aracı seçmeniz gerekir.

Sistem Mimarisi

Sistem mimarisi, bir sistemin genel tasarımıdır:

- Bilgisayarlar ve ağlar (örneğin, şirket içi bilgisayarlar, bulut hizmetleri)
- Arayüzler ve protokoller (ör. http, IMAP, ODBC)
- Veritabanları (ör. ilişkisel, dağıtılmış)
- Güvenlik
- İşlemler (ör. yedekleme, arşivleme, denetim izleri)

Geliştirme sürecinin bu aşamasında, şunları da seçmelisiniz:

- Yazılım ortamları (örneğin, diller, veritabanı sistemleri, sınıf frameworkleri)
- Test frameworkleri

Sistem Mimarisi için Modeller

Sistem mimarisi modellerimiz UML'ye dayanmaktadır.

Her sistem için bir model seçeneği vardır. Sistemi en iyi modelleyen ve herkes için en net olan modelleri seçin.

UML'de, her model hem bir diyagrama hem de destekleyici bir spesifikasyona sahip olmalıdır.

Ancak bu derste, destekleyici özellikler olmadan sistemin ana hatlarını veren diyagramlar bulunmaktadır.

Diyagramlar, sistemin parçaları arasındaki ilişkileri gösterir, ancak bir sistemi belirtmek için çok, çok daha fazla ayrıntı gerekir.

Alt sistemler (Subsystems)

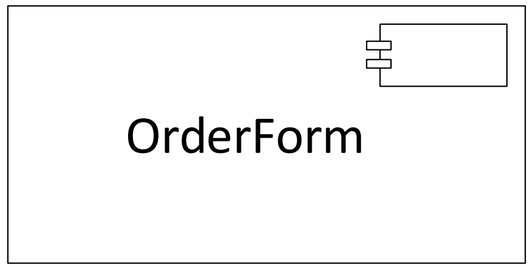
Alt sistemler

Alt sistem, bir sistemin parçasını oluşturan öğelerin gruplandırılmasıdır.

- **Coupling** iki alt sistem arasındaki bağımlılıkların bir ölçüsüdür. İki sistem güçlü bir şekilde birbirine bağlı ise birini değiştirmeden diğerini değiştirmek zordur.
- **Cohesion** bir alt sistem içindeki bağımlılıkların bir ölçüsüdür. Bir alt sistem birbiriyle yakından ilişkili birçok işlev içeriyorsa, uyumu (cohesion) yüksektir.

Karmaşık bir sistemin alt sistemlere ideal bir şekilde bölünmesi, alt sistemler arasında düşük bağlantıya (coupling) ve alt sistemler içinde yüksek uyuma (cohesion) sahiptir.

Component (Bileşen)



Bir bileşen, bir dizi arayüze (interface) uyan ve bunların gerçekleştirilmesini sağlayan bir sistemin değiştirilebilir bir parçasıdır.

Bir bileşen, bir alt sistemin implementasyonu olarak düşünülebilir.

Bir bileşenin UML tanımı

"Yazılım kodu (kaynak, binary veya çalıştırılabilir) ve bir insan sistemindeki iş belgeleri vb. dahil olmak üzere bir sistemin dağıtılabilir bir uygulama parçası."

Değiştirilebilir Elemanlar Olarak Bileşenler

Bileşenler, sistemlerin binary değiştirilebilir elemanlarla oluşturulmasına olanak tanır.

- Bir bileşen, arayüzlere uyan başka herhangi bir bileşen(ler) ile değiştirilebilir.
- Bileşen, bir sistemin parçasıdır.
- Bir bileşen, bir dizi arayüzün gerçekleştirilmesini sağlar.

Bileşenler ve Sınıflar (Class)

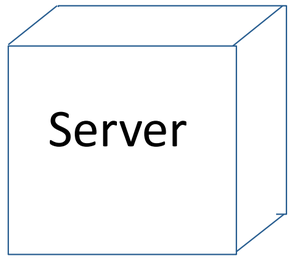
Sınıflar mantıksal soyutlamaları temsil eder. Nitelikleri (verileri) ve işlemleri (metot) vardır. Sınıflar programlar oluşturmak için birleştirilebilir.

Bileşenler, yalnızca **arayüzler** aracılığıyla erişilebilen işlemlere sahiptir. Bileşenler, sistemler oluşturmak için birleştirilebilir

Paket (Package)

Package, öğeleri gruplar halinde düzenlemek için genel amaçlı bir mekanizmadır.

Düğüm (Node)



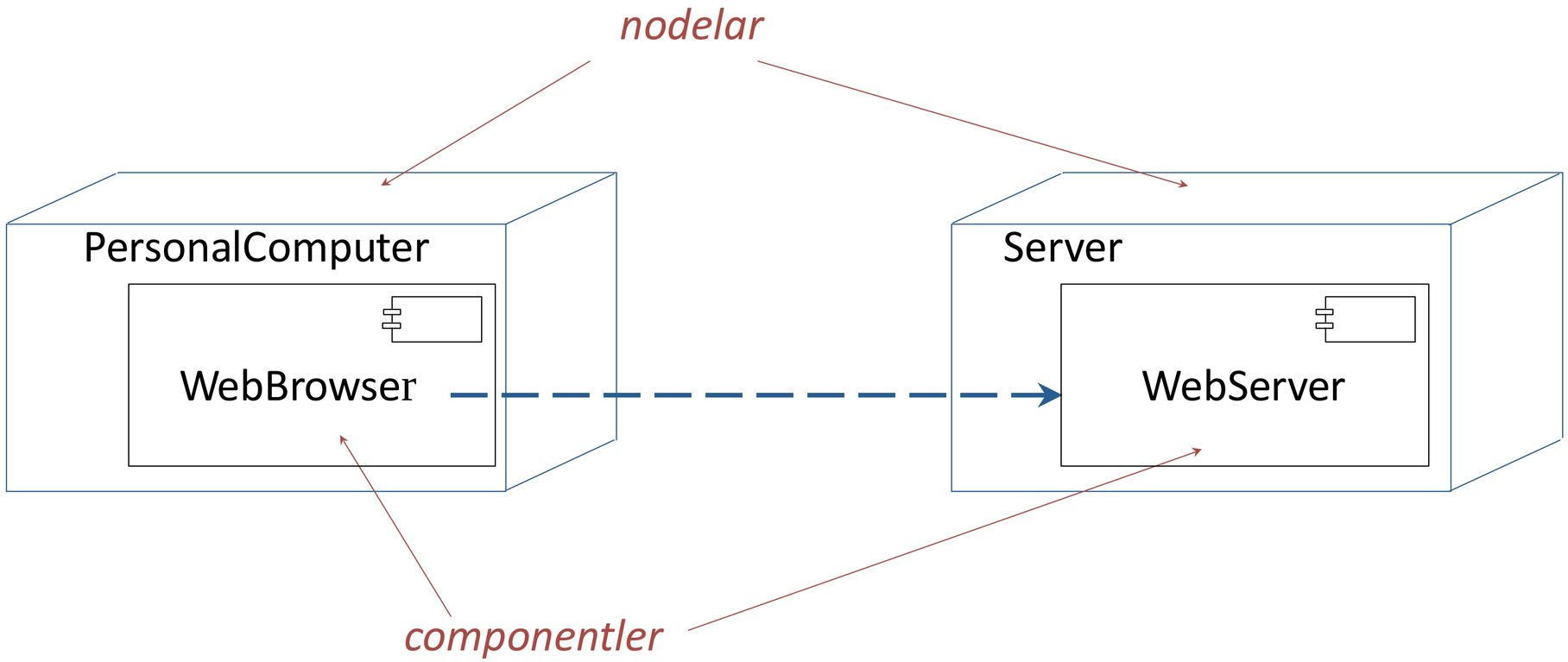
Düğüm, çalışma zamanında var olan ve bir bilgisayar, akıllı telefon, router gibi bir hesaplama kaynağı sağlayan fiziksel bir öğedir.

Bileşenler düğümlerde **yaşar ve** düğümler bileşenleri yürütür.

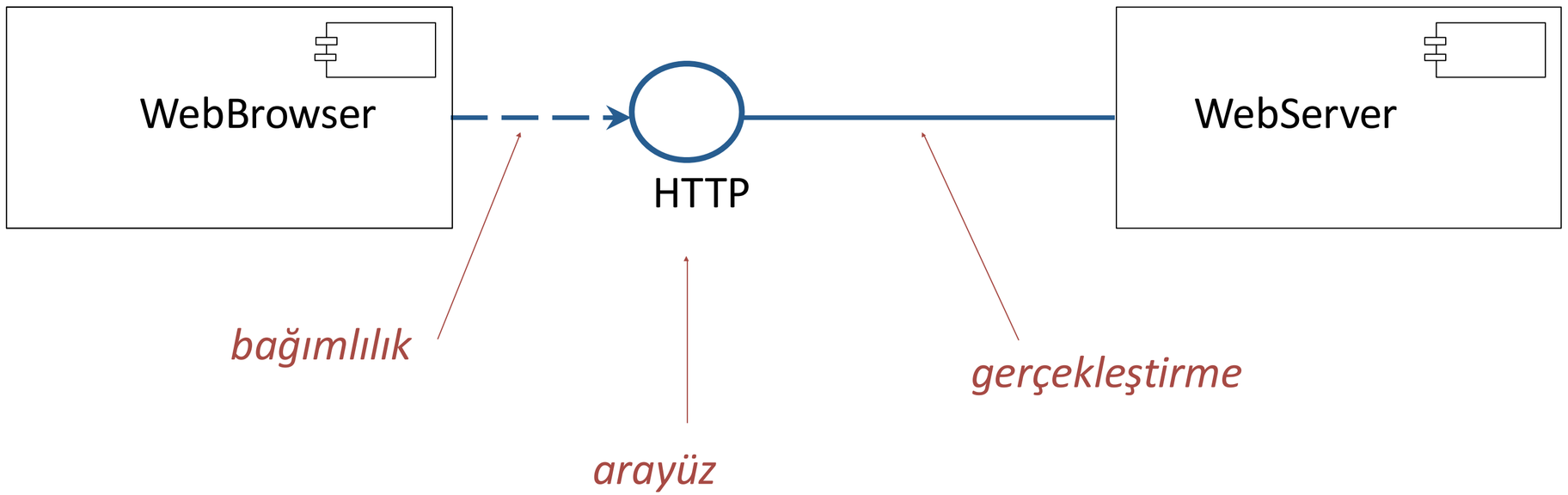
Örnek: Basit Web Sistemi

- Sunucudan gelen statik sayfalar
- Tüm etkileşimler sunucu ile iletişim gerektirir

Deployment Diyagramı

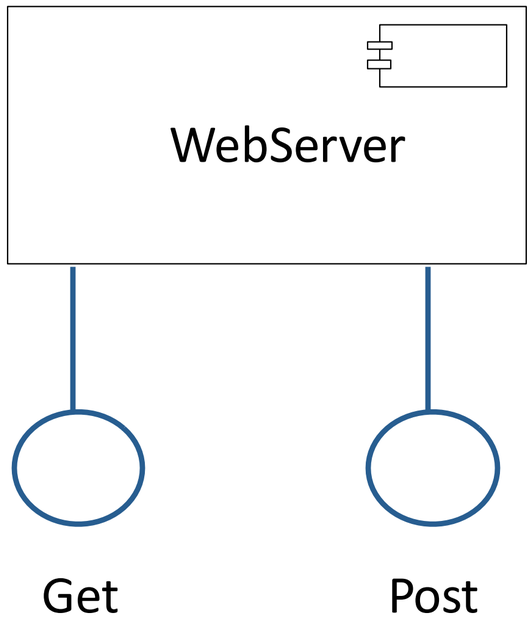


Component Diyagramı: Arayüzler (interface)



Application Programming Interface (API)

API, bir veya daha fazla bileşen tarafından gerçekleştirilen bir arayüzdür.



Mimari Tarzlar

Mimari stil, birçok farklı uygulamada tekrarlanan sistem mimarisidir.

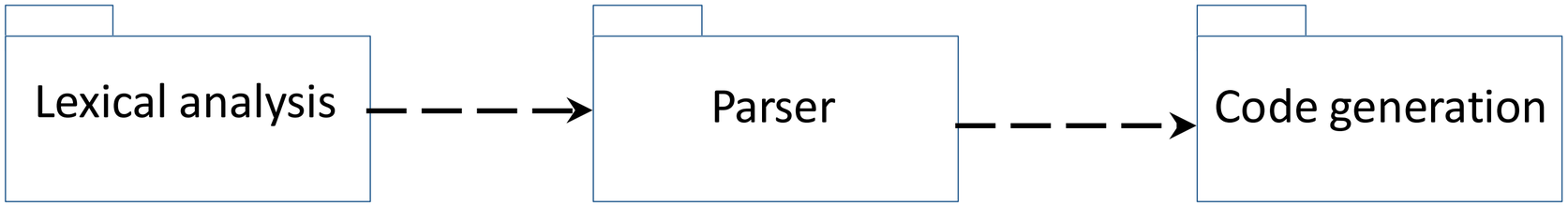
Bakınız:

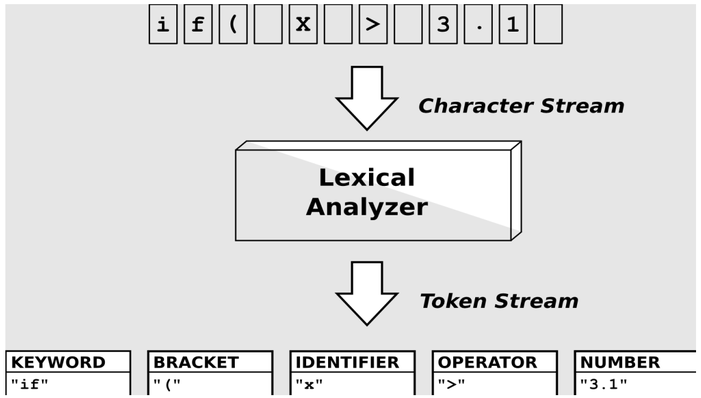
- Mary Shaw and David Garlan, *Software architecture: perspectives on an emerging discipline*. Prentice Hall, 1996
- David Garlan and Mary Shaw, *An Introduction to Software Architecture*. Carnegie Mellon University, 1994

  http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf

Mimari Stil: Pipe

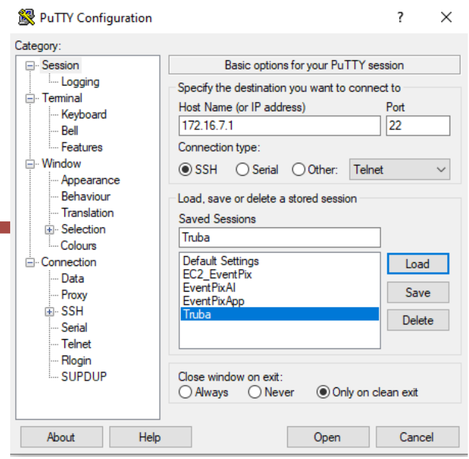
Örnek: Üç geçişli bir derleyici



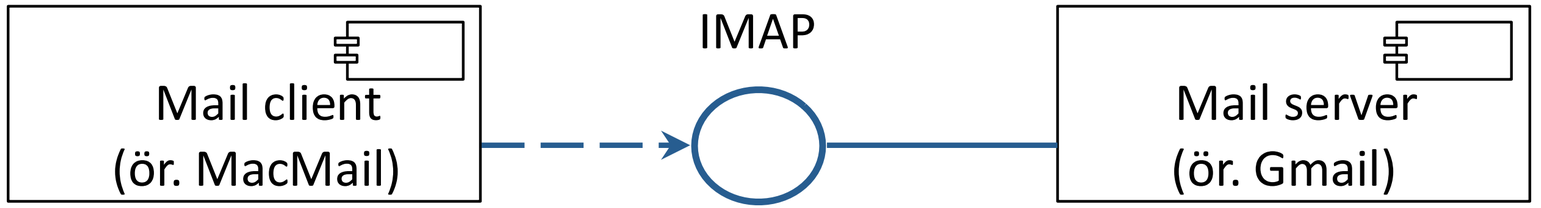


Bir alt sistemden gelen çıktı, bir sonrakine yapılan girdidir.

Mimari Stil: Client/Server



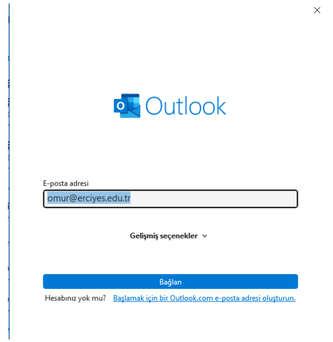
Örnek: Bir e-posta sistemi

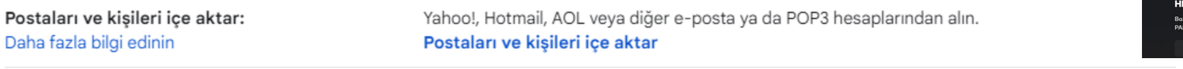


İstemci ve sunucudaki denetim akışları bağımsızdır.
İstemci ve sunucu arasındaki iletişim bir protokol ile gerçekleştirilir.

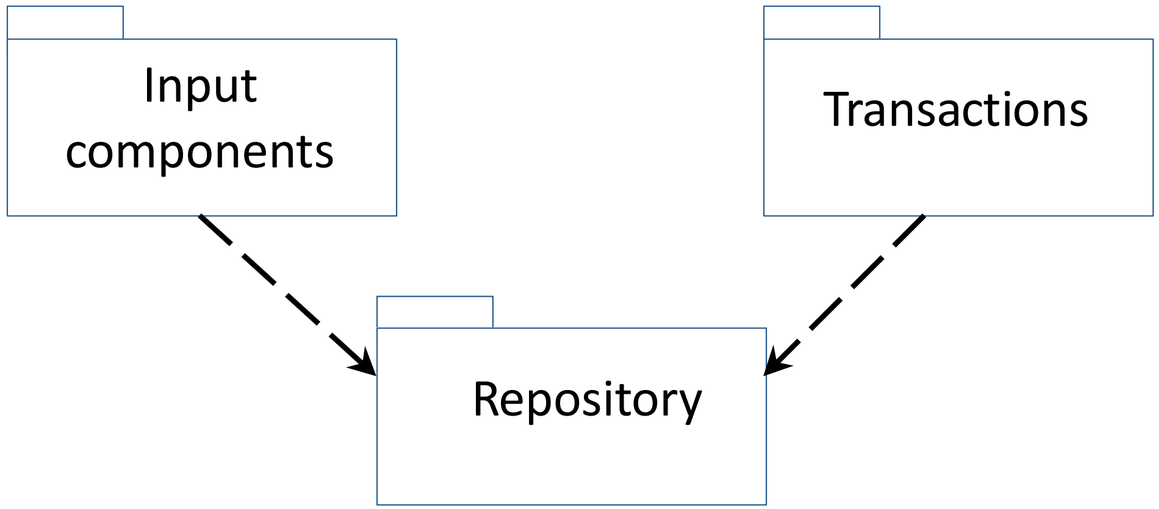
Bileşenler binary değiştirilebilir öğeler olduğundan, istemci veya sunucu, protokolü uygulayan başka bir bileşenle değiştirilebilir.

Peer-to-peer bir mimaride, aynı bileşenler hem istemci hem de sunucu olarak işlev görür.





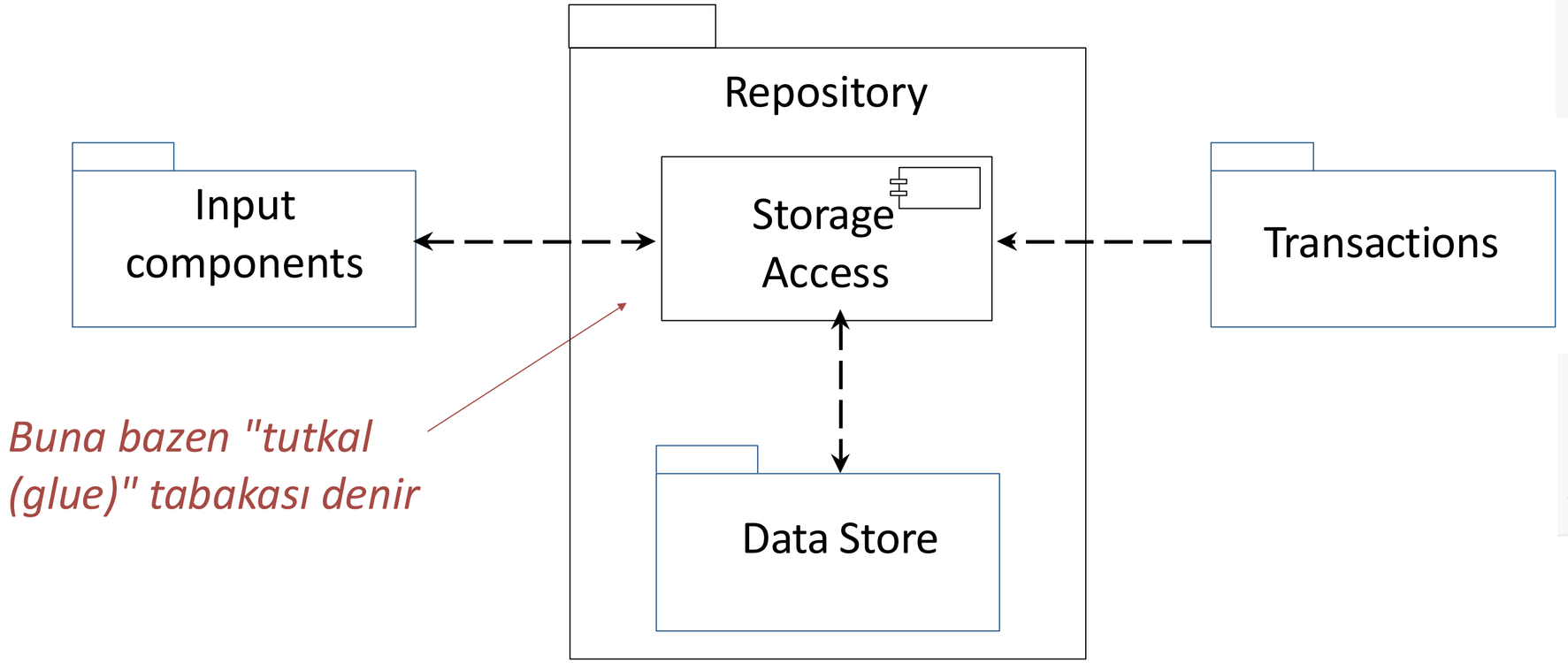
Mimari Stil: Repository



Avantajlar: Veri yoğun sistemler için esnek mimari.

Dezavantajlar: Diğer tüm bileşenler ona bağlı olduğu için repository'i değiştirmek zor.

Mimari Stil: Repository with Storage Access Layer



Avantajlar: Data Storage alt sistemi, Storage Access dışında herhangi bir bileşen değiştirilmeden değiştirilebilir.

Zaman Açısından Kritik Sistemler

Zaman açısından kritik (gerçek zamanlı) bir sistem, doğru çalışması üretilen sonuçlara ve bunların üretildiği zamana bağlı olan bir yazılım sistemidir.

- Sonuçlar gerekli zaman kısıtlamaları içinde üretilmezse, katı (hard) gerçek zamanlı bir sistem başarısız olur,

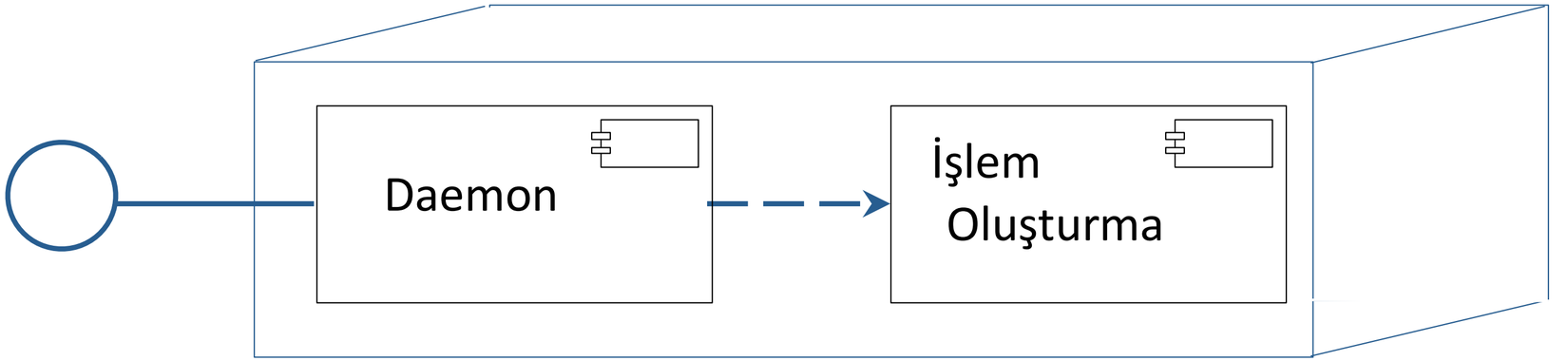
  Örneğin, bir uçak için kablolu uçuş kontrol sistemi, belirtilen zaman sınırları içinde yanıt vermelidir.

- Esnek (Soft) gerçek zamanlı sistem, sonuçlar gerekli zaman kısıtlamaları içinde üretilmezse bozulur,

  örneğin, bir ağ yönlendiricisinin bir paketi zaman aşımına uğratmasına veya kaybetmesine izin verilir.

Zaman Açısından Kritik Sistem: Mimari Stil - Daemon

Daemon, mesajların işlenme süresinden daha yakın aralıklarla gelebileceği durumlarda kullanılır.



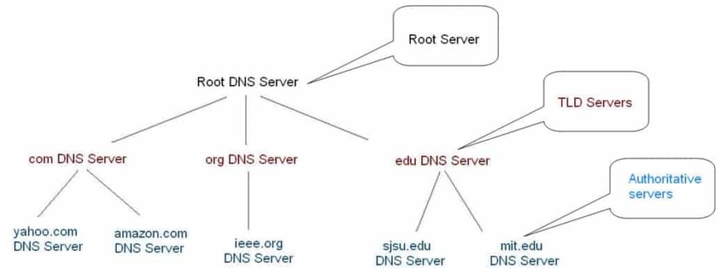
Örnek: Web sunucusu

Daemon 80 numaralı bağlantı noktasında dinler

Bir mesaj geldiğinde:

- İletiyi işlemek için bir işlem oluşturur
- 80 numaralı bağlantı noktasında dinlemeye geri döner

Dağıtılmış Veriler için Mimari Stiller



Çoğaltma (Replication):

Verilerin birkaç kopyası farklı konumlarda tutulur.

Mirror: Tüm veri seti çoğaltılır

Cache: Dinamik veri kümesi çoğaltılır (örneğin, en son kullanılan)

Çoğaltılan verilerde en büyük sorunlar eşzamanlılık ve tutarlılıktır.

```
C:\Users\Omur>ping google.com

Pinging google.com [172.217.17.142] with 32 bytes of data:
Reply from 172.217.17.142: bytes=32 time=52ms TTL=115
Reply from 172.217.17.142: bytes=32 time=52ms TTL=115
Reply from 172.217.17.142: bytes=32 time=52ms TTL=115
Reply from 172.217.17.142: bytes=32 time=52ms TTL=115

Ping statistics for 172.217.17.142:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 52ms, Maximum = 52ms, Average = 52ms
```

Örnek: Alan Adı Sistemi

Protokolün detayları için okuyabilirsiniz:

Paul Mockapetris, "Domain Names - Implementation and Specification". IETF Network Working Group, *Request for Comments*: 1035, November 1987.

http://www.ietf.org/rfc/rfc1035.txt?number=1035

PostgreSQL

Örnek Sınav Sorusu

Spor malzemeleri üreten bir şirket, çevrimiçi spor malzemeleri satmak için bir sistem oluşturmaya karar verir. Şirket, ürettiği ekipmanın açıklamasını, pazarlama bilgilerini ve fiyatlarını içeren bir ***ürün veritabanına*** *zaten sahiptir.*

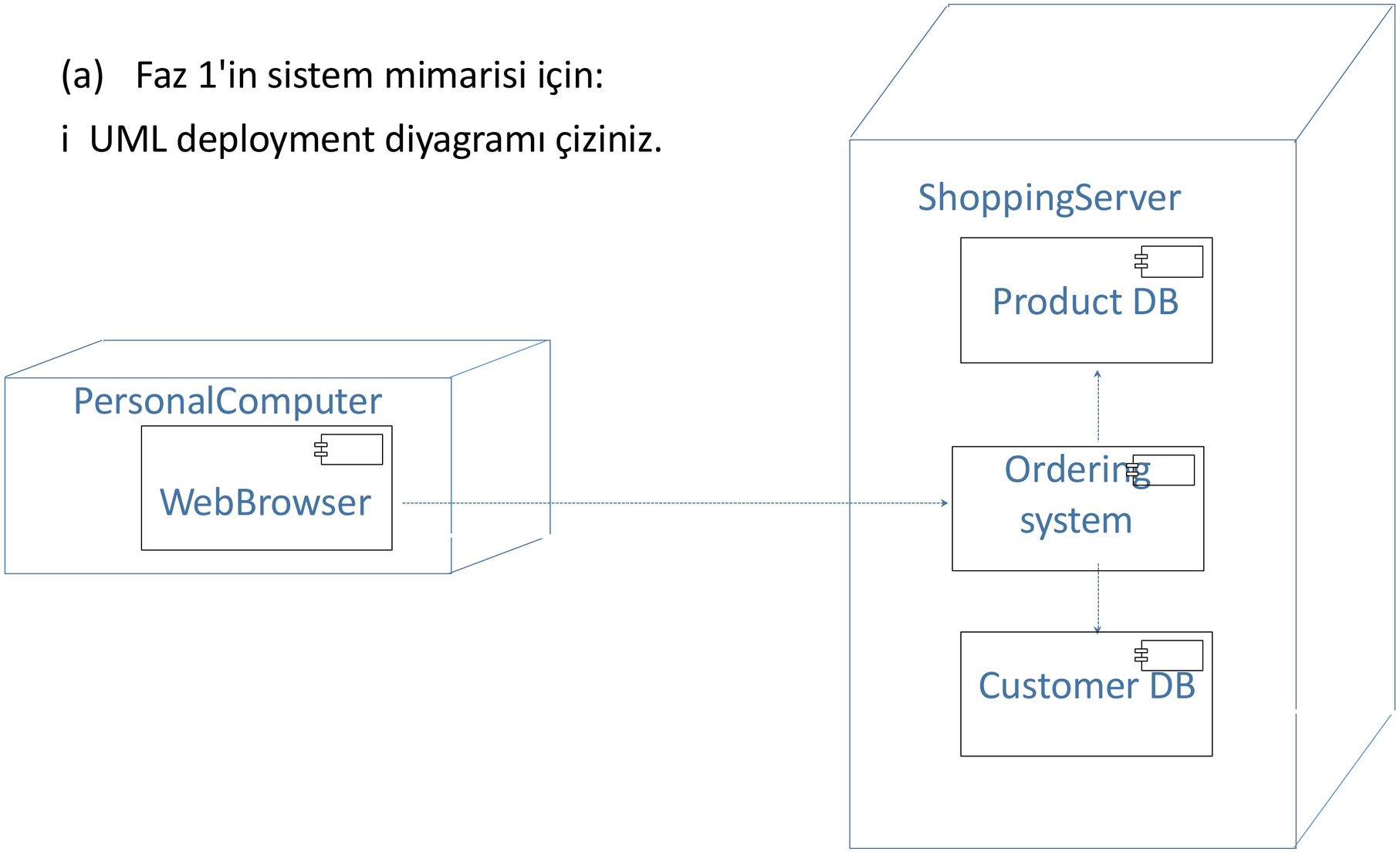
Çevrimiçi ekipman satmak için şirketin şunları oluşturması gerekmektedir: bir ***müşteri veritabanı*** *ve çevrimiçi müşteriler için bir* ***sipariş sistemi****.*

Plan, sistemi iki aşamada geliştirmektir. Aşama 1'de, müşteri veri tabanının ve sipariş sisteminin basit versiyonları production ortamına alınacaktır. Aşama 2'de, bu bileşenlerde önemli geliştirmeler yapılacaktır.

Örnek Sınav Sorusu

(a) Faz 1'in sistem mimarisi için:

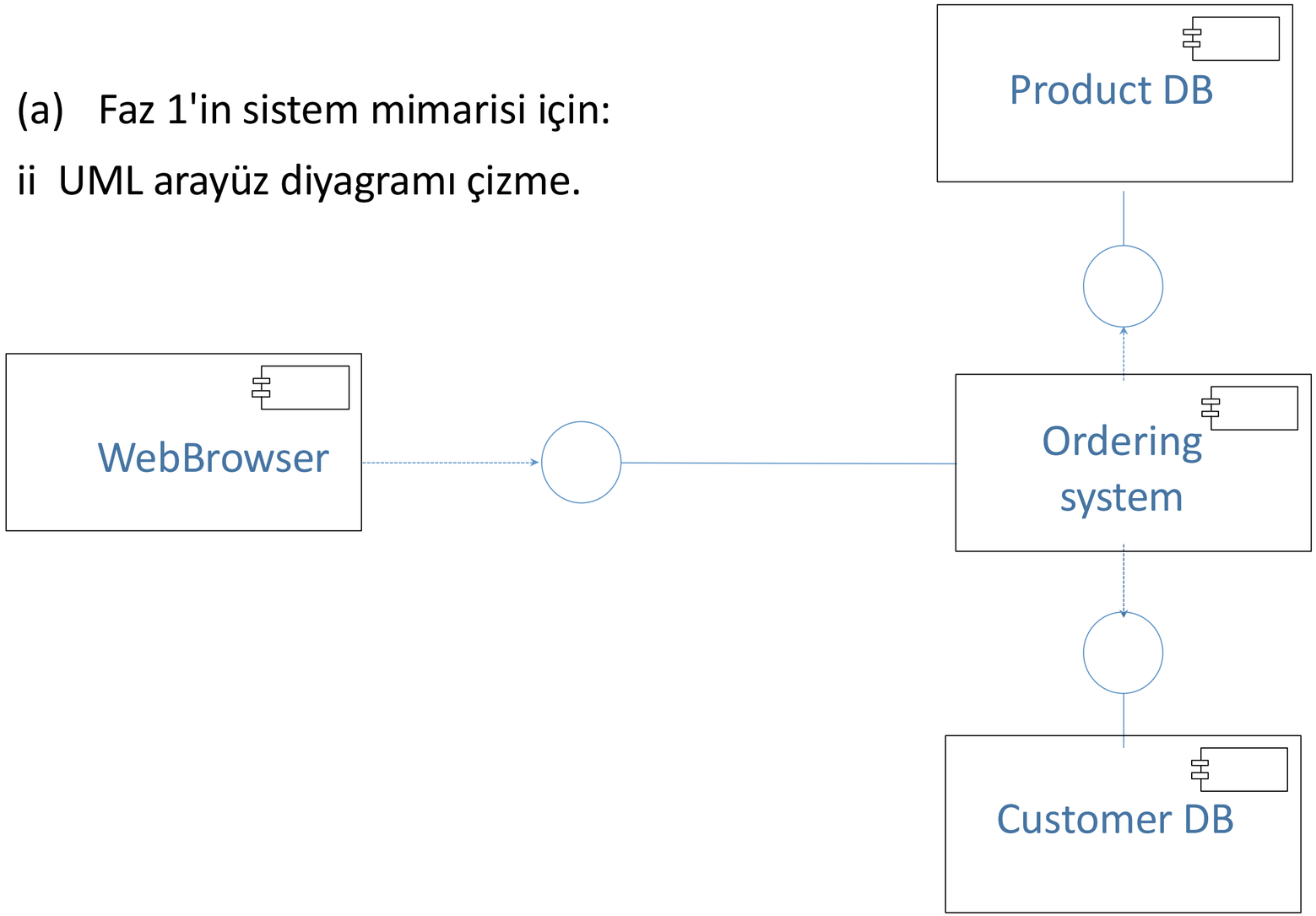
i UML deployment diyagramı çiziniz.



Örnek Sınav Sorusu

(a) Faz 1'in sistem mimarisi için:

ii UML arayüz diyagramı çizme.



Örnek Sınav Sorusu

(b) Aşama 1 için:

i Müşteri veritabanı için hangi mimari stili kullanırsınız?

Repository with Storage Access Layer

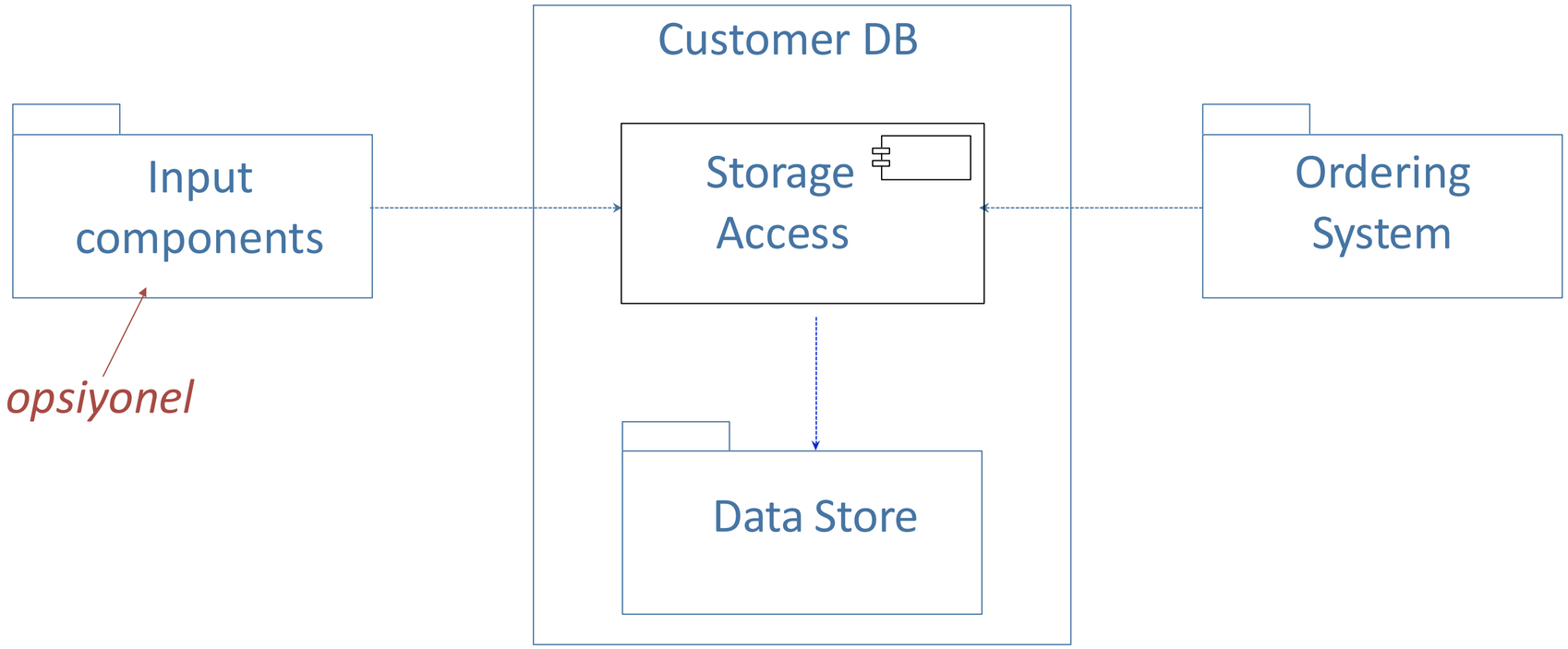
ii Neden bu stili seçtiniz?

Veritabanını kullanan uygulamaları değiştirmeden veritabanının değiştirilmesini sağlar.

Örnek Sınav Sorusu

(b) Aşama 1 için:

iii Bu mimari stilin uygulamadaki kullanımını gösteren bir UML diyagramı çizin.



Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ 313 Yazılım Mühendisliği

12. Sistem Mimarisi

Ders Sonu