YAZILIM MİMARİLERİ

Ömer CENGİZ

16260056

Yazılım Mimarisi

- Bir yazılım veya bilgisayar sisteminin yazlım mimarisi demek sistemi oluşturan yazılım bileşenleri ve görünür dış özellikler ile bunlar arasındaki ilişkiler demektir.
- Belirlenen teknik ve operasyonel ihtiyacın giderilmesi için performans, güvenlik ve yönetim vb. gibi genel kalite özelliklerini de karşılayacak şekilde yapısal bir çözüm tasarlama işlemidir.
- Yazılım mimarisi yapı ve davranışların yanı sıra kullanılabilirlik ,fonksiyonellik, performans, esneklik, yeniden kullanım, anlaşılırlık, ekonomik ve teknolojik kısıtlar gibi özellikleri de yansıtır. Tüm bu özellikleri ile "Yazılım Mimarisi" sistemin anayasasıdır.Tüm yazılım geliştirme sürecinin merkezinde durur ve her türlü faaliyete kılavuzluk eder.

Yazılım Mimarisi Tasarlarken Nelere Dikkat Edilmeli

- Yazılım mimarisinde kötü kararlar alırsanız projenin tamamen çöp olmasına sebep olabilirsiniz.Bu sebeple çok dikkatli kararlar almak lazım.
- İyi bir yazılım mimarisi gelişime açık olmalıdır.
- Risk alanlarına dikkat edilmeli.Gerçekten ihtiyaç olan hizmetleri oluşturmak için gerekli ihtiyaçları oluşturmak lazım
- Problemleri tespit ederken bir müşteri gibi davranmalısınız.
- Geri dönüşüm ve saydamlık maksimum olmalıdır.

Yazılım Mimarisi Ne İçerir

- Sistemin organizasyonu hakkında önemli kararlar
- Sistemin önemli yapısal elementleri ve bunların arayüzleri ile birbirleri arasındaki etkileşim
- Sistemin önemli yapısal ve davranışsal elemanlarının alt sistemlere dağılımı

Yazılım Mimarisinin Önemi

- Sistemin karmaşıklığını yönetmek ve bütünlüğünü korumak için , kontrol edilebilir bir yapı sunar.
- Yeniden kullanımı arttıran kullanışlı bir yapı sunar
- Proje yönetimi için temel teşkil eder:

Alt Sistemler

- **Bileşen** sistem içerisinde bulunan , iyi tanımlanmış arabirim ve davranışlarıyla içeriği hakkında güçlü bir kapsulleme sunan , aynı zamanda değiştirilebilir , kaynak ya da çalışabilir durumda ki bir gurup koda verilen isimdir.

- Sistem dizaynının ilk adımı sistemin ana bileşenlerinin tariflenmesidir.Bu ana bileşenler aynı zamanda **altsistem** olarak da adlandırılırlar.Altsistemler bir sınıf ya da fonksiyon değildirler.Altsistemler sınıflar , ilişkiler , operasyonlar , olaylar , kurallar gibi birbiriyle ilişkili pek çok kavramı barındıran paketlerdir.Altsistemler genellikle sundukları **servis** ile tanımlanırlar.Servis ortak bir amaca hizmet eden bir gurup fonksiyon olarak tanımlanabilir.Örneğin işletim sistemleri bünyelerinde bellek yönetimi , dosya yönetimi gibi alt sistemler bulundururlar.

İki alt sistem arasında kurulacak ilişkiler

- **İstemci – Destekçi** : Bu tür ilişkide istemci , destekçi'nin arabirimini bilmek zorundadır.Bu arabirim üzerinden istediği servisleri alır.Ancak bu tür bir ilişkide destekçi ; istemci hakkında bir bilgiye sahip olmak zorunda değildir.Tek taraflı bir bağımlılık söz konusudur.
- **Eşdüzeyli** : Bu tür bir ilişkide altsistemler birbirlerini kullanırlar.Bu tür ilişkiler daha karmaşıktır.Her iki altsistem de birbirinin arabirimini bilmek zorundadır.İletişimin dizaynı oldukça karmaşık ve hataya açıktır.

Katmanlar

- Katmanlar , sistem içersinde birbiri ardına gelen sanal dünyalar olarak nitelendirilebilir.En üst seviye katmanlar uygulamaya has fonksiyonalite sunarken , alt seviyeler daha çok çalışma ortamına özel bileşenleri içerirler.Genel amaçlı servisler ve iş mantığı servisleri ise genellikle orta seviyeler yer alır. Katmanlar içerisinde birbiriyle benzeşen nesneler olabilir.

- Katmanlar arasında bilgi tek yönlüdür.Her katman daha alt seviyede ki katmanlar hakkında bilgi sahibi olabilir ancak üst seviyeler hakkında bir bilgisi yoktur.Alt katmanlar ile üst katmanlar arasında istemci-destekçi tarzı ilişki söz konusudur.

Katmanlı Mimaride Kapalı Mimari ve Açık mimari

- **Kapalı Mimari:**

Kapalı mimaride katmanlar sadece takip eden katmanla iletişim kurabilir.Bu katmanlar arasında bağımlılıkları azaltır ve değişiklikler için daha esnek bir sistem sağlar.Çünki katman'ın arabirimleri üzerinde yapılacak değişiklik sadece bir katmanı etkiler.

- **Açık Mimari:**

Açık mimaride katman ; alt seviyelerin tümüne erişebilir .Bu operasyonların her seviyede yeniden tanımlanma gereksinimini ortadan kaldırır ve daha verimli ; küçük kodlar sağlar.Ancak değişiklikler pek çok seviyeyi etkileyebilir.Bu yüzden açık mimari , kapalı mimariye göre daha az güvenli bir yapı sunar.

Bileşenler

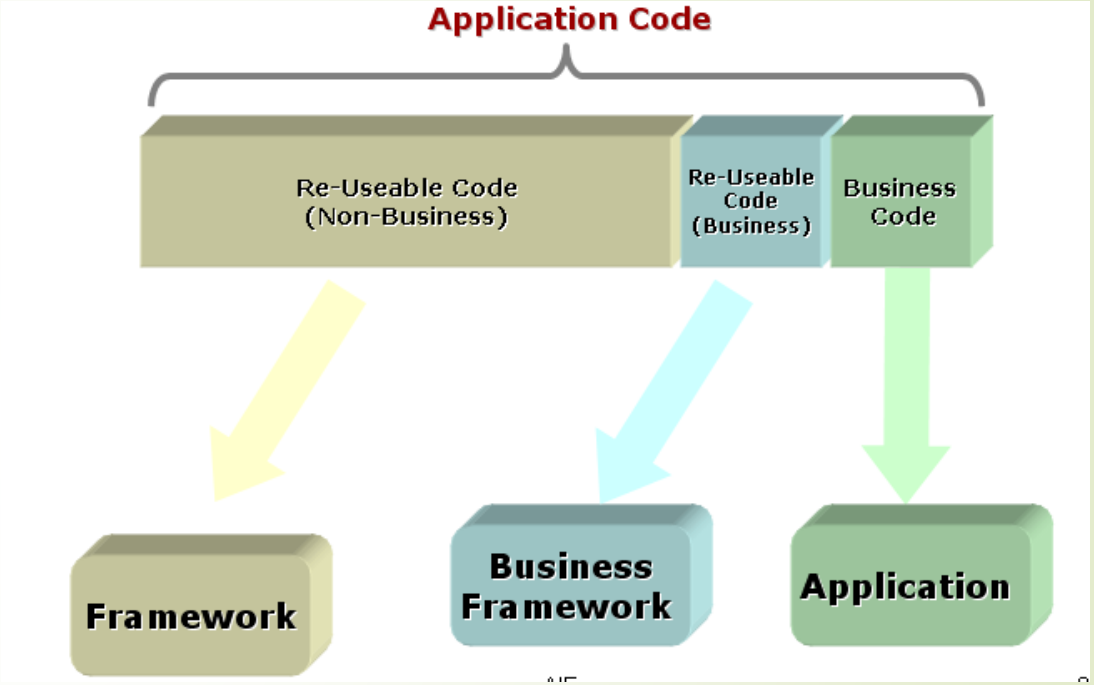
- Bileşenler sistem içerisinde modul , paket ya da altsistem gibi kesin sınırları olan ve sorumluluğu belirlenmiş elemanlar olarak da tanımlanabilir.Bileşenler sisteme entegre olabilir ve değiştirebilir yapılardır.Bu aynı zamanda bileşenlerin başka sistemler içerisine de entegre edilebilir ve yeniden kullanılabilir olduğunu anlatır.

- Moduler bir mimaride bileşenleri bağımsız olarak tanımlar , izole eder , dizayn eder ve geliştirebilirsiniz edebilirsiniz.Daha sonra bileşenler yine bağımsız olarak test edilip sisteme entegre edilebilirler. Bu bileşenlerden bir kısmı genel çözümler sundukları tesbit edilirse daha geniş alanlarda kullanılabilirler.Bu yeniden kullanılabilir bileşenler organizasyonun genel yazılım verimliliğini ve kalitesini arttırmanın bir yoludur.

Çalışma Alanı (Domain) Kavramı

- Sistemimizi yapısal olarak katmanlara ve bölümlere ayırabiliyoruz.Bir üçüncü boyut olarak gelen **çalışma alanı (domain)** aslında sistem içerisinde adreslenen çözümlerin genelden özele doğru yeniden kullanılabilir parçalar halinde düzenlenmesidir diyebiliriz.
- Örneğin muhasebe alt sisteminin genel , hazırlanan alana özgü (sigorta , bankacılık vb) ve şirkete özgü parçaları.Çalışma alanlarından katmanlar ve bölümler gibi yapısal bir parçalanma değil mantıksal bir parçalanma olarak söz edebiliriz.
- Yazılım mühendisliğini istekler doğrultusunda geliştirilen , tek sisteme yönelik bir çalışma olduğunu varsayarsak , çalışma alanı mühendisliği pek çok sisteme uyabilecek yeniden kullanılabilir çözümler sunan bir çalışmadır denilebilir.

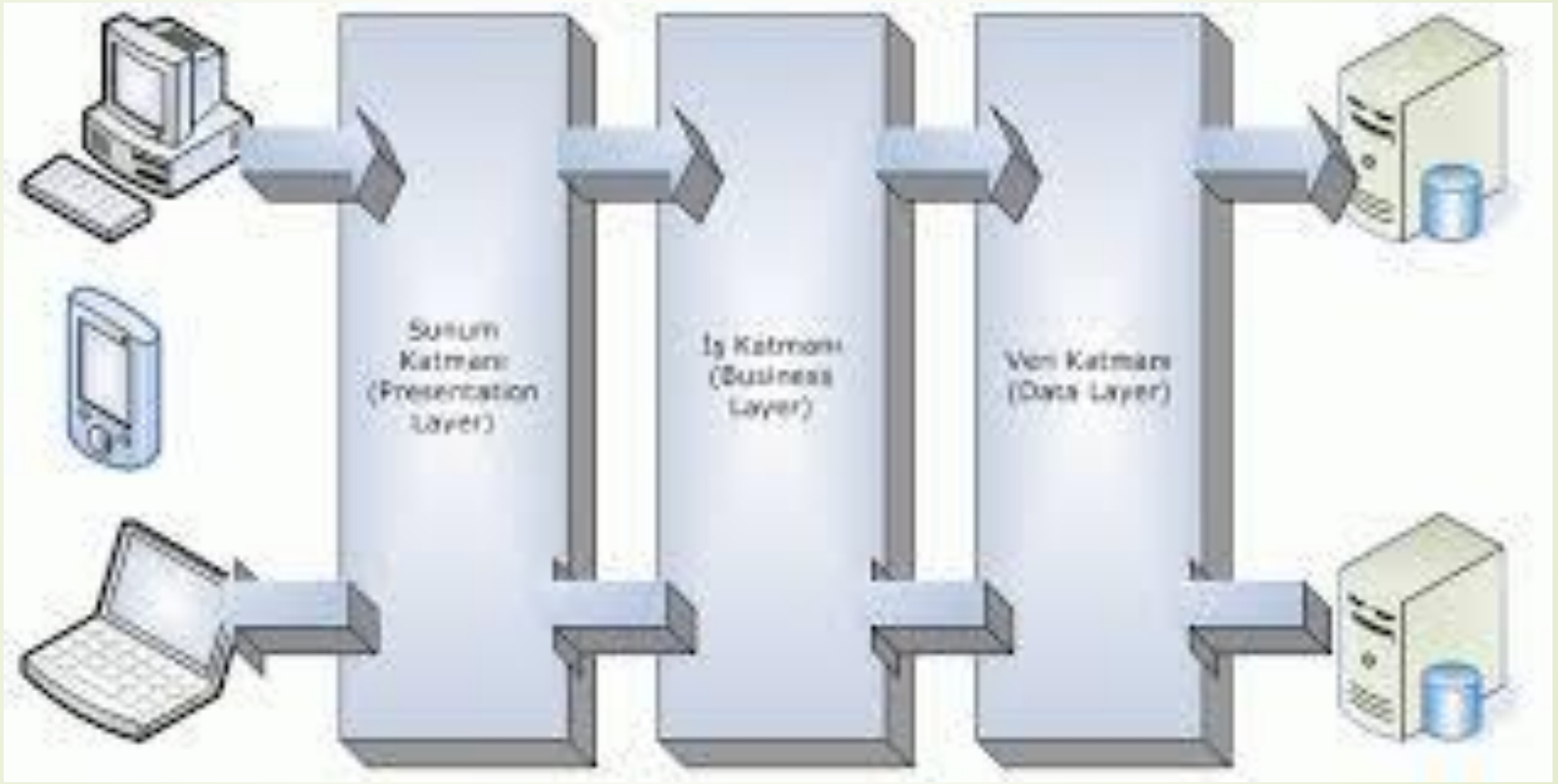
Örnek şekil bir uygulama parçalarının nasıl ele alındığını ve içindeki çalışma alanlarının nasıl tesbit edildiğini göstermektedir.Buna göre uygulama içerisindeki iş mantığı içermeyen yeniden kullanılabilir kodlar framework içerisinde toplanmış , işe özel ancak o iş içinde yeniden kullanılabilir kodlar bir başka parçya ayrıldıktan sonra geriye kalan kodlar uygulamaya özel olarak belirlenmiştir.



Yazılım Mimamrisinde Yazılım Katmanları

- Bir yazılım hazırlanırken, yazılımın kullanım amacına göre katmanlar oluşturulabilir veya tek katman üzerinde de çalıştırılabilir.
- **Veri Katmanı:** Verinin veritabanı sisteminden getirilmesi ve veritabanı sistemine gelen verilerin eklenmesi için kullanılan katmandır.
- **İş Katmanı:** Veritabanından gelen verilerin işimize uygun olarak düzenlenmesi ve kontrol edilmesini sağlayan katmandır.
- **Sunum Katmanı:** Kullanıcının göreceği ve kullanıcıdan girdiği verilerin alınacağı yada daha önceden girilmiş verilerin bir şekilde kullanıcıya gösterilmesi için gereken bir katmandır.

Yazılım Katmanları



Tek Katmanlı Yazılımlar

- Bu mimaride katmanlar tekbir bilgisayar ve yazılımda bulunmaktadır.Tek kullanıcılı bir sistem olduğu için hızlıdır fakat çok kullanıcıya destek sağlamadığı için kullanım oranı sınırlıdır.
- İlk yazılan programlar, tek katmanlı "Monolithic" olarak kodlanıyordu. Tek katmanlı uygulamalarda iş katmanı, sunum katmanı ve veritabanı katmanları tek bir bilgisayarda bulunuyorlardı. Buna en iyi örnek Adabas veri tabanı sistemi verilebilir. Adabas sistemleri bir veri tabanı ve bu veri tabanına direkt sorgulamalar yapabilecek formlardan oluşmaktadır.

- Bu yapıda bozulan kod bloğunun hatasının giderilmesi oldukça zordur. Çünkü o bölüm hata verdiğinde bütün sistemin çalışması durmaktadır.

Katmanlı Yazılım Mimarisi

- Katmanlı mimarinin en büyük yararı, kodlarımızı daha küçük yapılara bölerek kolay kontrol edilebilirlik ve güncellenebilirlik sağlar.Ayrıca verilerimizin güvenliğini de en yüksek seviyede korur.
- Örneğin bir web projesi için, tüm veritabanı işlemlerimizi, butonların arkasına yazılan bir sorgu olarak yaparsak, tasarım değişikliğinde tüm kodları tekrardan yazmamız gerekir. Ya da çıkan her hangi bir sorun anında, katmanlı mimaride sadece sorunlu kısmı inceleyecekken, *monolithic* adı verilen tek katmanlı yazılımda tüm projeyi incelememiz gerekir.

Tek Katmanlı Yazılım Mimarisi

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 sunucudur.

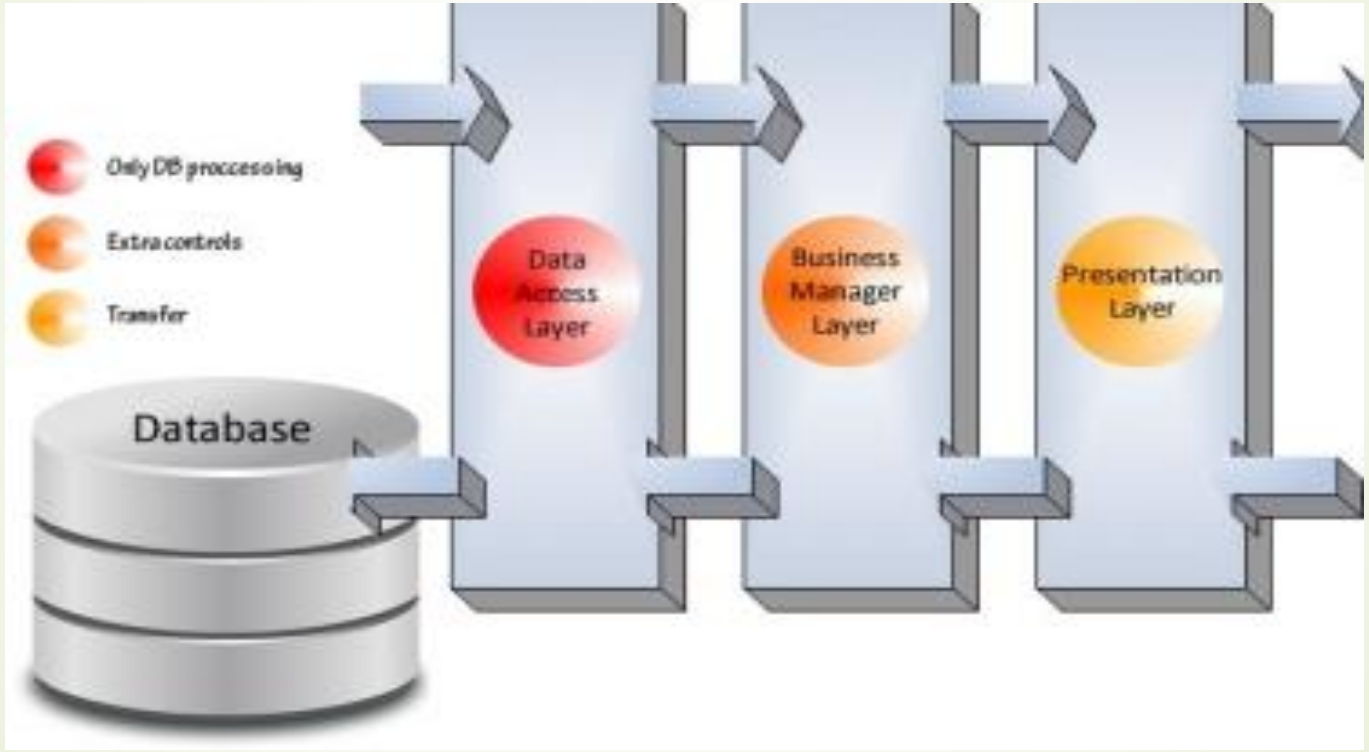
Üç Katmanlı Yazılım Mimarisi

- Kodlamalar sırasında işimizi kolaylaştıracak katmanlar mevcuttur. Genelde üç katman yapısı standarttır. İhtiyaca göre katman sayısı arttırılıp azaltılabilir. Çok katmanlı mimarinin tercih edilme nedeni proje yönetiminin kolay olması, ekip çalışmasına uygun olması, hata yönetiminin kolay olması vs. gibi nedenlerden tercih edilir.

  Genel olarak standart üç katman vardır;
  - **DAL** (Data Access Layer) - Veri Erişim Katmanı
  - **BL** (Busines Layer) - İş Katmanı
  - **PL** (Presentation Layer) - Sunum Katmanı

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

Üç Katmanlı Yazılım Mimarisi



İki ve Üç Katmanlı Mimarilerin Karşılaştırılması

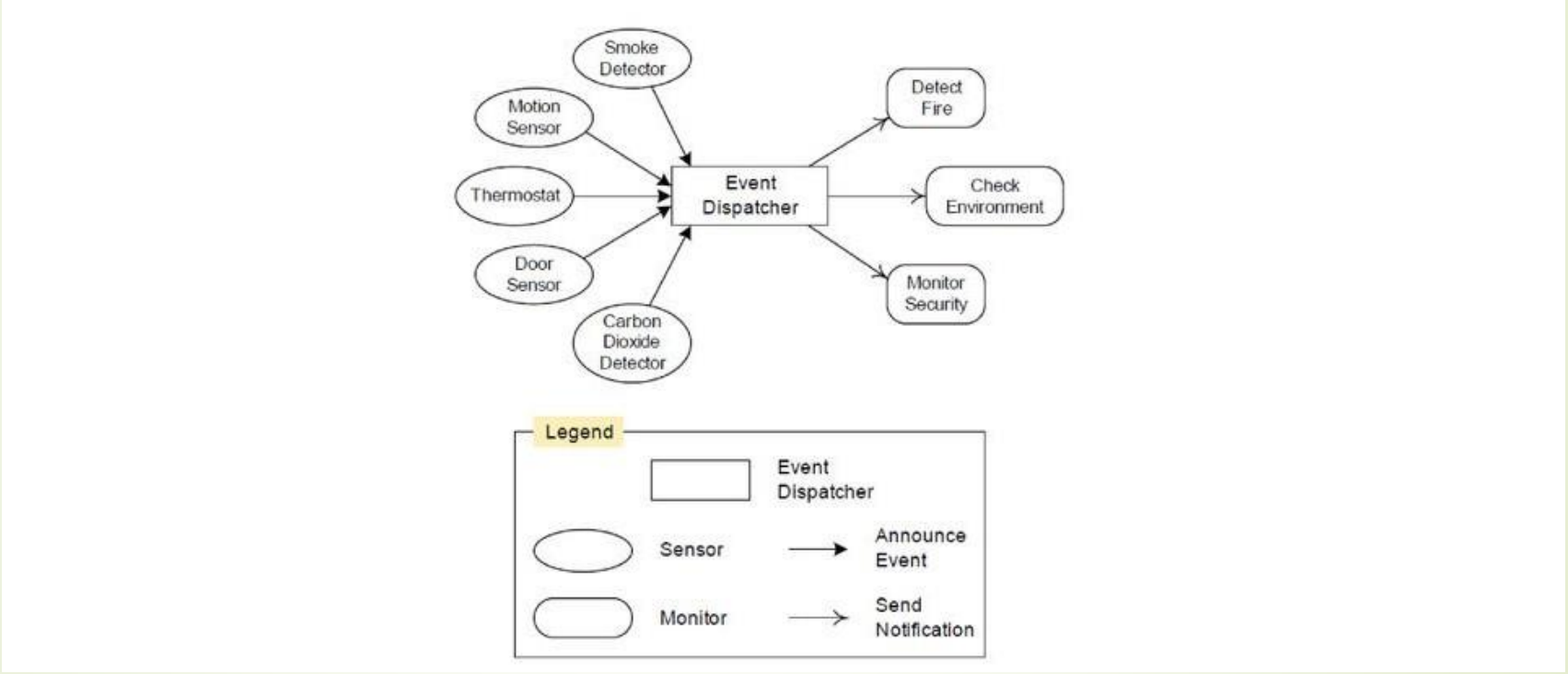
- İki katmanlı mimarilerde uygulama mantığı ve veri tabanının aynı bilgisayarda yer alması, sunucu tarafında geliştirilen uygulamanın VTYS'nin kendi dilinde olması nedeniyle birlikte çalışılabilirlik azalmaktadır.
- Üç katmanlı mimaride ise ara bir katmanın oluşu bu sınırlamayı kaldırmaktadır.
- Uygulama mantığının bu ara katmanda oluşu istemciyi veri tabanı sunucusundan bağımsız duruma getirmektedir.

Olay Güdümlü Yazılım Mimarisi

- Olay odaklı mimari (EDA), bir sistemin durumundaki önemli değişikliklerin (olaylar olarak bilinir) üretimini, algılanmasını ve bu sisteme tepki göstermesini ve buna tepki verilmesini teşvik eden bir yazılım mimarisidir. Bu, birbirine kısmen bağlı yazılım bileşenleri ve hizmetleri arasında olayları ileten uygulamalar ve sistemlerin tasarımı ve uygulanması yoluyla gerçekleştirilir.

- Bu mimariyi kullanan yazılımları programlama tekniğinde, olayları dinleyen fonksiyonlar yazılır. Beklenen olay numarası algılandığında, o olaya ilişkin alt işlev çağrılır. Beklenen olay karşısında bir fonksiyonun icra edilmesine olay yakalama (event capture) ya da sadece yakalama (capture) adı verilir. Beklenmeyen bir olay kodu geldiğine ise hiçbir işlem yapılmaz.

Olay Güdümlü Yazılım Mimarisi



Olay Güdümlü Yazılım Mimarisinin Avantajları

- Bileşenleri eklemek veya çıkartmak kolaydır.
- Bileşenler bağlı değildir bu sayede son derece yüksek yeniden kullanılabilirliğe ve değiştirilebilirliğe sahiptirler
- Bu mimariyi kullanılarak oluşturulan sistemler genelde sağlamdırlar ve hata töleransları yüksektir.

Servis Yönelimli Mimari (Service Oriented Architecture)

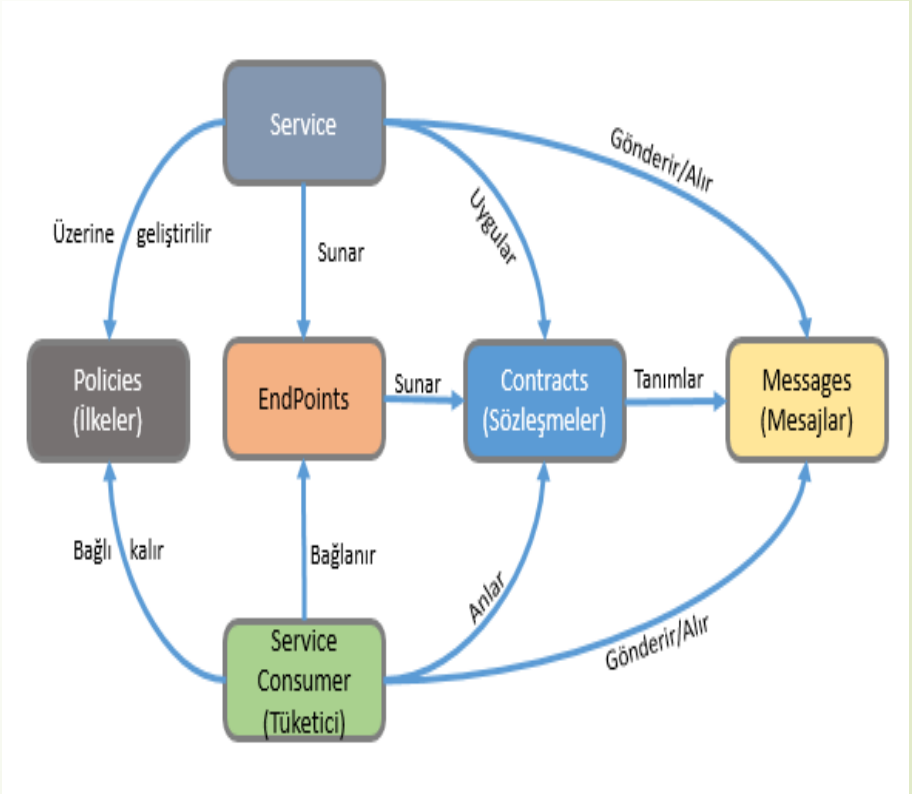
- SYM servis olarak adlandırılan, gevşek bağlı(loosely coupled), iri taneli(coarse grained-her ne demekse) ve özerk(autonoums) yapıdaki bileşenlere dayalı dağıtık sistemlerin geliştirilmesi için kullanılan mimari bir stildir. Zaten SYM'nın Architecture kelimesi bunun bir mimari yaklaşım olmasından dolayıdır. SOA farklı platformlar için belirli hizmetlerin sunulması esasına dayanmaktadır.

Servis Yönelimli Mimarinin (SYM)'nın Temel Bileşenleri

Service	Servisler **SOA**'nın n temel ve önemli üyesidir. Bir servis çoğunlukla ayrık bir iş fonksiyonelliği sunar.
Policies (İlkeler)	Bir servisin tüketicileri tarafından kullanılabilmesi için kısıtlar ve terimler tanımlar. **Security**, **Auditing**, **SLAs** vb **Policy** içerisinde yer alan dinamik özelliklerdir.
EndPoints	Bir **URI(Uniform Resource Identifier)** dir. Servisin bulunduğu adrestir. Endpoint' ler sözleşmeleri sunar.
Contracts (Sözleşmeler)	Servis tarafından desteklenen mesajların tümü, bir sözleşme ile sunulur.
Messages (Mesajlar)	**SOA** içindeki iletişim birimidir. **REST(Representational State Transfer), SOAP(Simple Object Access Protocol), JMS(Java Messaging Service)** vb türleri vardır.
Service Consumer (Tüketici)	Servisler ile mesajlaşma yoluyla iletişime geçen bileşenlerdir. Başka bir uygulama veya servis olabilir.

Servis Yönelimli Mimarinin Bileşenlerinin Birbiriyle İlişkileri

- Her bir servis(Service) sözleşmeler(Contracts) yoluyla iş süreçleri ve bunlara bağlı davranışlar sunar ki bu sözleşmeler keşfedilebilir EndPoint' ler üzerinden taşınabilecek mesajları(Messages) tanımlarlar. Bir servisin davranışı(Behavior), endüstriyel anlamda standartlaşmış çeşitli ilkeler(Policies) ile uyumlu olmak zorundadır. Servisleri tüketen taraflar(Service Consumers) ise, sözleşmeler ve mesajlar yoluyla tanımlanmış iş fonksiyonelliklerine(Business Functions) ulaşmaktadır.



Servis Yönelimli Mimarinin (SYM) Faydaları

- Öncelike SYM'nın dağıtık yazılım sistemlerinin kalitesini arttırma noktasında pek çok mimari kritere sahip olduğunu söylememiz gerekir. Yeniden kullanılabilirlik(reusability), uyumluluk(adaptability) ve bakım yeteneği(maintainability) bunlardan birkaçıdır.
- En önemlisi SYM'nın özellikle point-to-point entegrasyon yapan sistemlerdeki bağımlılıkları ortadan kaldıracak çözümleri içermesidir.
- Pek çok büyük sistem zamanla iş birimden gelen istekler doğrultusunda büyür ve genişler.

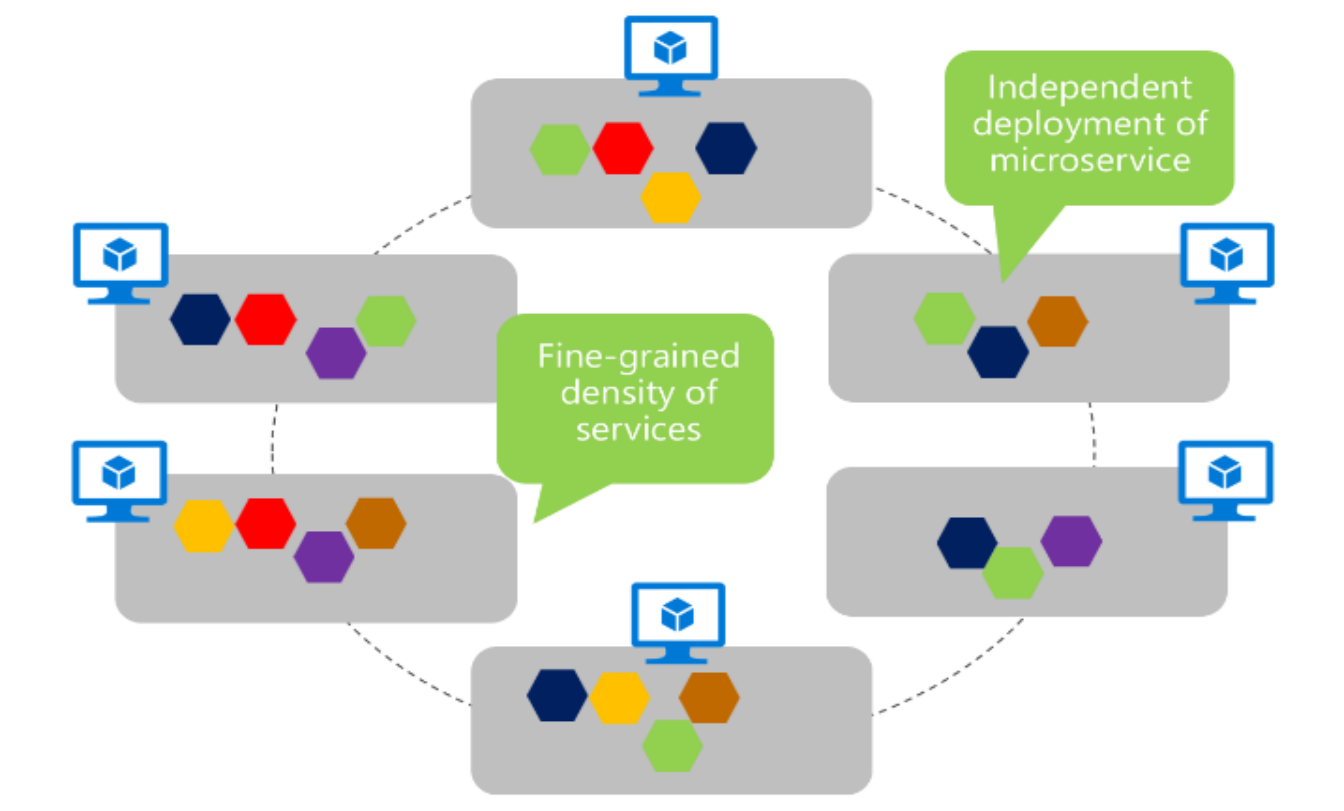
Mikrokernel mimarisi

- Mikrokernel mimarisi modeli (bazen eklenti mimari deseni olarak anılacaktır), ürün tabanlı uygulamaların uygulanması için doğal bir modeldir.
- Mikrokernel mimarisi deseni, çekirdeğe eklenti olarak ek uygulama özellikleri eklemenize olanak tanır ve genişletilebilirlik özellik ayrımı ve izolasyon sağlar.
- Mikrokernel mimari kalıbı, değişen sistem gereksinimlerine uyum sağlamalı yazılım sistemleri için geçerlidir. Minimal işlevsel bir çekirdeği genişletilmiş işlevsellikten ve müşteriye özel parçalardan ayırır. Mikrokernel aynı zamanda bu uzantıları takmak ve işbirliklerini koordine etmek için bir soket görevi görür.

Mikro Hizmetler Mimarisi

- Mikroservisler, SOA mimarisi ile yazılım geliştirenlerin tercih edebilecekleri bir mimaridir. Microservice Architecture olarak da bilinmektedir. Bu mimarinin de avantaj ve dezavantajları vardır.
- Büyük servisleri küçük parçalara ayırarak yönetilmesi , geliştirilmesi ve silinip atılması oldukça kolay olan bir yaklaşımı Mikroservis mimarisi geliştiriciye sunabilmektedir. Ancak küçük parçalara ayrılan ve birbirinden bağımsız servisler olarak çalışan mikro servislerin avantajlarının yanı sıra güçlükleri de bulunmaktadır. Mikroservisler iş mantıklarını ve kurallarını kendi içinde tutmalıdırlar. "Ne yapıldığı" servisler tarafında, "nasıl yapıldığı" ise servisleri kullanan uygulamalar tarafında olmalıdır.

Mikro Hizmetler Mimarisi



Mikro Hizmetler Mimarisi Avantajları

- Uzun vadeli çeviklik sağlar.
- Mikro hizmetler, her biri tanecikli ve özerk yaşam döngülerine sahip birçok bağımsız olarak dağıtılabilir hizmete dayalı uygulamalar oluşturmanıza izin vererek karmaşık, büyük ve yüksek ölçeklenebilir sistemlerde daha iyi korunabilirlik sağlar.
- Büyük servisleri küçük parçalara ayırarak yönetilmesi , geliştirilmesi ve silinip atılması oldukça kolay olan bir yaklaşımı Mikroservis mimarisi geliştiriciye sunabilmektedir

Uzay Tabanlı Mimari (SBA)

- Uzay tabanlı mimari (SBA), durumsal, yüksek performanslı uygulamaların doğrusal ölçeklendirilebilirliğine ulaşmak için kullanılan bir yazılım mimarisi desenidir . REST, SOA ve EDA unsurlarının çoğunu barındırır. Uzay tabanlı bir mimari ile uygulamalar, işleme birimleri (PU) olarak bilinen kendi kendine yeterli birimlerden oluşur. Bu birimler birbirinden bağımsızdır, böylece uygulama daha fazla birim ekleyerek ölçeklendirilebilir. SBA modeli, Google, Amazon.com ve diğer tanınmış şirketler tarafından kullanılan paylaşımsız mimari (SN) gibi uygulama ölçeklenebilirlik sorununu ele alan başarılı olduğu kanıtlanmış diğer kalıplarla yakından ilişkilidir. Bu model, menkul kıymet sektöründe ölçeklenebilir elektronik menkul kıymet ticareti uygulamaları için birçok firma tarafından da uygulanmıştır.

Kaynakça

- 2,3) https://ertankayalar.com/yazilim-mimarisi-tasarlama/
- 4) http://serhatdirik.blogspot.com/2011/04/yazlm-mimarisi-software-arhitecture.html
- 5) https://medium.com/yaz%C4%B1l%C4%B1m-mimarileri/yaz%C4%B1l%C4%B1m-mimarisi-temel-kavramlar-4de64353b4ac
- 6-12) http://serhatdirik.blogspot.com/2011/04/yazlm-mimarisi-software-arhitecture.html
- 13-17https://www.slideshare.net/aoguzhany/yazilim-mimarileriaoy,,,,,https://medium.com/kodcular/katmanl%C4%B1-mimari-9fb34ef8c376
- 18) https://www.slideshare.net/aoguzhany/yazilim-mimarileriaoy
- 19,21) http://yusufakdin.blogspot.com/2016/11/katmanl-ve-cok-katmanl-yazlm-mimarisi.html
- 22) https://slideplayer.biz.tr/slide/12307680/
- 23,24,25) https://idemarket.com.tr/blog/yazilim/en-cok-kullanilan-yazilim-mimarileri/ ,,,,,,https://docplayer.biz.tr/5692283-Bolum-9-mimari-stilleri-yzm211-yazilim-tasarimi-yrd-doc-dr-volkan-tunali-muhendislik-ve-doga-bilimleri-fakultesi-maltepe-universitesi.html
- 26,27,28,29) http://devnot.com/2015/soa-service-oriented-architecture-nedir/,,,,, https://idemarket.com.tr/blog/yazilim/en-cok-kullanilan-yazilim-mimarileri
- 30) https://idemarket.com.tr/blog/yazilim/en-cok-kullanilan-yazilim-mimarileri
- 31,32,33) https://docs.microsoft.com/tr-tr/dotnet/architecture/microservices/architect-microservice-container-applications/microservices-architecture